Applications of Machine Learning and General-Purpose GPU programming in Computational Drug Discovery

Sam Alexander Martino

A dissertation submitted is partial fulfilment

of the requirements for the degree of

Doctor of Philosophy

of

University College London

Department of Physics and Astronomy

University College London

2024

I, Sam Alexander Martino, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

.....

Abstract

Recent advances in parallel computing and machine learning are revolutionising the computational techniques used to investigate potential protein drug targets and identify compounds capable of modulating them. This work provides an overview of the context surrounding this revolution, along with an in-depth discussion of my work applying and developing these methods. It introduces the theory behind graphics processing units, machine learning, and molecular dynamics. Further work is categorised into 2 chapters, focusing on applications of these for investigating targets or screening ligands. They contain 5 Case Studies relating to various stages of drug discovery, highlighting my research contributions to the area. Target Identification and Investigation introduces methodologies used to analyse both experimental and simulated protein data, before focusing on 3 projects in this area. Case Study 1 focuses on applying variational autoencoders to learn the conformational space of BRAF activation segments from experimental structures. Case Study 2 analyses kinetic data on NSP13, a protein vital to SARS-CoV-2, and attempts to simulate the systems critical translocation function. Case Study 3 introduces a novel graph neural network based strategy able to optimise a Markov chain clustering by preserving Kemeny's constant. Large scale screening covers working with small molecules at large scales, explaining how they are processed and evaluated for potential as drugs. Case Study 4 combines several of the introduced methods to build a drug discovery pipeline aimed at finding compounds for NSP13 in the CACHE2 drug discovery challenge, resulting in experimental confirmation of a novel binder. Case Study 5 takes this pipeline and applies it to RAS, another prolific oncogene, and evaluates the beneficial performance of active learning strategies in improving ligand selection.

Impact Statement

Drug discovery is one of the most important research topics of our time. Identifying new treatments for diseases or providing safer alternatives to existing strategies aims to improve public healthcare and alleviate suffering. The contents of this work provide effective methods for improving our computational capability to both investigate drug targets and discover new small compound based treatments. It also covers practical applications of such methods to a variety of protein targets, crucial to better understanding them and fighting diseases.

This includes two targets in the MEPK/ERK pathway, a signalling cascade featuring prolific mutations across cancers. Work outlines a working basis for investigating the structural diversity of BRAF, a kinase with many unanswered questions concerning its dynamics and activation. This not only provides a method for better understanding BRAF, but also a strategy capable of generalising to other targets with clear functional motifs and many existing experimental structures. Furthering this study can help to solve current issues, such as paradoxical and monomeric activation. The other target in this cascade, RAS, is the most prolific oncogenic signalling protein known. Effective treatments for mutated RAS have so far been mostly limited to one particular cysteine mutation, and have failed to generalise to other more common mutations. For this system, a novel machine learning enabled drug discovery pipeline was constructed for a lesser treated mutation. Enabling the discovery compounds capable of a novel inhibition method, lowering the mutated barrier of GTP hydrolysis. This has a clear and simple extension to other mutations, and provides groundwork for future researchers to build upon in the hunt for a pan-RAS inhibitor.

The other major target of this work is NSP13, a helicase protein critical to the replication of the SARS-CoV-2 virus. Targeting this protein enables a

treatment strategy without many of the mutation inducing and unintended effects of those already available. It also contributes to the emerging field of using helicase inhibitors as a general technique to prevent viral replication, and offers to improve our fledgling structural understanding of the translocation process. This work provides an analysis of the dynamical behaviour of NSP13 across a large amount of simulation data, providing a kinetic basis for future studies. It additionally develops an ultimately unsuccessful strategy for simulating the translocation process, which could provide an insightful negative result for future researchers. Work on this target led to two published papers: on the dynamics and my contribution to a review summarising the helicase translocation process.

Finally, two of the included projects provide clear and novel method development contribution to the field. This includes a published novel graph neural network strategy for clustering Markov chains, providing an alternate method to extract relevant kinetic states from protein simulations. Graph neural networks are also applied to better predict a small compound's potential in drug discovery pipelines. This study demonstrates a clear improvement to selection efficiency, which can be applied to both improve performance and increase efficiency. Overall, all projects provide a foundation for future drug discovery projects.

Acknowledgements

Academically, I would like to thank Prof. Edina Rosta for giving me this opportunity, her supervision, and help with the research direction of these projects. Working in the Rosta group has introduced me to many colleagues who have fortunately turned into friends, and I would like to express my gratitude for all the members I have had the pleasure of working with. A special thanks goes to Dr. Dénes Berta for his tireless help and assistance during my studies, and Dr. João Morado for brightening up many rainy London days with great conversation. I would finally like to thank Teddy and TJ for providing countless laughs and office camaraderie.

Personally, I am blessed to have so many people who deserve thanks.
This work is dedicated to my parents Allison and Anthony. Without their unyielding and unconditional support, I would not have been able to pursue my childhood dreams. I will be eternally grateful for the freedom they have provided me, and I cherish all the love and help they still selflessly give to this day. None of this would have been possible without my Amy, who has provided me with a kind and patient love over these past years which has been my strength. It is a reciprocated love I am so lucky to have, and I know with her at my side things will always be okay. I would additionally like to thank my close family: Matthew, Christopher, and Rachel, for their help and eternal companionship.
I am honoured to have perhaps the closest support network that I know of, and I would like to take this opportunity to thank the many friends who I probably don't thank enough. To avoid picking favourites, I will let Python sort the lucky few:

> *Alex, Alice, Ben, Daisy, Elanor, Irene, Jess, Jim, Joe, Laurence, Matt, Poppy, Robbie, Tom, Tommy.*

I cherish you all and every one of you has made this possible. I look forward to the many more happy memories left to come.

"I never saw moons knew the meaning of the sea."

NICK DRAKE

Contents

Abstract **3**

Impact Statement **5**

Acknowledgements **7**

Contents **9**

List of Figures **13**

List of Acronyms **17**

Outline & Overview **21**

Introduction **25**

1 General Theory **31**

1.1 GPUs 31

1.1.1 The Demand for the GPU 32

1.1.2 The Architecture of the GPU 46

1.1.3 Coding on GPUs 50

1.2 Machine Learning 56

1.2.1 Automatic gradients on computational graphs 56

1.2.2 Neural Network Architectures & Data Modality 69

1.2.3 Practical ML 78

1.3 Molecular Dynamics 85

1.3.1 General Theory 85

1.3.2 Forcefields 87

1.3.3 Solvent 92
1.3.4 Integrators 94
1.3.5 Boundary Conditions 96
1.3.6 Ensembles 100
1.3.7 Minimisation and Equilibration 107

2 Target Identification & Investigation 109
2.1 Introduction 109
2.2 Methodology 111
2.2.1 Target Identification 111
2.2.2 Target Investigation 127
2.3 Case Study 1: Exploring the Conformational Diversity of the Activation Segment of BRAF 140
2.3.1 Overview 140
2.3.2 Background 141
2.3.3 Methodology 147
2.3.4 Results & Discussion 151
2.3.5 Conclusion & Future Work 157
2.3.6 Appendix 160
2.4 Case Study 2: The functional dynamics of NSP13 164
2.4.1 Overview 164
2.4.2 Background 165
2.4.3 Methodology 172
2.4.4 Results & Discussion 178
2.4.5 Conclusion & Future Work 186
2.4.6 Appendix 188
2.5 Case Study 3: Kemeny Constant optimization 193
2.5.1 Overview 193
2.5.2 Introduction 193
2.5.3 Theory 196
2.5.4 Methodology 201
2.5.5 Results & Discussion 208
2.5.6 Conclusion & Future Work 215

2.5.7 Appendix 219

3 Large Scale Screening 223

3.1 Introduction 223

3.2 Methodology 226

3.2.1 SMILES strings & Molecular Fingerprints 227

3.2.2 Structure Generation 239

3.2.3 Docking Software 245

3.2.4 MMGBSA/MMPBSA 257

3.3 Case Study 4: CACHE2 Drug Discovery Challenge 262

3.3.1 Overview 262

3.3.2 Background 263

3.3.3 Methodology 267

3.3.4 Results & Discussion 274

3.3.5 Conclusion & Future work 283

3.3.6 Appendix 286

3.4 Case Study 5: Active-Learning to Improve Ligand Selection 290

3.4.1 Overview 290

3.4.2 Introduction 293

3.4.3 Theory 298

3.4.4 Methodology 303

3.4.5 Results & Discussion 310

3.4.6 Conclusion & Future Work 317

3.4.7 Appendix 320

4 Conclusion 329

List of Figures

1 Overview of the full drug design process. 22

1.1 Basic Ray-traced image 40
1.2 Demonstration of AA 40
1.3 Figure showing the results of rendering a single sphere both with and without AA. 42
1.4 A two sphere render with ssAA and reflections. 43
1.5 Comparison between different renderers 48
1.6 Example of a computational graph 66
1.7 A specific example of a computational graph. 67

2.1 An example of FASTA format file. 112
2.2 The values of the BLOSUM62 matrix. 115
2.3 Example of a protein contact map. 120
2.4 Example of a scree plot 133
2.5 Graphical representation of an AE architecture. 135
2.6 Graphical representation of an VAE architecture. 138
2.7 RAF CR3 motifs. 144
2.8 ATP-APO BRAF structure comparison. 145
2.9 A schematic of the fitting process performed across the data. . . . 148
2.10 The MUSTANG structural alignment results across the PDB data. 152
2.11 PCA results across the fitted BRAF segments. 153
2.12 Visualisation of fitted BRAF activation segments. 154
2.13 BRAF fitted segment data projected onto the first two VAE latent dimensions. 155
2.14 VAE projections of fitted BRAF trajectory data. 157

2.15 PDB IDs of homologous BRAF structures 161
2.16 Structure of novel inhibitor PLX5568 162
2.17 Latent space hyper-parameter test for the BRAF VAE. 162
2.18 Homologous outliers identified by the VAE. 163
2.19 Visualisation of the RNA transcription complex. 167
2.20 The Domains of NSP13. 168
2.21 SF1B motifs highlighted on NSP13 structure. 170
2.22 A comparison between motifs in RecD2 and NSP13. 171
2.23 Models built for each stage of the NSP13 ATP cycle. 175
2.24 Schematic of the biasing force applied to the ssRNA strand in NSP13 simulations. 176
2.25 Results of DCC and PCA analysis on NSP13 monomer simulations. 179
2.26 Identified pockets during NSP13 simulations. 180
2.27 RNA-Protein contacts observed during holo simulations 181
2.28 The RNA-NSP13 contacts identified for biasing. 182
2.29 Plots of the CV values across all 6 NSP13 simulations performed . 183
2.30 Plots of the CV over NSP13 biasing simulations. 184
2.31 Comparison between the beginning and end state of forward biased apo simulations. 185
2.32 All DCC maps calculated for NSP13 monomer simulations. 188
2.33 All PCA RMSD modes calculated for monomer NSP13 simulations. 189
2.34 Calculated Residue-Pocket PCCs for dimer simulations. 190
2.35 Calculated Residue-Pocket PCCs for monomer simulations. 191
2.36 $\pi - \pi$ stacking observed in biased NSP13 backwards simulations. . 192
2.37 Distances to Domain1B from both RecA domains during the forward biased NSP13 simulations. 192
2.38 Architecure of the GNNs used for KC optimization 205
2.39 Schematic of the training process used for parallel GNN training . 206
2.40 GraphSAGE + LD GNN performance on SBM model networks . 209
2.41 GraphSAGE + LD feature comparison on the 1D network example 211
2.42 Comparison between all GNNs on the 1D Example. 212
2.43 Comparisons of the optimal clusterings and GraphSAGE + LD features. 214

2.44 Minimum DBI values for optimal clusterings found by each method, and pentapeptide GNN comparison. 215
2.45 Pentapeptide example hyper-parameter search results. 216
2.46 ΔK values found by GraphSAGE + LD and PCCA+ on four SBM networks. 219
2.47 DBI values corresponding to clusterings found for the 1D example. 220
2.48 Equilibrium probability distributions for the pentapeptide example. 221
2.49 A comparison of the optimal ΔK clusterings found by each GNN architecture. 221

3.1 Outline and Timeline of the CACHE2 Challenge 266
3.2 Crystallographic fragments showing binding to the RNA cavity from PANDDA study. 268
3.3 Comparison of docking sites used. 270
3.4 Comparison between QM regions for QM/MMGBSA calculations used for the different pockets 273
3.5 Comparison of identified crystallographic fragments. 275
3.6 Comparison of fragment docking scores attained with AutoDock Vina and AutoDockGPU 276
3.7 Comparison between the docking scores for the top 25% of molecules in SS1. 277
3.8 QM/MMGBSA Results from the first round 278
3.9 RNA and stalk docked structures of the best experimentally performing molecule from SS1. 279
3.10 Comparison of the docking scores and QM/MMGBSA predictions attained for all groups round 1 molecules. 281
3.11 QM/MMGBSA distribution for SS2 molecules. 282
3.12 Correlations across various molecular properties and docking results for SS2. 286
3.13 Molecular diagrams for the round 1 selected molecules. 291
3.14 Molecular diagrams for the 75 Round 2 selected molecules. 292
3.15 RAS-GAP complex with GDP bound. 296
3.16 The GNN architecture used for RAS Barrier Prediction. 308

3.17 Visualisation of the known binders bound to RAS. 309
3.18 Errors for the different featurizations tested. 311
3.19 Scatter plot of the QM/MM calculated barrier against the best GNN model training using internal distances features. 312
3.20 Vina docking score distribution across found similar ligands. . . . 314
3.21 Histogram of QM/MM calculated barrier heights for the 10,656 tested ligands. 315
3.22 AL enrichment over a pure docking strategy calculated for all values of N tested. 315
3.23 Selection efficiency comparison between the GNN with $N = 750$, docking score, and a random selection. 316
3.24 Location of the Switch I and Switch II pockets in RAS 320
3.25 Free energy barrier calculated for RAS hydrolysis 321
3.26 Selected RAS-GAP complex residues for GNN features. 321
3.27 Tanimoto Similarity between known RAS inhibitors. 322
3.28 Molecular diagrams for the 5 RAS binders used for similarity searching. 322
3.29 Performance of the ChemProp baseline model on the ZINC dataset. 323
3.30 Centre of geometries for all ligands in the ZINC dataset. 323
3.31 Centre of geometries for all ligands in the ZINC dataset alongside preliminary Vina docking results. 324
3.32 The averge MSE errors for all 10 GNN layer types across the training and test sets. 325
3.33 The PCC values between predicted and calculated barrier heights for all 10 GNN layer types across the training and test sets. . . . 326
3.34 Ligand selection for $N = 750$ using the GNN model to predict barriers. 327
3.35 Ligand selection for $N = 750$ using docking scores to evaluate ligands. 328

Acronyms

+ssRNA	positive-sense single stranded RNA	165
AAs	Amino-acids	113
ADP	Adenosine diphosphate	174
AEs	Autoencoders	134
AFE	Alchemical Free Energy	257
AI	Artificial Intelligence	28
AL	Active Learning	294
AM1	Austin Model 1	250
AMBER	Assisted Model Building with Energy Refinement	87
API	Application Programming Interface	31
ATP	Adenosine triphosphate	141
BAC	Biologically Active Compound	21
BCC	Bond Corrective Charge	250
BLAST	Basic Local Alignment Search Tool	117
CHARMM	Chemistry at Harvard Macromolecular Mechanics	87
CNNs	Convolutional Neural Networks	72
CPU	Central Processing Unit	31
CUDA	Compute Unified Device Architecture	31
CV	Collective Variable	129
DAG	Directed Acyclic Graph	65
DBI	Davies-Bouldin index	200
DCC	Dynamic cross-correlation	130, 173

DDP	Drug Discovery Pipeline	224
ECFPs	Extended-Connectivity Fingerprints	235
ELBO	Evidence Lower Bound	136
EM	Electromagnetic	90
ESPs	Electrostatic potentials	305
FEP	Free Energy Perturbation	257
FES	Free-energy Surface	127
FF	Force-field	87
GAPs	GTPase-activating proteins	295
GAT	Graph Attention Network	78
GCN	Graph Convolutional Network	77
GNNs	Graph Neural Networks	76
GP	graph partitioning	194
GPGPU	General-Purpose GPU	29
GPUs	Graphics Processing Units	29
GRU	Gated Recurrent Units	73
GTP	Guanosine triphosphate	142
HepC	Hepatitis C	264
HPIs	Helicase Primase Inhibitors	264
HSV	Herpes Simplex Virus	264
HTS	High Throughput Screening	226
JIT	Just-in-time	32
JVP	Jacobian-Vector Product	62
KC	Kemeny constant	195
KL	Kullback-Leibler	136
LD	Linear Decoder	205
LJ	Lennard-Jones	91
LLMs	Large Language Models	57
LO	Lead-Optimization	224
LSTM	Long Short-Term Memory	73
MAPK	Mitogen-Activated Protein Kinase	141
MCs	Markov chains	193

MD	Molecular Dynamics	85
MFPTs	Mean First Passage Times	195
ML	Machine Learning	27
MM	Molecular Mechanics	87
MM/GBSA	Molecular Mechanics with Generalized Born and Surface Area solvation	258
MM/PBSA	Molecular Mechanics with Poisson-Boltzmann and Surface Area solvation	258
Mpro	Main protease	165
mRNA	Messenger RNA	165
MSA	Multiple Sequence Alignment	125
MSE	Mean-Squared Error	134
MSMs	Markov State Models	194
NLP	Natural Language Processing	26
NNs	Neural Networks	27
NSPs	Non-Structural proteins	165
NTP	Nucleoside triphosphate	166
PBCs	Periodic Boundary Conditions	96
PCA	Principal Component Analysis	131
PCC	Pearson Correlation Coefficient	130
PDB	Protein Data Bank	26
PME	Particle Mesh Ewald	99
PMF	Potential of Mean Force	250
PTVC	Parallel Tempering Variational Clustering	208
QM	Quantum Mechanics	87
QM/MMGBSA	QM-enabled MM/GBSA	261
RAF	Rapidly Accelerated Fibrosarcoma	142
RCs	Reaction Coordinates	129
RdRp	RNA-dependent RNA polymerase	166
RMSD	Root-Mean Square Deviation	119
RNNs	Recurrent Neural Networks	73
RTC	RNA-transcription complex	165

SARs	Structure–Activity Relationships	224
SARS-CoV-2	Severe Acute Respiratory Syndrome Coronavirus-2	165
SASA	Solvent Accessible Surface Area	259
SBM	Stochastic Block Model	207
SGD	Stochastic Gradient Descent	82
SIMD	Single-Instruction Multiple Data	29
SM	Streaming Multiprocessor	50
SMILES	Simplified Molecular Input Line Entry System	227
SPR	Surface Plasmon Resonance	266
SS1	Similarity Search 1	274
SS2	Similarity Search 2	280
ssAA	Super-sampling Anti-Aliasing	40
TD	Transformer Decoder	204
TIP	Transferable Inter-molecular Potential	93
VAEs	Variational Autoencoders	135
VdW	van der Waals	90
VJP	Vector-Jacobian Product	62
VS	Virtual Screening	226
ZBD	Zinc Binding Domain	168

Outline & Overview

Historically, the majority of new drugs have been synthesised from natural agents or found via a serendipitous process of trial and error. This process has relied on experimentally testing compounds until a desired therapeutic effect is observed. Now, equipped with a deeper understanding of biological systems, researchers look to design drugs from first principles by targetting specific biological targets to inhibit or enhance known pathological cellular functions. This field, known as rational drug design, is inherently interdisciplinary and requires a plethora of experts from different backgrounds to ensure a complete and efficient process. Teams often comprise of a plethora of technical experts, such as biochemists, molecular biologists, toxicologists, geneticists, cell biologists, biophysicists, physiologists, pathologists, to name just a few. For this reason a comprehensive understanding of all areas surrounding the process is rare, and even contributors to the field have their own personal expertise.

While experimental work is the definite standard to measure the efficacy of a Biologically Active Compound (BAC), it is far too time-consuming and resource intensive to use on the number of compounds required to check in a modern drug hunt. Computational methods have emerged as a mainstay of modern pipelines in order to save time and resources for those compounds which display the most potential. The challenge of modern drug design is scaling up investigative methodologies to the vast size of chemical space, and improving the correlation between computational methods and empirical results. This work focuses on the computational methods that have emerged as key tools to aid our understanding of diseases and search for new drug-like compounds. It specialises in how computational scientists can aid in both investigating potential drug targets and finding small ligand molecules to modulate proteins for therapeutic benefit.

Alongside this it attempts to highlight the great benefit modern computational hardware and machine learning modelling can provide in this area to help scale.

As the driving force between much of the data-driven focus of this area, this work begins with an overview of the theory behind both graphics cards and modern machine learning. Some of the context of this work will be discussed in the following introduction, followed by an in-depth overview of both and the widely used molecular dynamics in Chapter 1. This begins with a self-contained practical exploration of the problems fuelling the initial development of GPUs in Section (1.1), using ray-tracing as an example of the class of issues which required a new level of parallel computing. It also introduces the GPU architecture, and looks at how this can be used and exploited by modern science for heavy numerical computation. Subsequently this turns to introducing modern machine learning as the beneficiary of these advancements in Section (1.2). It takes an implementation focused overview, looking at automatic differentiation and discussing some of the recent progress in the area, alongside a discussion of practical considerations. This is followed by a comprehensive overview of the techniques required to run modern simulations of biomolecules in Section (1.3). Forming the backbone underpinning many of the key methodologies used in this thesis, this section outlines all the necessary considerations required for realistic and useful simulations.

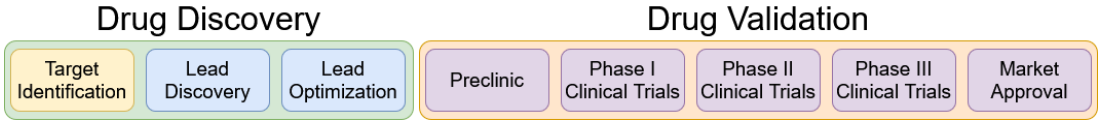


Figure 1: Figure illustrating some of the major steps performed during the investigation of potential drugs. This thesis focuses on "Drug Discovery", with the topics in yellow being the focus of Chapter 2 and the topics in blue being discussed in Chapter 3.

It subsequently moves to drug design, the focus of the work performed for my PhD. The overall process is split into two parts- discovery and validation- and an outline of these steps is provided in Fig. (1). The latter part focuses on the experimental work in the pipeline- encompassing initial validation of the effects of a small molecule, later clinical trials, market approval, and other important facets to the overall process. These are critical, but happen far downstream of the contents of this work. Instead this document focuses at the start of this process,

on the discovery of effective targets, relevant structures, and small BACs capable of interfering with biological processes.

Chapter 2 highlights how computational scientists can use alignment algorithms, biomolecular simulations, and numerical analysis techniques to help with target identification and investigation. The general methodologies for these projects are outlined in Section (2.2). This splits the work into two topics of equal importance. The identification topic, carried out by biologists and bioinformaticians, involves working with sequence and structure databases to mine relevant useful information. Investigation instead looks at elucidating results from simulated systems via effective dimensionality reduction procedures.

This then turns to 3 case studies exemplifying different approaches to this field of study. Case Study 1 investigates the oncogenic signalling kinase BRAF. It looks to use modern machine learning strategies to better understand the conformations available to a small loop-like segment, involved in the active to inactive structural change. Case Study 2 focuses on NSP13, the helicase of the SARS-CoV-2 virus. It presents an analysis of several long simulations of the system, alongside a potential strategy for simulating its translocation process. Case Study 3 is a method development project introducing a novel clustering algorithm for Markov Chains, used to process simulations using a more state-based perspective.

Once a target has been successfully identified, the next stage is covered in Chapter 3. Here the aim is to generate physical structures for the protein with small molecules bound, and evaluate their potential as a BAC via large scale screening protocols. Methods for this are introduced in Section (3.2). This will mainly focus on working with small molecules, taking the reader from finding a structure with a suitable pocket using the information from the previous section, to running a small molecule through parameterization, docking, and options available to better estimate the binding affinity of the complex. It has a heavy focus on computational methodologies for processing large numbers of ligands, and looks to provide both an overview of available methods and how they can be combined together to form a cohesive strategy for drug design pipelines.

The two Case Studies of this section showcase a very large scale approach to screening ligands, alongside the assessment of a novel ML strategy for improving this process. Case Study 4 returns to NSP13, instead to look for ligands capable

of binding to the protein. This focuses on my contributions to the CACHE2 drug discovery challenge, culminating in experimental procedure to check my groups proposed compounds. Case Study 5 develops upon this strategy by reapplying it to RAS, another oncogenic signalling protein, and implementing a novel active learning strategy to improve ligand selection efficiency. It focuses on developing a test system to quantify the benefit this introduction can have on selection efficiency.

Finally Chapter 4 summarises many of the key conclusions from these projects.

Introduction

While the fundamental principles of the scientific method are unchanging, the paradigm researchers use to investigate the physical world is dynamic[1]. Historically science has progressed as the elegance of human thought developed intricate models to bring clear order to measurements of physical phenomena[2,3]. Disagreements within the community were a fundamental and regular part of the process, inspiring passionate defences of models by their proponents and giving an increased vigour to experimentalists[4]. Corroboration with experiments provided countless harsh truths to many proponents of various theories and models, who often found their life's work lost to historical textbooks or had to reconcile their intuition about the nature of the physical world. There was no aether, no miasma, and god does indeed play dice with the universe[5–7]. While there was and still is a necessity for cynicism about the nature of any performed experiments, over time the correct results were eventually accepted, reproduced, and remembered by the community. The impetus was then upon the theorists to know when to use their instincts and when to put their belief into the evidence they were presented with, no matter how unintuitive it may seem, to create a cohesive rational system. Only with an informed perspective could the next set of hypotheses begin their path to becoming scientific theories, and new experiments constructed to verify them. By the time computers started to become commonplace in research, this process of attaining knowledge had been continually refined and abstracted. It was hoped the emergent technology could be leveraged to solve the many still unsolved problems across all scientific disciplines[8]. With this goal, computerised systems began to accumulate and store measurements at a rate once unimaginable, and a metamorphosis of the scientific strategy began.

Jumping forward to the modern context surrounding this work reveals a different approach to scientific development. The afterglow of the computing revolution has brought with it a revised focus on incremental advances, leading to a shift in how the scientific community approaches many of the remaining difficult problems[9]. For many of these, models providing an ab initio understanding are no longer the main goal. Perhaps these issues are too complex to expect elegant or intuitive solutions, and so instead the focus has become more empirical and moved towards measuring as much as possible. All measurements have been abstracted to **data**, the common currency across modern science[10,11]. Quantised bits of information stored in machines, held in an almost infallible regard as information the community can agree upon. A significant proportion of the modern scientific workload is now the collection, consolidation, and processing of data into both public and private datasets[12–14]. Modern computing power has allowed this to reach new extremes and there are countless examples of massive publicly available data resources. Text corpora used for Natural Language Processing (NLP) regularly reach over 800GB, the Large Hadron Collider produces around a petabyte of data a second, public captioned image datasets extend to over 12 million examples, and satellite imagery datasets often reach terabytes to name just a few[15–18]. Data has become the core focus of a large amount of research, far surpassing the models which have become increasingly iterative and unintelligible. In this new world of data abundance, the greatest challenge has been finding strategies capable of effectively using all of the information available.

The field of structural biology has felt this change in focus dramatically. It is intrinsically an area with an abundance of data waiting to be collected, from 3D-structures, primary sequences, and functional annotations amongst others. Recent advances in experimental techniques and algorithmic labelling have led to a massive increase in the rate of information production[19–23]. At the time of writing, there are 227,344 protein structures available on the Protein Data Bank (PDB), with over 10,000 added just last year. This is a far-cry away from its humble beginnings in 1976 when a mere 13 structures were deposited[24]. The UniProtKB database provides researchers with 248,838,887 non-redundant protein sequences and functional annotations in one central repository[25,26]. Additionally researchers in the area often sit on large quantities of simulation data, routinely generating

gigabytes of dynamic trajectories which are being pushed to be made more available[27]. Computational rational drug design provides a practical application for much of this data, as researchers look to find small compounds to improve our ability to treat disease. Small molecule databases, such as ChEMBL and Zinc, offer billions of potential ligands with the aim of finding potential compounds with therapeutic properties[28,29]. Screening these based on structural or chemical criteria to find selective inhibitors has shown to be a viable approach across a variety of systems, and looks to revolutionise our capabilities to fight previously untreatable diseases[30–32]. However such strategies have long struggled from issues of scale, as with the current data zeitgeist the amount of information available is far too much to check, compare, and analyse by hand. The solution to this has been an increased reliance on automated and algorithmic methods to develop comprehensive models capable of covering all observed edge-cases. Interestingly, it is only comparatively recently that models capable of scaling to the amount of data while retaining sufficient complexity are making rapid progression[33–35]. Age-old problems such as protein structure prediction, increasing feasible simulation time-scales, and ligand generation have all received heightened attention in the past decade, with significant progress being made towards usable empirical solutions[36,37]. Each one of these problems requires processing an amount of data which would have been thought infeasible even two decades ago, providing clear examples of the successes of the data-driven paradigm. Still, it raises questions about what has enabled this new level of data processing to occur.

If there is one term in current scientific parlance which can rival the prominence of data, it is Machine Learning (ML). As the name suggests, it has emerged as the term encompassing the dominant approaches to automating the core scientific principles across the sciences and beyond[38]. If the scientific community as a whole studies abstract data to form hypotheses about the behaviour of complex systems, these strategies look to develop algorithms capable of doing the same. Neural Networks (NNs) have become the central approach to performing many of these complex modelling tasks required, and have leapt to prominence over the past decade due to their non-linear modelling capabilities[39]. The development of these methods is often seen as the core driver of the great acceleration in progress on many of these difficult remaining problems. However with all the

recent excitement, it is easy to forget that the core principles underpinning this technology is nothing new. Early literature on the subject was first published using the Artificial Intelligence (AI) nomenclature in the 1950s, and the base mathematical theory underpinning most modern optimisation algorithms goes back as far as Leibniz[40,41]. These early attempts were often too inefficient and ineffective for practical applications, struggling to scale to the magnitude of data required, and rarely moved beyond theoretical intrigue. Additionally there was much lament over the "Curse of dimensionality" that plagued research, a phrase coined in the 1950's to explain much of the difficulty faced when fitting various statistical models to high dimensional datasets[42]. The relatively low density of data-points in such sets meant models often struggled to accurately estimate probabilities. Research in automating reasoning stalled and this led to the proclaimed "AI winter" of the 1970s, where the Lighthill report in 1971 stated[43]:

> In no part of the field have the discoveries made so far produced the major impact that was then promised.

Funding for the area fell, and many of these non-linear modelling theories looked to become another victim of the historical textbooks.

Instead, the current situation could not be more different. Far from their historic place of being bound to lecture halls and academic textbooks, the same techniques are commonplace to the point of being discussed *ad nauseam* in both technical and non-technical environments. To the layman, the phrases "*AI*", "*Machine Learning*", and "*Neural Networks*", have become an interchangeable nomenclature for describing the cutting edge of these methods as they begin to impact society. These terms have moved outside of the realm of science to become a newspeak to explain a world significantly impacted by largely unintelligible algorithmic models. While this vernacular exists outside the realm of technical terminology, it has become popular enough to be used throughout the literature on the subject by practitioners as the uptake of these methods increases. Cutting through the hyperbole there is little revolutionary theory that would indicate the presence of anything beyond statistical modelling techniques finally being able to resolve some of its problems of scale, linearity, and dimensionality. The

interesting question then still remains: what has enabled structural biology to finally take advantage of so much data, ML to break free of the constraints of scale and dimensionality, and general science to become so reliant on a data-focused approach?

The answer to these questions comes from a very unexpected place. Away from the physical sciences, there was a disparate field encountering the same problems of performing numerical processing tasks at huge scales from a very different perspective. Between the Lighthill report and the recent ML fervour, the development of computer graphics has too accelerated at an unrelenting pace[44]. The move from scan-line screens to fully rendered 3D environments took little more than two decades[45]. Driven by consumer demand for better visuals in films and video-games, market competition led to the profound development and adoption of Single-Instruction Multiple Data (SIMD) hardware architectures. Known as Graphics Processing Units (GPUs), they are purpose-built and meticulously crafted for these tasks, embodying the core principles of SIMD[46,47]. Initially responsible for offloading the rendering of complex and realistic 3D-scenes from CPUs, these have seamlessly transitioned from specialised units into a fundamental component of everyday computing[48,49]. In recent decades, the scientific community has began exploiting their pervasiveness and found an incredibly powerful tool enabling a significant boost to numerical data processing and especially ML model performance. It is impossible to separate their repurposing for general purpose calculations and the solutions of many of the aforementioned computational difficulties of scale faced by the big-data revolution. This transition from graphics-oriented tasks to a broader computational scope epitomises the adaptability and efficiency of SIMD hardware in tackling diverse and complex numerical computational challenges which has been key in its widespread adoption. Its design is particularly well-suited for the parallel tasks inherent in linear algebra and floating-point arithmetic, critical for a wide range of simulation, data analysis, and ML training or inference tasks encountered in modern science.

At the heart of this thesis lies the intricate relationship between the rise of General-Purpose GPU (GPGPU) computing and the remarkable evolution of modern ML and GPU data-processing techniques in the field of computational drug design. It focuses on how these technological advances have enabled the area

to develop at an incredible pace due to the long-suffered issues of scale which can now be tackled effectively. Harnessing the potential of the aforementioned large datasets available, enabling quicker generation of processed data, and providing the unifying thread between all projects worked on during my PhD. Despite this revolutionary capability even to many even technical end-users in science, its usage and prominence can easily go unnoticed. For computationalists working in drug design, working at great scales and training ML models are quickly becoming everyday tasks, and as such an in-depth understanding of this technology is of paramount importance to being able to work at the necessary pace and scale. Therefore this work shall begin with a relatively self-contained overview of the GPU, introducing readers to both an example of the initial demand for new hardware and the general architecture. This is to highlight how it has been repurposed by computational scientists for large scale numerical work, and showcase the unexpected parallels between issues faced in both graphics and modern science.

Chapter 1

General Theory

1.1 GPUs

To understand why GPUs enable modern algorithms and help scientists process gigantic biological and chemical datasets, some comprehension of the development of the hardware is required. GPUs are at the heart of the current computing revolution as they unlock using the quantities of data modern science produces, at speeds thought impossible just a few decades prior. Despite this, as their name suggests they were originally designed for purposes very different to general scientific computation. This section is relatively self-contained, and aims to give interested readers an insight into this prevalent technology and how it has revolutionised many fields by drawing parallels between the problems faced by data-driven science and early rendering.

To begin, we will focus on introducing the class of problems fuelling the initial consumer demand for massively parallel hardware by summarising an initial major use case for GPUs in graphics. To see how they impacted the hardware design of GPUs, a basic ray-tracing implementation will be constructed as an example. This will cover some of the necessary code in C++ to produce an image on the CPU, before later looking at how this could be ported to the GPU using NVIDIA's proprietary Compute Unified Device Architecture (CUDA) C++ API[50]. This is to highlight how the hardware follows on from solving performance issues first realised through the development of advanced rendering algorithms. After this, the focus briefly turns to the hardware implementation, before covering the

tooling of modern GPUs and diving into a brief overview of some of the practical coding aspects. Investigating how researchers can use GPUs at a low-level in the C++ language using CUDA directly, and the more common approach at a higher level via Python bindings and Just-in-time (JIT) compilation libraries. This will showcase the impressive ability of the GPGPU to enable high-throughput speeds with little to no burden on the user to change their code or deal with directly interfacing with the card itself. The current state of this technology is that it requires almost zero specific knowledge to use as part of many modern software libraries, and there are numerous implementations available for a wide variety of applications. However in order to use them effectively and to implement cutting edge techniques, it is required to know the basics of how GPUs work as part of a modern heterogeneous computing architecture. Understanding these initial applications offers a good introduction as to why the hardware evolved the way it has, showcasing how this was then able to be taken advantage of by research programmers, and displaying the limitations of the hardware.

1.1.1 The Demand for the GPU

The introduction to this work discussed the importance of GPUs as accelerators for scientific programming, and this section takes a short detour to the world of graphics to study why they first came to exist. The development of computing systems is a two-fold process. On one side there is the hardware, the physical implementation of a computer, and on the other is the software, determining how everyday users interact with the system. These are interlinked, and progress is staggered as new developments in one emerge to meet the demands or capabilities set by the other. GPU origins lay firmly in the software section, where creatives were learning how to harness computing technology to develop new entertainment experiences. Producing 2D-images for display on a screen requires the usage of rendering algorithm, or renderers, able to take a specification of some physical topology and project that onto a viewing plane.

At their core, most renderers are clear examples of embarrassingly-parallel problems to some extent, as determining the colour of a specific pixel is independent of any other. The architecture of modern cards are still very much based on

solving problems like this with degrees of parallelism at their core, and GPUs fail to provide any acceleration on problems requiring sequential evaluation. The power of the GPU becomes abundantly clear when considering a graphics algorithm such as ray-tracing, which serves as one of the most intuitive and earliest rendering algorithms[51–53]. On top of being one of the most basic and effective methods, it shall become clear how this seemingly specific use-case is very open to generalisation, and mimics much of the work required for ML and data processing.

Ray-Tracing

Ray-tracing is a rendering algorithm for generating an image of a static scene from a camera, by calculating the paths of rays of light as they traverse through a given geometry. Expressing vector quantities using a bold typeface, all possible rays in a given $3D$-scene in Cartesian coordinates can be expressed using linear functions of the form:

$$\mathbf{P}(t) = \mathbf{A} + t\mathbf{d} \tag{1.1.1.1}$$

Where $\mathbf{P}$ is a position on the ray parametrised by scalar t, $\mathbf{A}$ is a position along the ray, and $\mathbf{d}$ is a vector pointing in the direction of the ray. For positive real values of $t \in \mathcal{R}$, this form summarises the whole path of any ray of light which hit the point A before being reflected in direction d.

The simplest geometries to work with are spheres, and so shall serve as the considered example.
Points on a sphere centred at position $\mathbf{C} = (C_x, C_y, C_z)$ with radius r satisfy the equation:

$$(C_x - x)^2 + (C_y - y)^2 + (C_z - z)^2 = r^2$$

This can be reformulated in terms of points on rays $\mathbf{P}(t)$ which fulfil the condition:

$$(\mathbf{C} - \mathbf{P}(t)) \cdot (\mathbf{C} - \mathbf{P}(t)) = r^2 \tag{1.1.1.2}$$

Directly substituting in Eq. (1.1.1.1) allows for the solutions to be expressed as a quadratic in t:

$$t^2\mathbf{d} \cdot \mathbf{d} - 2t\mathbf{d} \cdot (\mathbf{C} - \mathbf{A}) + (\mathbf{C} - \mathbf{A}) \cdot (\mathbf{C} - \mathbf{A}) - r^2 = 0 \tag{1.1.1.3}$$

Which can then be solved using the quadratic equation, often reformulated with $b = 2h$ for computational simplicity:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \rightarrow \frac{h \pm \sqrt{h^2 - ac}}{a}$$

The discriminant of this polynomial, $h^2 - ac$, gives us the number of intersections. A negative discriminant means the ray misses the sphere completely, a discriminant of zero means a tangential intersection, and a positive discriminant means there are two intersection points at the front and back of the sphere. Computing the intersection of any ray and sphere then requires calculating the following:

$$a = \mathbf{d} \cdot \mathbf{d} \tag{1.1.1.4}$$

$$h = \mathbf{d} \cdot (\mathbf{C} - \mathbf{A}) \tag{1.1.1.5}$$

$$c = (\mathbf{C} - \mathbf{A}) \cdot (\mathbf{C} - \mathbf{A}) - r^2 \tag{1.1.1.6}$$

When there are two solutions, choosing the negative sign ensures the fist intersection on the side closest to the ray origin is taken. A potential intersection should be rejected if it is behind the origin of the ray A relative to the direction d, which is clear if the root is negative. There are 3 required classes thus far: for representing rays, spheres, and recording the points of intersections between the two. Listing 1 shows the ray, sphere, and hit class for these respective purposes.

Now the geometry is formalised, a very simple ray-tracing engine can be formed by considering a basic pinhole camera. In real life, light generated by any light-source reflects off a given geometry before being reflected into a camera, interfering to form an image of a scene. Computationally, it is possible to save substantial effort by considering that the number of rays entering the camera is significantly smaller than all the rays present in a scene. It is possible to only consider the subset that enters the camera by working backwards, propagating light from the camera towards the scene. Mathematically modelling this camera consists of adding a single point to the scene where all light emanates from, and placing in-front a $2D$-plane called the view-port where the image is captured[i].

[i] In an actual pinhole camera, the view-port is in the opposite direction and forms the image upside down. In a computational model however, this can be ignored.

```
class ray {
public:
  vec3 A;
  vec3 d;

  ray(){};
  ray(vec3 origin, vec3 direction) {
    A = origin;
    d = direction.norm();
  }
  vec3 at(float t) { return A + d * t; }
};

class hit {
public:
  vec3 p;
};

class sphere {
public:
  vec3 centre;
  float radius;

  sphere(){};
  sphere(vec3 c, float r) {
    centre = c;
    radius = r;
  };
  // To determine if ray hits sphere
  // Return true if does, false if doesn't,
  // and update record with the hit details.
  bool hit_sphere(ray r, hit &record) {
    vec3 oc = centre - r.A;
    float a = r.d.mag();
    float h = r.d.dot(oc);
    float c = oc.mag() - radius * radius;
    float discriminant = h * h - a * c;

    if (discriminant < 0) { // No intersection
      return false;
    } else { // Intersection
      float root = (h - std::sqrt(discriminant) / a);
      if (root < 0) {   // If the intersection is behind A relative to the
                        // direction of travel of the ray then reject it.
        return false;
      }
      record.p = r.at(root);
      return true;
    }
  }
};
```

Listing 1: Ray and Sphere Classes in C++

Using terminology from photography, the distance from the view-port to the camera is called the focal length f, and the projection of light from the scene onto this plane will become our final image. The coordinates on the plane form the image space M, where the image is separated into $(U \times V)$ sections, called pixels, which define the resolution of our image. Each ray which reaches this plane is a pixel of the camera's image, emanating from the camera location and travelling towards the geometry.

Before putting this section into code, some conventions must be outlined. The camera shall be placed at $C = (0, 0, 0)$ with $f = (0, 0, 1)$ and the convention of right-handed coordinates will be used where the x-axis is horizontal, y-axis is vertical, and z-axis points in the viewing direction. The size of the view-port will be kept constant as a 2×2 plane, with the resolution being variable. Space I will consist of $U \times V$ points $p \in \{(0, 2), (0, 2)\}$, with the origin $M_{00} \in M$ being in the top-left corner. Each pixel will be a $\Delta u \times \Delta v$ square, and the pixels ray will be taken from the centre of this square. These constants can be calculated as:

$$\Delta u = \frac{2}{U} \tag{1.1.1.7}$$

$$\Delta v = \frac{2}{V} \tag{1.1.1.8}$$

$$M_{00} = C - f - \left(\frac{\Delta u \cdot U}{2}, \frac{\Delta u \cdot U}{2}, 0 \right) \tag{1.1.1.9}$$

Again these can be stored in a class outlined in Listing 2. Ignoring the file IO code, three more things remain before this code can produce an image.
Firstly a colour for the spheres is required. A basic colouring scheme by extending the `sphere::hit_sphere` function to collect the unit normal $\hat{n}$ at the point a ray intersects the sphere by extending the hit class to record this. Then giving this record to the new `normal_material` function returns a colour for this point based on its angle relative to the sphere origin. Both additions are outlined in Listing 3. Next a topology must be generated. For ease of use, a `std::vector` of spheres can be generated, allowing for extension to an arbitrary number of spheres and simplifying iterating over them. To produce a simple single sphere scene, a simple function returning a vector can be used as shown in Listing 4. Finally, introducing a `camera::render` method to perform the final rendering of a given

```
class camera {
public:
  int image_width;
  int image_height;

  camera(int width, int height) {
    image_width = width;
    image_height = height;
  }

private:
  vec3 centre;
  vec3 M00_loc;
  vec3 delta_u;
  vec3 delta_v;

  void initialize() {
    centre = vec3(0, 0, 0);

    float focal_length = 1.0;

    vec3 viewport_u = vec3(2.0, 0, 0);
    vec3 viewport_v = vec3(0, -2.0, 0);

    delta_u = viewport_u / image_width;
    delta_v = viewport_v / image_height;

    // Calculate the location of the upper left pixel.
    vec3 viewport_upper_left = centre - vec3(0, 0, focal_length)
                               - viewport_u / 2 - viewport_v / 2;
    M00 = viewport_upper_left + (pixel_delta_u + pixel_delta_v) * 0.5;
  }
};
```

Listing 2: Camera class in C++.

vector topology, and a `camera::ray_colour` function which returns the sphere colour if an intersection occurs otherwise giving a background gradient. All this is added to a main function, linking together all previous parts shown in Listing 5. Fig. (1.1)(a) shows the output image of this code, which successfully produces a rendering of a sphere. However Fig. (1.1)(b) shows that this basic implementation suffers from a lack of sharpness around the boundary of the sphere. This is due to the ray-generation strategy of taking each pixel as the centre of its defining region, causing the observed boundary of the sphere to be discrete. In reality, as human vision does not work on a grid-like pixel system, each section of our sight is comprised of an average of many light rays interfering as they hit the retina. This can be considered a consequence of the more general issues caused by sampling

```
class hit {
public:
  vec3 p;
  vec3 normal;
};

class sphere {
...
  // To determine if ray hits sphere
  bool hit_sphere(ray r, hit &record) {
    vec3 oc = centre - r.A;
    float a = r.d.mag();
    float h = r.d.dot(oc);
    float c = oc.mag() - radius * radius;
    float discriminant = h * h - a * c;

    if (discriminant < 0) { // No intersection
      return false;
    } else { // Intersection
      float root = (h - std::sqrt(discriminant) / a);
      if (root < 0) {   // If the intersection is behind A relative to the
                        // direction of travel of the ray then reject it.
        return false;
      }
      record.p = r.at(root);
      vec3 hit_normal = (record.p - centre) / radius;
      record.normal = hit_normal;
      return true;

      float root = (h - std::sqrt(discriminant) / a);

      record->t = root;
      record->p = r.at(root);
      vec3 hit_normal = (record->p - centre) / radius;
      record->set_face_normal(r, hit_normal);
      return true;
    }
  }
};

inline vec3 normal_material(hit record) {
  return (record.normal + vec3(1, 1, 1)) * 0.5;
}
```

Listing 3: Extending the hit and sphere classes to record the surface normal.

```
std::vector<sphere> generate_topology() {
  std::vector<sphere> topology;
  topology.push_back(sphere(vec3(0, 0, -5), 2.5));
  return topology;
}
```

Listing 4: Function to generate a basic topology consisting of a single sphere.

```
class camera {
public:
...
  void render(FILE* output_file, std::vector<sphere> topology) {
    initialize();
    for (int j = 0; j < image_height; j++) {
      for (int i = 0; i < image_width; i++) {
        vec3 pixel_centre = M00 + (delta_u * i) + (delta_v * j);
        vec3 ray_direction = pixel_centre - centre;
        ray r(centre, ray_direction);
        vec3 pixel_color = ray_colour(r, topology);
        // Additional code for file output omitted
        ...
      }
    }
  }

private:
...
  vec3 ray_colour(const ray &r, std::vector<sphere> topology) {
    hit *hit_r;
    for (sphere s : topology) {
      if (s.hit_sphere(r, hit_r)) {
        return (hit_r->normal + vec3(1, 1, 1)) * 0.5;
      }
    }
    // Generate background colour
    vec3 unit_direction = r.d;
    float a = 0.5 * (unit_direction.y + 1.0);
    vec3 result = vec3(1.0, 1.0, 1.0) * (1.0 - a) + vec3(0.5, 0.7, 1.0) * a;
    return result;
  }
};

int main() {

  // Generate topology
  std::vector<sphere> topology = generate_topology();

  // Image settings
  ...
  int width = 800;
  int height = 800;
  camera cam(width, height);

  cam.render(output_file, topology);
}
```

Listing 5: Additions to the code to generate a basic image.

data below the Nyquist frequency, as the resolution of the image is incapable of adequately representing the frequency spatial fluctuations[54,55]. Anti-aliasing is a technique that has emerged to combat distortions caused by this, and draws clear

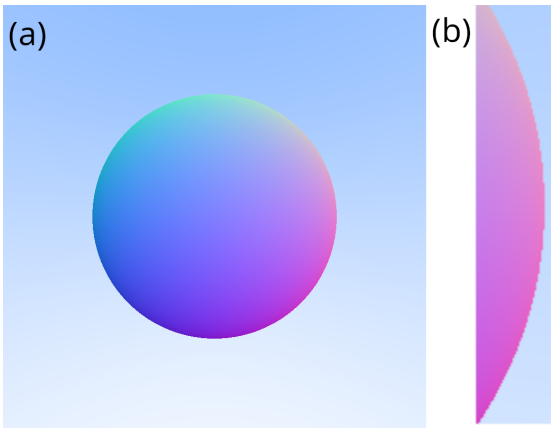


Figure 1.1: Image produced by the basic ray-tracer.

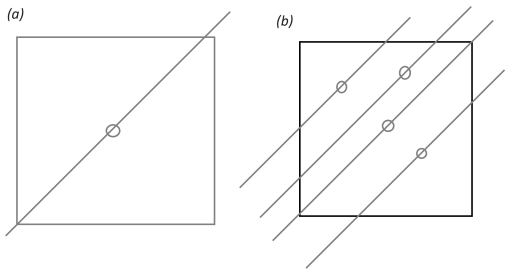


Figure 1.2: Figure showing the difference between (a) the initial ray-tracing algorithm without anti-aliasing and (b) a simple random ssAA implementation where multiple points in the pixel are sampled and averaged to get an overall colour for the pixel.

parallels between this and the common sampling methods used across scientific disciplines[56]. While the term anti-aliasing encompasses a wide variety of different techniques, the simplest form is Super-sampling Anti-Aliasing (ssAA), which can be applied here with little effort.

The basics of a ssAA scheme is outlined in Fig. (1.2). To approximate the real situation where a boundary is somewhere in the middle of a pixel, its colour can be approximated by oversampling this region using a Monte-Carlo scheme. Such an unbiased sampling scheme requires a few new additions to the code. Firstly random numbers can be generated using the C++ standard library with `std::random`, which can be scaled into the range $[0, 1)$.
Next the pixel location code can be moved into a new camera method `camera::get_ray(i, j)` which generates random locations on the view-port plane in the square bounded from $(i * \Delta u, j * \Delta v) \rightarrow ((i + 1) * \Delta u, (j + 1) * \Delta v)$.

```
inline float random_float() {
  return std::rand() / (RAND_MAX + 1.0);
}
class camera {
public:
 samples = 100;
...
  void render(FILE* output_file, std::vector<sphere> topology) {
    initialize();
    for (int j = 0; j < image_height; j++) {
      for (int i = 0; i < image_width; i++) {
        vec3 pixel_colour = vec3(0, 0, 0);
        for (int s = 0; s < samples; s++) {
          ray r = get_ray(i, j);
          pixel_colour += ray_colour(r, topology, 0) * sample_scale;
        }
        ...
      }
    }
  }
private:
...
  ray get_ray(int i, int j) {
    vec3 sample = sample_pixel();
    vec3 pixel_loc = pixel00_loc + (pixel_delta_u * (sample.x + i)) +
                     (pixel_delta_v * (sample.y + j));
    vec3 ray_origin = centre;
    vec3 dir = pixel_loc - ray_origin;
    return ray(ray_origin, dir);
  }

  vec3 sample_pixel() {
    return vec3(random_float() - 0.5, random_float() - 0.5, 0.);
  }
};
```

Listing 6: Random number generation and updating the camera methods to utilise 100 ssAA samples for each pixel.

Then the `camera::render` function can loop and sum the sampled rays to generate the final pixel colour, which must then be normalised as shown in Listing 6.

Fig. (1.3) shows the differences between two images with different numbers of samples taken. Clearly the increased sampling rate smooths out the edges and variations in the colour of the sphere, giving a smoothness to the render characteristic of ray-traced images.

The final addition to our basic ray-tracer is to simulate light rays bouncing between different members of the topology. This allows for the appearance of

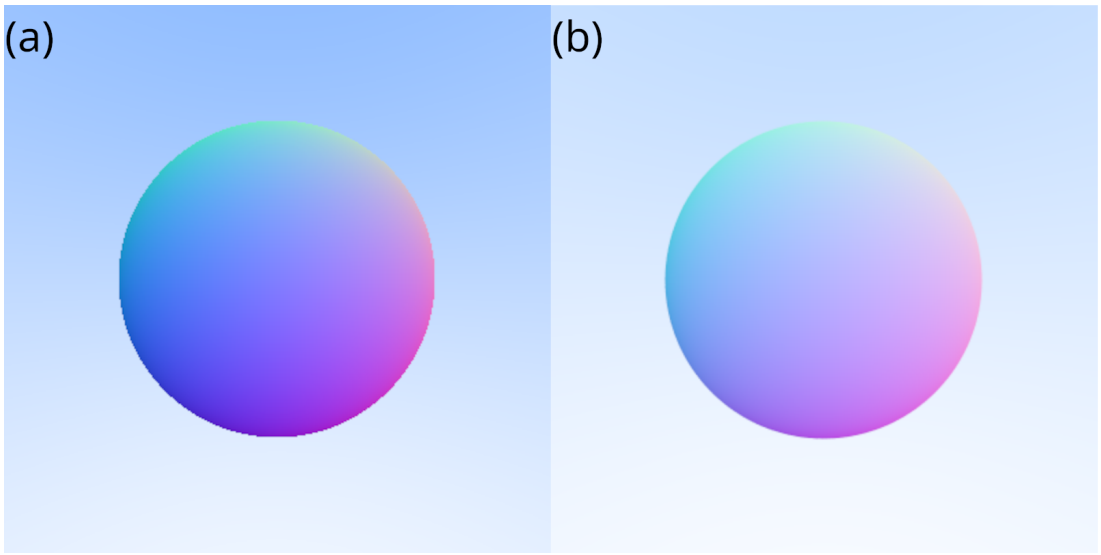


Figure 1.3: The output image from the code both (a) without any anti-aliasing and (b) with 100 ray samples of ssAA. The difference in brightness is caused by the additional sampling, which requires a more robust normalisation scheme not discussed here.

```
inline float random_float() {
  return std::rand() / (RAND_MAX + 1.0);
}
ray reflect(ray r1, hit *intersection) {
  vec3 normal = intersection->normal;
  float t = intersection->t;
  vec3 ref_A = r1.at(t);
  vec3 ref_d = r1.d + normal * (r1.d.dot(normal)) * (-2);
  return ray(ref_A, ref_d);
}
```

Listing 7: Ray reflection implementation.

basic reflective materials, such as metals or glass. Doing so requires the angle of reflection d_r of an incident light ray upon a surface, which can be calculated using the vector form of Snell's law[57]:

$$d_r = d - 2\hat{n}(\hat{n}\dot{d}) \tag{1.1.1.10}$$

Where as before d is the direction vector of the incident light ray. In the code itself, this takes the form specified in Listing 7 as a `ray::reflect` function taking in an incident ray and a record of an intersection, and returning the reflected ray. A simple form of reflection can then be implemented by introducing a boolean attribute to each sphere, specifying whether or not it is reflective or not as in

```
class sphere {
public:
  vec3 centre;
  float radius;
  bool reflective;
  sphere(vec3 c, float r, bool is_reflective = false) {
    centre = c;
    radius = r;
    reflective = is_reflective;
  };
    ...
};
```

Listing 8: Updated sphere class enabling reflections.



Figure 1.4: Image produced by the final code, featuring both ssAA and a basic reflection implementation.

Listing 8. Finally to showcase the new addition the main code is changed to generate 2 spheres, one reflective and one standard, and the camera class is updated to take a maximum number of bounces, or depth. Additionally the `camera::ray_colour` method is changed to recursively call itself on reflected rays up to the specified depth. Both these changes are shown in Listing 9. The image generated by this final code is shown in Fig. (1.4).

From start to finish, this is how early ray-tracing renderers can physically simulate the basic properties of light in order to generate an image of a simple topology comprised of spheres. Understanding the computational issues posed by problems such as this sets the basis for the following discussions of the design and

```
class camera {
public:
  ...
  int max_bounces;

  camera(int width, int height, int ns, int mb) {
    ...
    max_bounces = mb;
  }

private:
  ...
  vec3 ray_colour(const ray &r, std::vector<sphere> topology, int depth) {
    hit *hit_r = new hit;
    for (sphere s : topology) {
      if (s.hit_sphere(r, hit_r)) {
        if (depth >= max_bounces) {
          return vec3(0, 0, 0);    // Add no more colour if above limit
        } else if (s.reflective) { // reflect ray with 50% intensity
          ray reflected_ray = reflect(r, hit_r);
          vec3 reflected_col = ray_colour(reflected_ray, topology, depth + 1);
          return reflected_col * 0.5 + (hit_r->normal + vec3(1, 1, 1)) * 0.5;
        } else {    // Colour by normal
          return (hit_r->normal + vec3(1, 1, 1)) * 0.5;
        }
      }
    }
    vec3 unit_direction = r.d; // Check background col is consistent
    float a = 0.5 * (unit_direction.y + 1.0);
    vec3 result = vec3(1.0, 1.0, 1.0) * (1.0 - a) + vec3(0.5, 0.7, 1.0) * a;
    return result;
  }
};

std::vector<sphere> generate_topology() {
  std::vector<sphere> topology;
  topology.push_back(sphere(vec3(-3, 0, -5), 3));
  topology.push_back(sphere(vec3(3, 0, -5), 3, true));
  return topology;
}
```

Listing 9: Updated camera and main to enable reflections.

further uses of GPUs. Problems such as these are the exact kind of issues they are designed to solve, and modern science presents much more issues of this kind than one might think. The most important quantity here is simply the number of function calls to `camera::ray_colour` required, and this type of function is commonly referred to in computer graphics literature as a **shader**. Shaders are functions or subroutines of a renderer which display a highly parallel nature and require a substantial number of calls, often becoming the bottleneck of an

algorithm. Taking a standard modern high-definition desktop monitor resolution of 1920×1080, the most basic renderer considered here requires a monumental $2,073,600$ calls for an image unsuitable for most modern uses due to it's jagged edges and inability to simulate anything but basic materials.

Adding in the simplest form of ssAA and using 16 samples, this increases to $33,177,600$ function calls alongside even more significant overhead for sampling, summation, and random number generation. Simulating reflective materials on top of this, using a maximum depth of only 50 reflections then brings the maximum number of possible calls required to an incredible $1,658,880,000$.

It quickly became clear to developers at this time that continuing to improve the running times and expressiveness of these algorithms would require a new and creative approach.

1.1.2 The Architecture of the GPU

While the discussion so far has considered only static image producing algorithms, today even many modern video-games utilise ray-tracing models enabled in real time for accurate shading and reflection[58–61]. Achieving such speeds has only been enabled by the development of graphics specific hardware, which is now being repurposed to enable efficient scaling across modern science. This section turns to introduce GPUs by first identifying the key issues from the previous that they were designed to alleviate, studying how this inspired the development of a new computing paradigm. Modern GPUs can usually seem obtuse when considering their modern applications, however the reasons for this quickly become apparent when considering them in this context.

While the number of shader calls quickly becomes astonishingly large drifting into scales of 10^9, it is important to note some key qualities of such problems. The two main elements of any graphics pipeline are the topology and the shaders. The main scaling bottleneck can be identified as the shaders, as these key functions determine the colour of each pixel, and must be called for every sample in the image. The geometry can nearly always be considered as static, as it is at a minimum varying at a much slower rate than the propagation of light within a scene. These functions are called only once the topology is generated, and critically all use the same topology as a parameter and the same code to run, only differing in the origin and direction of the initial rays generated. These high-throughput floating point calculations are independent at run-time and subsequently embarrassingly parallel, and this is the key facet which inspired the next generation of computing hardware.

The simplest approach to speed this up is to use parallel programming to split shader calls across individual threads, requiring no additional hardware. This was the earliest approach, but it was quickly realised the CPU cannot simply be repurposed for such tasks as the scale of parallelism required is simply impossible on even cutting-edge processors. Even though they are capable of elementary SIMD operations, these are often limited to 8 or 16 elements at a time. The number of cores on a CPU usually ranges around $4 - 32$ on most machines, and the theoretical benefit of simply separating calls between threads then turns into

a simple division calculation. The realised benefit however can be much smaller due to the need for threads to be synchronised and available for OS work, via a process known as context switching handled by the scheduler[62]. With this in mind, it is useful to jump ahead to the comparison of speed between CPU and GPU implementations of rendering the simple two sphere scene considered in the previous section. Fig. (1.5) outlines these results, providing a comparison in the time taken to produce an image for each of the rendering algorithms using the two sphere scene topology.
In this example, utilising a GPU results in an incredible 285× speed increase. In practical examples this is often significantly greater given the increased complexity of scenes and shaders used, and here still provides an order of magnitude greater than what a CPU-bound parallelism scheme could provide.

This provides a good basis for a high level analysis of the design of GPU architectures, as it outlines some of the key aspects desired in dedicated hardware. Each ray thread will run the exact same code, but with different initial data allocations. For example using the nomenclature from the previous section, thread k forming the pixel at point $(i*\Delta u, j*\Delta v)$ in image space will simply be initialised with the ray from the origin to this point, meaning only loop parameters `i` and `j` in the `camera::render` method will differ between threads. The shader code, determining how this ray propagates and interacts with the geometry, is the exact same across all threads in the program, and this only needs to be stored once for all threads in a quickly accessible location. The loop determining `i` and `j` can subsequently be unwrapped, and given to the individual thread as a parameter. These concerns lead to a design of a system capable of sustaining as many threads as possible, each with dedicated caches capable of storing necessary machine code and parameters.

Each of these threads will then operate upon the same data structure, i.e. a topology, to form the final result. One of the key related hardware issues is the inefficiency of inter-process communication. A common concern when designing efficient multiprocessing code is that sharing data via parameter calls requires several copies of the data to be generated, wasting resources. The alternative approach using shared system memory requires complex access management and detailed memory addressing especially on distributed systems, which can lead

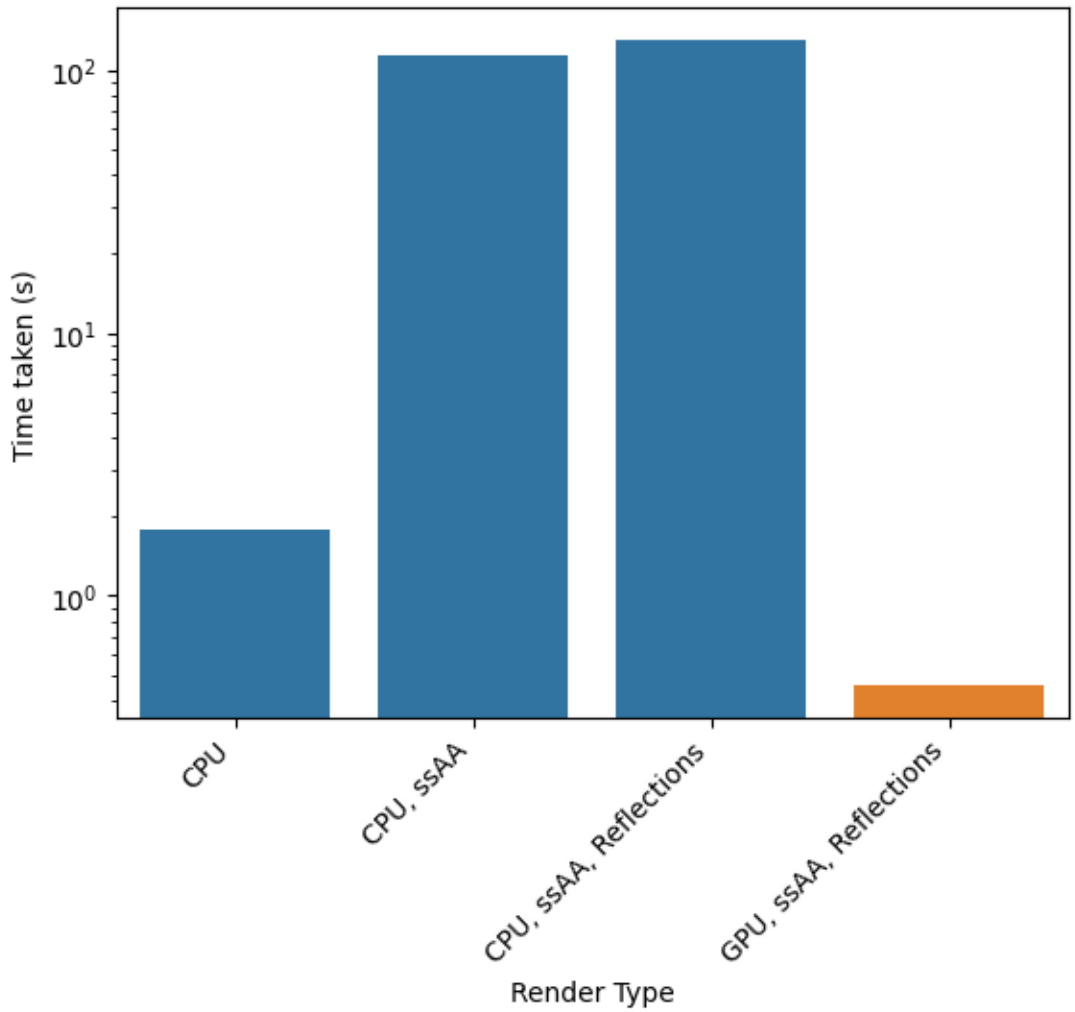


Figure 1.5: A speed comparison of the different renderers considered on the two sphere scene. All runs were performed on the same machine with a AMD Ryzen Threadripper 1950X 16-Core CPU and a NVIDIA GeForce RTX 4090 GPU. All runs with ssAA consistently used 100 samples and reflections were calculated up to a maximum depth of 50. Run times were averaged over 10 repeats each.

to access deadlocks[63]. Therefore it is often best practice to avoid data-sharing where possible by minimising the number of transfers. In an ideal scenario, the necessary data for every process would only be stored once in a single location which should allow for multiple access from all threads. Additionally, as only the final result from each thread is required and all other intermediate elements of data are obsolete when this is generated, it is possible to minimise communication to solely the initial transmission of code and data and final reception of the results. Due to the size of the final results, often requiring billions of floats-per-second for

uses such as real-time ray-tracing, this memory should be large and focused on high-bandwidth data transfer.

All of these issues point towards a solution involving dedicated SIMD hardware consisting of several processors, quick access caches, and dedicated high-bandwidth memory. The more this solution is considered, the more the benefits of having a heterogeneous system become even clearer. This allows for a somewhat asynchronous execution, as the CPU can handle ensuring the data in memory is correct and retrieving the results, while the external device focuses on maximising the calculation throughput. As all threads on the main CPU are open to perform general OS work assigned by the CPU scheduler, they have complex architectures necessary to perform context-switching from task to task and handle branching control paths efficiently. Context switching alone is an incredibly power inefficient processes which puts severe limitations on CPU cores performing parallel tasks as they constantly require to communicate with the rest of the system and designate cycles to performing OS work. By separating the hardware not only can this be completely avoided, which is ultimately a much more effective solution than rewriting the scheduler to avoid certain cores, but it also means the processors themselves can have a much simpler design. Processes on the dedicated hardware can be incredibly light-weight with significantly smaller overhead when compared to their CPU equivalents, meaning far fewer processors are required for a significant speed increase. Removing complexities such as branch prediction and frequent scheduler induced caching means smaller local memory is required for each processor, and it can focus on committing every cycle to running the desired numerical code. Additionally due to the same code being reused, each processor can be responsible for several threads at once to maximise parallelism.

Now that some of the motivations for GPU design is clear, the common nomenclature of implementations of the hardware can be introduced using NVIDIA's CUDA architecture as an example. In the nomenclature used in this work, the GPU is often referred to as the device, and the CPU machine as the host. Contrary to how a CPU core is commonly thought of as a single cohesive unit, in practice GPUs make a clear distinction between cores and processors and utilise them as two distinct entities[64]. This is due to their inherent SIMD nature, allowing hardware designers to utilise a simple control flow engine capable of distributing

basic instruction sets to many separate cores which each perform grouped computations. The previous terminology of a **shader**, the type of embarrassingly parallel program benefitting from SIMD which inspired the GPU, becomes known by the more generic **kernel**. A **thread** refers to a single run of a kernel's instruction set operating on a single data point or parameter set stored in the dedicated RAM or VRAM, or an onboard cache. The processor, which manages the distribution of the instruction set alongside the current heap and stack of a collection of threads, is implemented as a **Streaming Multiprocessor (SM)**. A **core**, **CUDA core** (NVIDIA), or **compute core** (AMD), refers to the specific unit of the device which performs the computation- and there are many specific versions for speeding up various different types of computation. NVIDIA devices can include CUDA cores for general computation, ray-tracing cores for the uses discussed, and tensor cores for high-dimensional matrix multiplication. Each core can only run threads in multiples of 32, called a **warp**, and the core must wait for each thread to finish the last instruction before continuing with the program. Each SM has a collection of cores varying between GPU models, but commonly between 32-128. Threads are organised into collections known as **blocks** which are guaranteed to run on a single SM allowing for intra-block thread communication, and are limited to a maximum of 1024 threads.

1.1.3 Coding on GPUs

Low-Level

Now that the general construction of a GPU is clearer, lets view how the prior ray-tracing example can achieve a 285× speed increase by utilising some basic functionality of the hardware. The main difference is to use CUDA enabled C++ and the NVIDIA CUDA compiler[65]. All functions but the main body must be prefixed with a device identifier specifying the device they are intended to run on and their scope:

- `__device__` : Exists on the GPU only, can only be called from other GPU-bound functions and runs on the GPU.

- `__host__` : Exists on the CPU only, can only be called on CPU-code and runs on the CPU.

- `__global__` : Runs on the GPU, but can be called on code by either device.

While this may seem complex, such differentiation means `__global__` functions can serve as entry-points from host-bound code into pure-GPU sections of device-based functions. With this setup, interaction between devices is explicit and each device can function as an independent system requiring no further communication. In this example, as it is relatively small, functions can be offloaded onto the device apart from the main loop, causing all computation but file output to be GPU-accelerated. Anything called within the main loop must be specified as global, and anything not can be a pure device function.

Each time a GPU function is called, the syntax `kernel<<<GRID, THREADS>>>` is used to specify a grid of blocks and the number of threads per block. The grid is purely an aid for programmers to organise blocks and the containing threads by assigning indices. Both arguments within the angled brackets can be 1D, 2D, or 3D vectors using CUDAs inbuilt `dim3` vector class, as long as the number of elements in each block is below the total limit of 1024. This allows the unwrapping of loops to be performed easily, by either accessing VRAM data using thread-specific variables or using the indices to perform calculations such as locating the pixel in this case. CUDA allows each thread to access unique values for block-assigned positions `threadIdx.x/y/z` and `blockIdx.x/y/z`, and kernel-wide block size variables `blockDim.x/y/z`. In this case, using 2D arguments allows each thread to match up to a unique pixel. As threads are run in warps, it is often important to use indices to also place guards preventing extra threads with invalid indices from being run and resulting in memory access errors.

Finally, all objects in memory required for the kernel must be moved to device memory. Simplest way to achieve this is via the `cudaMalloc(**devicePointer, size)` function, passing both a pointer which will be assigned towards a device memory location of the size specified by the second argument. As kernels cannot return values to the CPU stack, these allocated device pointers must be then used in global or device functions. The topology is required to move to the VRAM, as these will be accessed by all threads, and a results array is now necessary as

colours will need to be moved to the host once calculated. Additionally, when these arrays are generated as only one copy is needed it should be executed on a single thread.

Firstly the topology generating function and results array can be generated. This requires a move from using a `std::vector` for the spheres to a static memory array with a fixed size. It also requires generating two copies of the results array on both the host and device, and transferring the data over once the calculation is finished. Each of these generating functions can be run on the device with a single thread, alongside an index-catching if clause to prevent additional executions. Also the now global `cam.render` function instead takes the device pixel array and the device topology. It is called with a number of threads equal to the number of pixels, separated into 625 blocks allocated into a square grid, each composed of a further 32×32 square grid of threads. The initialisation of the camera variables are additionally made public and called outside the render function so that they can only be performed once. The necessary changes to the code for this memory management, kernel calls, and grid set-up is shown in Listing 10.

The only other change required is to rewrite the `render` function to utilise the GPU-grid indexing to unwrap the main bottleneck of the for loop. Each thread will now calculate indices i and j from its location in the CUDA grid, and perform all samples of its pixels colour independently. The final colour will be stored in the device array, which can later be written by host code which is omitted again for brevity. Revisions for this are shown in Listing 11.

With just these simple changes an incredible speed up is achieved. Through just a few basic API calls this renderer can be made orders of magnitude faster. The issue with a significant amount of GPU programming is not the technical changes required, but the conceptual difficulties in understanding what the code is doing and how it can be accelerated most effectively. The aim of this prelude was to show that considering the initial environment GPUs were made for, many of these design choices make perfect sense and result in a huge speed-up. Contrasting the view achieved when solely considering modern science applications, where their functionality can sometimes seem unintuitive and obtuse.

For clarity however, it is important to note that in practice there would be a lot of boiler plate code around these device function calls in order to query for

```
__global__ void generate_topology(deviceMem) {
    if (threadIdx.x == 0 && blockIdx.x == 0) {
        deviceMem[0] = new sphere(vec3(-3, 0, -5), 3);
        deviceMem[1] = new sphere(vec3(3, 0, -5), 3, true);
}


int main() {
    // Create device topology
    sphere **deviceTopo;
    cudaMalloc((void **)&deviceTopo, 2 * sizeof(sphere *));
    generate_topology<<<1, 1>>>(deviceTopo);
    ...
    // Create results array on both host and device
    vec3 *pixels_device;
    int num_pixels = width * height;
    cudaMalloc((void **)&pixels_device, num_pixels);

    vec3 *pixels_host = new vec3[num_pixels];
    ...
    dim3 grid_size(25, 25);
    dim3 block_size(32, 32);
    cam.initialise<<<1, 1>>>();
    cam.render<<<grid_size, block_size>>>(pixels_device, deviceTopo)
    cudaMemcpy(pixels_host, pixels_device,
               num_pixels*sizeof(vec3),
               cudaMemcpyDeviceToHost);

    ... // Write the image using CPU bound code
}
```

Listing 10: Generating the topology and results array on device memory, alongside moving it back once finished.

error codes. If the device throws an exception it will not stop the host unless explicitly told to do so and to check if the call failed, and manual error checks must be frequent. It is also important to note that in this example it is fortunate that there is absolutely no interaction between threads, but this cannot always be avoided. If instead the code was further parallelised to place each `ray_colour` call into individual threads, memory access for the `pixel_colour` variable would then become problematic. Due to the asynchronous nature of GPU scheduling, there is no guarantee on threads having exclusive access to the variable. This means it is possible for one thread to access it, change the value, and rewrite it, all while another thread does the same with a few milliseconds difference. A situation like this is known as a race condition, and generates an incorrect and unpredictable value for the accessed variable. There are several ways to avoid this.

```
class camera {
public:
    __global__ initialise() {
        if (threadIdx.x == 0 && blockIdx.x == 0) {
            ...
        }
    }
...
    __global__ void render(vec3 *result, sphere **topology) {
        int i = threadIdx.x + blockIdx.x * blockDim.x;
        int j = threadIdx.y + blockIdx.y * blockDim.y;
        if ((i >= image_width) || (j >= image_height)){
            return
        }
        int pixel_index = j * image_width + i;
        vec3 pixel_colour = vec3(0, 0, 0);
        for (int s = 0; s < samples; s++) {
            ray r = get_ray(i, j);
            pixel_colour += ray_colour(r, topology, 0) * sample_scale;
        }
        result[pixel_index] = pixel_colour
        }
    ...
};
```

Listing 11: Rewriting the renderer to work in parallel.

Atomic operations provide basic arithmetic calls via API calls which block access while a variable is in use, or the code can be rewritten for example to store values to vector which is then summed. While this will not be explored further here, it is probably the most common issue plaguing GPU programming.

High-Level

In practice, many modern developers can use much of the benefit of modern GPUs without delving into the hardware bindings. This is due to the wide applicability of GPU code, focusing simply on the numerical heavy lifting applicable to a broad range of issues. Python has become the de-facto language across much of science, especially in ML and data-science. While it may seem initially surprising that in such performance focused environments a relatively slow language like Python has become the de facto language of choice, there is more to it. An often overlooked benefit of the language is it's ease of extensibility, thanks to the interpreter and underlying C++ libraries. This has led to a furthering of the splitting of research developers between researchers and software engineers, with the latter focusing

on performant and general code. While both write software, it is common for the lower level C++ and CUDA bindings and implementations to be handled by the engineers, while the researchers can continue to code quickly and simply.

The interpreter enables Python code to quickly call performant C++ code, rather than much slower Python code, when specific criteria are checked at run-time. This can often be GPU accelerated for even greater speed increases. Popular libraries like NumPy and SciPy provide high-level mathematical functions which call performant pre-compiled code under the hood[66,67]. Libraries like CuPy and numba provide a very similar interface to GPU accelerated functions, giving the Pythonic ease of these libraries with significant speed increases[68,69]. Enabling all researchers familiar with any of these APIs to interchangeably and easily write GPU accelerated code. Also both GPU-based examples provide an API similar to CUDA with the ease of Python through either using a simple JIT compilation decorator or directly writing kernel code.

One of the greatest benefits is increased speed of large matrix multiplication, and parallels to the necessary operations can be drawn from ray-tracing example. GPUs are capable of splitting the products of massive matrices into grids, and combining them in perhaps the most efficient way with the only limiting factor being the VRAM and maximum concurrent threads[70]. In large data processing centres, state-of-the-art servers regularly link multiple GPUs together in parallel and use this to process the almost limitless data produced by modern consumers. This can be seen as the enabling factor in the wide-spread of ML techniques, which rarely go beyond utilising element-wise or matrix operations. The development of Python tensor libraries have accelerated this spread, and provide one of the easiest ways to interface with the modern GPU requiring no more than a single call to move data between devices. This is used to make dramatic speed-ups in linear algebra tasks, and the development of modern ML tooling has used this speed to make complex non-linear modelling commonplace through a few simple enhancements. This is the next focus of this work, studying how the technology introduced here to solve a very different problem has had a profound impact on the state of modern science.

1.2 Machine Learning

This section looks to formally introduce the field of modern ML. While the basic theory of ML never strays far from elementary calculus first developed long before modern GPUs were invented, it's implementation and usage has risen dramatically over the past decade[71–75]. This increase in popularity is intrinsically linked to the recent progress in computational speed on GPUs and their widespread adoption[76–81]. These advances have allowed for many forgotten theories to flourish, as they are found to require a previously unobtainable numerical scale to work effectively on difficult problems in modern science. This scaling, along with their capability to perform expressive non-linear modelling, have played fundamental roles in the prevalence of ML theory today. It has become a necessary part of any modern scientists toolkit, and it is hard to find an area not yet impacted by these tools. This section begins with a brief overview of the mathematics underpinning ML, introducing back-propagation as an example of a wider class of problems solved by automatic differentiation. This touches upon the theory of why ML is so successful at dealing with non-linear high dimensional problems. The focus then turns to studying some of the common model architectures used today, looking briefly at how the modality of data impacts which models often work the best. Finally finishing with a look into some of the practical considerations required to implement modern ML strategies.

1.2.1 Automatic gradients on computational graphs

Backpropagation is often touted as the major breakthrough which allowed the current rapid development of AI, however the theory itself is far from modern[82,83]. The first references to similar error propagation theories began in the 1960's, with a full implementation comparable to the modern versions coming almost a decade later[84,85]. At it's core, it is a practical implementation of basic calculus and as such its history goes back much further to the forefathers of the subject in the 1600's who first developed the chain rule for differentiating composite functions[86]. The most ground-breaking development leading to the current data-driven revolution is instead the efficient implementations of automatic differentiation, or autodiff for short. This system allows for the numerical calculation of gradients for arbitrary

functions, as long as all operations admit a closed form expression of their gradient[87]. As a tool it provides the back-end to nearly all modern techniques, and in recent decades its speed and scale have increased dramatically. This has allowed for large models such as Large Language Models (LLMs), protein structure prediction networks, and image generation algorithms to comprise of billions of parameters while still performing inference tasks quickly. Despite also evaluating gradients, autodiff differs greatly from related differentiation tools such as symbolic or finite-difference methods, focusing instead on computational speed and feasibility over generality.

Formally in this work, we use the term ML interchangeably with neural network, to refer to statistical modelling techniques where models are optimised, or trained, using autodiff based gradient-descent methods. While this covers a large number of modern methods, including supervised, deep, and reinforcement learning, it excludes many historical techniques, such as correlation matrix based methods and classical regression, which are commonly considered ML techniques. It shall be seen that autodiff is prevalent across modern science, leading to both its success and inherent limitations on the kind of models which can be considered. This provides the key unifying thread between disparate areas of modern research and many of the methods used within this work. As shall be seen, when considering modern ML from the perspective of autodiff, backpropagation becomes a relatively simple application.

Succinctly, modern ML looks to build models from data as efficiently as possible. By constructing general classes of solutions to generic numerical problems, it is possible to limit the amount of domain expert input required, such as establishing priors or detailed relationships. This can be viewed as the general aim of modern ML, and is a direct response to the size and complexity of many modern datasets. Achieving this aim means looking to push modelling capabilities to cover non-linear relationships, where closed-form expressions of optimization criteria are nearly always unobtainable. To first build any such model, the problem needs to be clearly outlined.

The initial step of model building is to obtain a dataset $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}$, formed of a collection of N pairs of observations, or dependent variables, $y \in \mathcal{Y}$, and data-points, or independent variables, $x \in \mathcal{X}$.

The data-space $\mathcal{X}$ is a computational representation of the measurable quantities associated with the system considered, and can cover any numerical space capable of being stored in a machine-readable format[88,89].
The observations can take many forms such as a space of categorical identifiers, an associated quantity, or a related statistical distribution for example. Including, as is often the case in deep learning applications, equivalence to x.
While the field inherits the terminology of dependent and independent variables from statistics, in practice they are often used quite loosely in ML.

A model may then be considered as a function $\Phi(\mathbf{x};\theta) = \hat{\mathbf{y}}$, where $\theta = \{\theta_i\}_{i=1}^{\Theta}$ are the Θ variables which parametrise the model. Its general aim is to best approximate the underlying relationship between $\mathbf{x}$ and $\mathbf{y}$, by predicting a value $\hat{\mathbf{y}} \in \hat{\mathcal{Y}} \cong \mathcal{Y}$ from a data point $\mathbf{x} \in \mathcal{X}$. Comparison then between y and $\hat{\mathbf{y}}$ allows for the quantification of a models accuracy, using a loss function $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$. This importantly yields an effective mechanism to rank different predictors.
While this set-up may seem abstract, it allows for a general scheme encompassing most familiar and rudimentary models. For example, consider the simplest linear least squares model predicting a straight line relationship between $(x, y) \in (\mathcal{R}^1, \mathcal{R}^1)$, taking the form:

$$\Phi(x, \theta) = \theta_0 x + \theta_1 = \hat{y} \tag{1.2.1.1}$$

The loss function then takes the form of the sum of squared residuals:

$$\mathcal{L}(y, \hat{y}) = \sum_{i=0}^{N} (y_i - \hat{y}_i)^2 \tag{1.2.1.2}$$

Clearly the smaller this value, the better the model performance.

While this trivial example does not provide much useful information, the general method to solve such a linear least squares model is surprisingly the basis for current techniques. As Eqn. (1.2.1.2) is convex, its minimum admits a solvable closed-form expression and is located where the partial derivative with respect to each parameter is zero. While in general the complex forms $\mathcal{L}$ will take do not admit such an expression, it is still often possible to calculate a derivative for

them. Therefore an intuitive way to optimise arbitrary models such as Φ is to use the gradient of $\mathcal{L}$ with respect to θ, yielding the direction in parameter space which maximally decreases the error.
When a closed form solution is unable to be found, using this in an iterative fashion to find a potential change in θ forms the basic theory of gradient descent:

$$\theta_{t+1} = \theta_t + \Delta\theta_t = \theta_t + \alpha\frac{dL}{d\theta_t} \tag{1.2.1.3}$$

Where the total derivative d represents the sum of all partial derivatives with respect to θ, α is a constant or function representing the magnitude of the change, and t indexes successive steps of the algorithm.
Note the abuse of notation here, as although the terminology of a total derivative is being used Φ is also dependent on $\mathbf{x}$ which is not considered.
Evaluation of the gradient on the right is the fundamental quantity key to gradient descent methodologies. Here, it provides the key quantity autodiff aims to calculate and in the most general case ensuring it exists requires several assumptions.

For the laws of calculus to be well defined, the minimal constraint placed on $\mathcal{X}$ and $\mathcal{Y}$ is that they are smooth manifolds. Here, for brevity, the details of how this is formally defined will not be covered and interested readers are pointed towards the many textbooks on the subject[90,91]. In general, the consequence of this is that every point in these spaces has a local open subset, or ball, within which the space is isomorphic to Euclidean space. The space itself is covered by the union of these balls, allowing for smooth local translation between points and ensuring limits can be taken such that derivatives are well-formed. This places no constraints on global structure, encompassing more complex non-linear geometries.
Additionally for a derivative of a model to be well-formed, the functions $\mathcal{L}$ and $\Phi(\theta)$ must also be differentiable themselves. This critical constraint places significant restraints on the number of functional forms a model can take. If these conditions are met, any model can be effectively optimised using a derivative-based strategy. These minimal constraints lead to different interpretations of ML theory attempting to explain the vast computational power and wide-applicability of these methods. From simple statistical modelling in $\mathcal{R}$, to more complex interpretations

where fitting is seen as a topological process approximating an underlying data manifold[92,93].

Practically implementing a gradient-based training scheme for this general modelling task requires several theories from multi-variate calculus. Usefully the existence of the local isomorphism to Euclidean space on a smooth manifold means all calculus derived in Euclidean space is applicable to them in local patches.
To begin, some basic multivariate calculus shall be covered as a generalised strategy for evaluating the gradients of composite functions is built.
The simplest real-valued function $f : \mathcal{R} \to \mathcal{R}$, is said to be locally differentiable at point $p \in \mathcal{R}$ if:

$$\exists \;\; f'(p) := \lim_{\delta \to 0} \frac{f(p+\delta) - f(p)}{\delta}$$

The derivative term may be encompassed by the fraction to lead to an equivalent formulation:

$$\lim_{\delta \to 0} \frac{f(p+\delta) - f(p) - f'(p)\delta}{\delta} = \lim_{\delta \to 0} \frac{f(p+\delta) - f(p) - \lambda(\delta)}{\delta} = 0 \qquad (1.2.1.4)$$

Where the linear transform $\lambda(\delta) = f'(p)\delta$ has been introduced.
The existence of $\lambda : \mathcal{R} \to \mathcal{R}$ is therefore an equivalent condition as to whether f is differentiable at p. This formalises the notion of the local manifold translation which maintains differentiability.
Usefully, the form Eq. (1.2.1.4) allows for a clear generalisation to arbitrary dimensional mappings $f : \mathcal{R}^n \to \mathcal{R}^m$ if the scalar δ is promoted to a vector $\boldsymbol{\delta} \in R^n$, originating from p:

$$\lim_{\boldsymbol{\delta} \to 0} \frac{|f(p+\boldsymbol{\delta}) - f(p) - \lambda(\boldsymbol{\delta})|}{|\boldsymbol{\delta}|} = 0 \qquad (1.2.1.5)$$

It can be shown that the function λ is unique, leading to its common name as the directional derivative of f relative to $\boldsymbol{\delta}$[94].
Given any vector at p, this quantifies of how it changes under f as it is mapped to $\mathcal{R}^m$. There is a clear freedom here with the direction of $\boldsymbol{\delta}$, allowing us to take the directional derivative relative to any vector. Taking the directional derivative for each of the natural basis of unit vectors $\mathbf{e} = \{e_1, ..., e_n\} \in \mathcal{R}^n$, yields a measure of how the total structure of the vector space changes under the transform f. This

allows for identification of $\sum_i \lambda(e_i)$ with the total derivative of f at p, $D_p f$, or simply df assuming f is differentiable at all p:

$$df = \sum_i \sum_j \frac{\partial f_i}{\partial e_j} \tag{1.2.1.6}$$

Where f has been decomposed into components along each resulting dimension $f = \{f_1, ..., f_m\}$.
This quantity expresses the closest estimate of the local gradient of f across all directions at p. Importantly, this can be used to find the direction of steepest descent vital to the optimisation problem.

While the total derivative is the fundamental quantity of interest, when optimising functions it is necessary to have an operator form. This not only allows for quick evaluation of the total derivative, but also gives an efficient way to study how an arbitrary vectors v are transformed under f while avoiding having the perform specific directional derivatives each time. This total derivative operator can be found using the $m \times n$ Jacobian matrix $\mathcal{J}(f)$, defined equivalently as:

$$\mathcal{J}(f) = \frac{df}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \cdots \frac{\partial f}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \tag{1.2.1.7}$$

This gives a practical and explicit matrix-operator formulation, able to calculate both the total derivative of a differentiable function and the effective action of the function on an arbitrary vector using a linear map.

It is important to note that this derivation has not been explicit about tangent vector spaces. So far has only dealt with Euclidean space where the tangent space $T_p\mathcal{R}^n$ is isomorphic to the domain space $\mathcal{R}^n$, and therefore such details are superfluous. However it is important to note that by generalising the derivative to Eqn. (1.2.1.4), the promotion of δ to a vector means it becomes, along with all other from around p, a part of the tangent space at p, T_p. In smooth manifolds, as there are fewer constraints on global structure, there is no guarantee on how $v \in T_p\mathcal{R}^n$ transforms as $p \to q \in \mathcal{R}^n$ or under a potentially non-linear transform induced by f. This means that when considering $\mathcal{J}(f)$ in differential geometry

terms, it is a mapping from the coordinate vector $\mathbf{e} \in T_p\mathcal{R}^n$ to the image tangent space $T_{f(p)}\mathcal{R}^m$. This transfers the canonical coordinate basis from the domain tangent space to the image, inducing a similar differential structure there. This provides the vital link between the tangent spaces of the image and the domain, allowing for an explicit calculation of the change in vectors under any non-linear differentiable transform.

These technicalities are of upmost importance to the theory of autodiff, and can be seen as the critical reasons why it is able to model such complex relationships with relative ease. In this context $\mathcal{J}(f)$ is technically the push-forward of f, a class of functions which play a key role in physics applications of differential geometry such as general relativity. In physics, the concern is often how vector-valued quantities such as velocity and length transform along a spacetime manifold under a Lorentz group transformation, giving rise to many of the perplexing qualities of relativity. For our usage however, it is enough to know the push-forward in general is defined for any smooth function between any two manifolds $\mathcal{X}$ and $\mathcal{Y}$, and maps a vector $v \in T_pX$ to the associated vector in the image's tangent space $w \in T_{f(p)}\mathcal{Y}$. This means taking the Jacobian-Vector Product (JVP) of the Jacobian and an arbitrary vector $\mathcal{J}(f) \cdot v$, evaluates the change in any vector through f, enabling a vector to be "carried" along a transform. As with many mappings in mathematics, the push-forward also has a dual categorical mapping known as the pull-back. This intuitively can be used to transform a vector $\mathbf{a} \in T_{f(q)}\mathcal{Y}$ to the related vector $\mathbf{b} \in T_q\mathcal{X}$. In practical terms, this can be equated with simple pre-composition, and fortunately in the canonical basis takes the form of the conjugate adjoint, or simply transpose in real space, of the Jacobian $\mathcal{J}(f)^\dagger$. It is therefore possible to evaluate such a function in reverse, moving a vector from image space to the domain using the Vector-Jacobian Product (VJP) $v \cdot \mathcal{J}(f)$, or equivalently $\mathcal{J}(f)^\dagger \cdot v$. The push-forward and pull-back therefore give a rigorous theoretical grounding to moving vectors between the tangent vector spaces through any arbitrary smooth mapping between two manifolds. The push-forward defines how vectors transform through a transformation, and the pull-back can be used to perform this journey in reverse[95].

While this general case of moving vectors along with transforms is relatively simple, the power of the method becomes clear when considering derivatives

as ordinary members of the tangent space. This outlined theory then allows derivatives to be propagated along any differentiable function f between two smooth manifolds. Returning to the problem of evaluating the derivative in Eqn. (1.2.1.3), this can be used to begin implementing a training scheme for generic differentiable models. The first step towards doing so is identification of f as a composite of both Φ and $\mathcal{L}$, which have already been defined as smooth functions. The second, is noting if both functions are required to be smooth, all of the above theory can also be applied to any function or operation composing them, requiring only use of the chain rule:

$$\frac{du}{dv} = \frac{du}{dt}\frac{dt}{dv} \tag{1.2.1.8}$$

For any smooth functions u, v, and t.
Applying this to the derivative for the loss splits the derivative into a form easier to evaluate:

$$\frac{dL}{d\boldsymbol{\theta}} = \frac{dL}{d\Phi}\frac{d\Phi}{d\theta} = \frac{dL}{d\hat{\mathbf{y}}}\frac{d\Phi}{d\boldsymbol{\theta}} \tag{1.2.1.9}$$

Therefore the aim is to evaluate this by obtaining two numerical gradients, one of the loss with respect to the model's predictions, and the other of the models predictions with respect to the parameters.

Considering Φ as a composite function:

$$\Phi = D \cdot C \cdot B \cdot A(\mathbf{x}) \tag{1.2.1.10}$$

extending the chain rule is trivial:

$$\frac{d\Phi}{d\boldsymbol{\theta}} = \frac{d\hat{\mathbf{y}}}{dC}\frac{dC}{dB}\frac{dB}{dA}\frac{dA}{d\boldsymbol{\theta}} \tag{1.2.1.11}$$

This gradient can be numerically evaluated by simply multiplying the gradients of the component functions together. By only considering functions with explicitly and easily calculable gradients, it is possible to perform the gradient calculation by swapping the original function for it's derivative and evaluating. This is the simplicity of autodiff. Early implementations used the intuitive approach of a dual space style calculation, where simultaneous evaluations of the function and its gradient were performed in parallel[96].

A naive way to evaluate Eqn. (1.2.1.10) is then to multiply the corresponding Jacobians $\mathcal{J}(\circ)$ of each transform together to build the $\Theta \times \dim(\hat{Y})$ composite Jacobian. However this very quickly becomes inefficient in practice, as $\mathcal{J}$ is a very sparse matrix once evaluated and the majority of these calculations are needless. Consider for example an element-wise squaring operation performed on a K-dimensional vector. The associated Jacobian to this matrix is of size $(K \times K)$ despite the only non-zero values being across the leading diagonal.

The final result does not require the whole basis evaluated at every step. A much more efficient strategy is to realise the required gradients only depend on the effect these intermediary Jacobians have on a set of linearly independent vectors in the initial domain or final image space. Instead by considering the JVPs and VJPs of each composite transform, only the necessary elements of each individual transform can be evaluated so long as the final size is correct and the derivative vector can be pulled or pushed through each transform. This provides the vital speed-up necessary to make large-scale modelling tasks computationally feasible, and applying this concept to Eqn. (1.2.1.10) arrives at two equivalent methods for evaluating the derivative, made clearer through the use of parentheses and introduction of initial gradient vectors:

$$\frac{d\Phi}{d\boldsymbol{\theta}} = \frac{d\hat{\mathbf{y}}}{dC}\left(\frac{dC}{dB}\left(\frac{dB}{dA}\left(\frac{dA}{d\boldsymbol{\theta}}\frac{d\boldsymbol{\theta}}{d\boldsymbol{\theta}}\right)\right)\right) \quad \text{Forward (JVP)} \tag{1.2.1.12}$$

$$\frac{d\Phi}{d\boldsymbol{\theta}} = \left(\left(\left(\frac{d\hat{\mathbf{y}}}{d\hat{\mathbf{y}}}\frac{d\hat{\mathbf{y}}}{dC}\right)\frac{dC}{dB}\right)\frac{dB}{dA}\right)\frac{dA}{d\boldsymbol{\theta}} \quad \text{Backward (VJP)} \tag{1.2.1.13}$$

In the first strategy, known as forward mode autodiff, the initial gradient vector is initialised, or seeded, as $\frac{d\boldsymbol{\theta}}{d\boldsymbol{\theta}} = 1$, and gradients propagate through the components of Φ using using the push-forward. This seems a sensible strategy as the order of function computation matches the order of derivative evaluation, leading to a simple parallel calculation between functions and their derivatives that can be done at compute time as each function is called.

However, this is very inefficient. From an implementation standpoint, this will require n calculations at each composite function to fully realise $\mathcal{J}$, as each time a different dimension of the domain is seeded as 1 and the rest 0. This approach

is equivalent to building $\mathcal{J}$ one column at a time.

Alternatively the VJP can be used in a methodology known as reverse mode autodiff. This instead uses $\frac{d\hat{\mathbf{y}}}{\hat{\mathbf{y}}} = 1$ as the initial seed, and propagates the gradient backwards from the final result. Resulting in only m calculations required, equivalent to building $\mathcal{J}$ one row at a time. As most loss functions are scalar quantities, ensuring $m << n$ in nearly all circumstances, reverse mode autodiff is the prevalent implementation in practice. Its importance is highlighted by the fact it has cropped up as groundbreaking research in over a dozen disparate areas[97].

Reverse mode differentiation is therefore the backbone of modern ML, and is widely used to train models like Φ throughout science and industry. Its practical implementation is significantly more complex than that of it's forward counterpart, as it may only be calculated once the final results $\hat{\mathbf{y}}$ are calculated. Tensor libraries written mostly in Python with C++ bindings- such as PyTorch, TensorFlow, and JAX, have emerged to handle the difficult back-end implementation schemes[98–100]. These libraries at first glance are very similar to standard numerical libraries like NumPy and SciPy, however they have a few basic features that enable an efficient and fast autodiff implementation which shall now be covered using PyTorch as an example.

Tensor libraries enable the fast computation of gradients via reverse mode differentiation and are widely used to train ML models. While their naming convention comes from the related object in physics, it relates only in the similar usage storing multidimensional quantities rather than properties under transforms. A tensor in a autodiff library such as PyTorch is the central data structure, and it differs from most other counterparts by its ability to store metadata about its usage. This metadata can be used to build a computational graph capable of evaluating the relevant gradients post execution.
Consider the model hypothesised in Eqn. (1.2.1.10). An equivalent way to consider the relationship between the derivatives involved is to look at the whole function as a Directed Acyclic Graph (DAG), such as in Fig. (1.6).
Tensor libraries compose such a graph as they are calculated by storing recursive references to each operation performed. In PyTorch, whether this happens is controlled by the `requires_grad` attribute of a tensor, and when this is true the

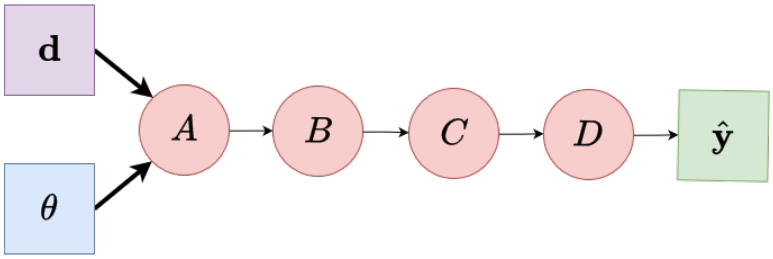


Figure 1.6: An example of a computational graph produced by a tensor library. Forward mode autodiff considers this graph from the left-hand side to the right, requiring Θ *calculations for a full derivative with respect to the parameters* $\boldsymbol{\theta}$*. Reverse mode instead traverses this graph from right-to-left, requiring instead* $\dim(\hat{\mathbf{y}})$ *calculations.*

`grad_fn` attribute of the tensor serves as a node or entry point to this DAG. Once a computational graph has been established, the next step is to actually calculate the required gradients. This is implemented in PyTorch using two class methods on the abstract `torch.autograd.Function`, named `forward` and `backward`. Intuitively the forward is the conventional implementation of a function, performing the expected operation, and backward is the associated gradient computation.
As an example, consider a basic one-parameter example model utilising a prediction of the form:

$$\hat{y} = \sin[(x + \theta_1)(3\theta_1 + 1)] \tag{1.2.1.14}$$

The gradient DAG formed by a tensor library is shown in Fig. (1.7). Each end node (square) here represents an input (bottom) or output (top) tensor, and each edge is a derivative showing its value in a white square box. Calculating the partial derivative of any tensor in the graph requires following this graph from the top to the node, and multiplying each route together while adding the different branches due to the associativity of derivatives. For example, for the parameter θ_1:

$$\frac{\partial \hat{y}}{\partial \theta_1} = \cos(35)(7 \times 1 + 5 \times 2) \approx 13.926$$

Therefore simply substituting and evaluating the derivatives associated to each composite function gives a simple way to calculate the derivative at run time.

In PyTorch, the back-end engine responsible for running this is autograd, and it generates the full computational graphs only when called using the `Tensor.backward()` method. Once this is called, autograd generates the nec-

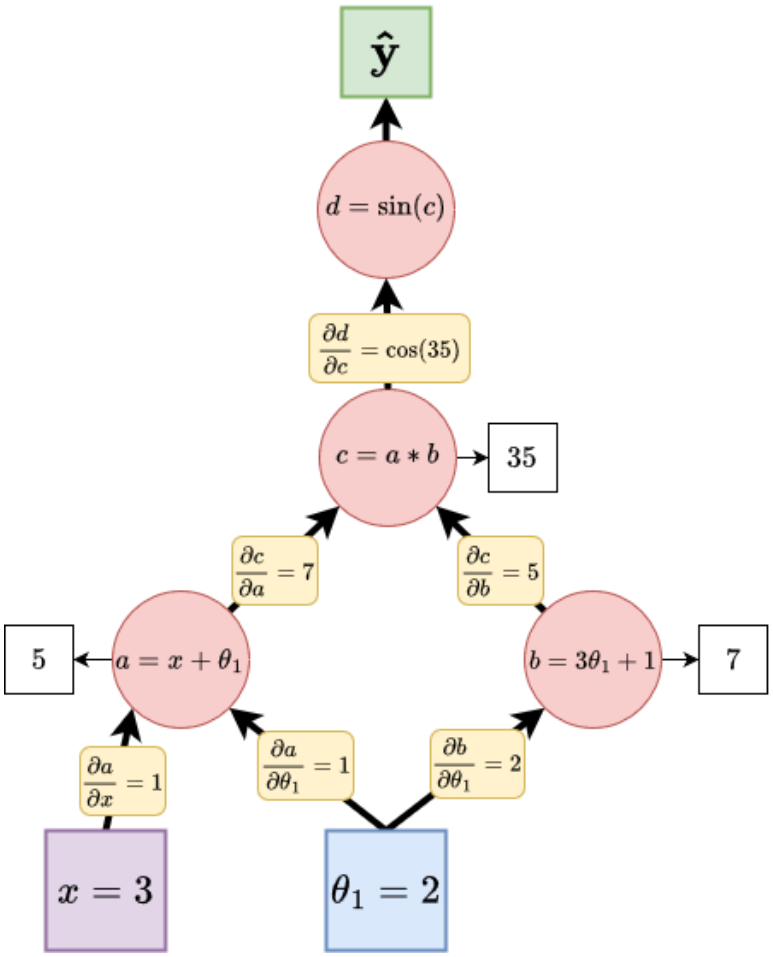


Figure 1.7: An example of a computational graph that may be generated by a tensor library.

essary graph up to the tensor invoking the call by calling specific computation kernels. This moves backwards through all associated tensors, filling their `.grad` attributes with the associated gradient evaluated at their corresponding node.

By wrapping these all up in a tensor library, Python code can be used to call GPU-accelerated C++ code, and again great computational speed can be reached over a wide variety of numerical calculations with little to no extra user effort. As the functions are evaluated after initial calculation, many of the gradient calculations are embarrassingly parallel given the branching nature of the DAG. This lends itself to SIMD hardware, enabling this to reach incredible speeds with recent models regularly exceeding billions of parameters. Due to this, GPUs are fundamental to the rise of ML, to the point where PyTorch itself allows

for all tensors, models, and function calls to reside and operate purely on the GPU.

From simple calculus principles, a gradient-based autodiff training scheme has emerged which harnesses the theory of differential geometry to allow for advanced generic modelling capabilities. While this brief introduction merely covers the minimal amount to understand the basic theory, there is a rich and deep framework underlying these models which is still being formalised. Extending this scheme to more complex systems should be clear, and this forms the fundamentals of how modern models like Φ are trained purely by the gradients of functions like $\mathcal{L}$ which quantify model performance.

1.2.2 Neural Network Architectures & Data Modality

Now the necessary theory and back-end is covered, it is clear how arbitrary differentiable functions can be optimized effectively. The form introduced in Eqn. (1.2.1.10) covers a general scheme extensible to cover nearly all models used across modern ML, and there are a collection of widely used functions which compose models like Φ. Using autodiff to implement the training scheme places strict constraints on what can be considered, however there is still a wealth of available parts to compose. The most common functions are those which are capable of approximating a diverse class of functions, and there is a significant amount of theoretical work covering the limits of what can be approximated efficiently[101–103]. In general, most methods utilise matrix multiplication operations composed with a non-linear activation function ρ to approximate complex intra-data relationships. This enables ML to take advantage of the speed of GPU-based matrix multiplication algorithms in both the forward and backward passes, leading to larger models becoming feasible and often consisting of billions of parameters. In deep models, each successive function in a model forms a single layer, parametrised by an independent set of floating point numbers. As it shall be shown, modern deep learning often consists of stitching together these functions in such a way best capable of modelling the underlying relationships within the data, or the data modality. The discovery of a new way of processing data is often tied to a major breakthrough in the area, and here the most common shall be introduced along with the context of their discovery.

Of the functions which have emerged as fundamental to modern ML, the artificial neuron, or linear layer, is by far the most widely used and successful. Today it is used as a fundamental part in all models, and it has stood the test of time from its first conception in the earliest research into ML during the 1950s[104,105]. Despite it's place as a driving force behind the subject, its form is surprisingly simple and similar to the multivariate matrix formulation of the basic model considered in Eqn. (1.2.1.1):

$$\hat{y} = \rho(x\mathbf{w}^T + \mathbf{b}) \tag{1.2.2.1}$$

Where $\hat{y}$ is the function output, x is the input. The parameters $\theta = \{\mathbf{w}, \mathbf{b}\}$ are a $\dim(\hat{y}) \times \dim(x)$ matrix known as the weight and a $\dim(y)$ vector called the bias, respectively. Emerging in the earliest literature on ML, proposed as a discrete model based on biological neurons, this function forms the backbone of all modern NNs. Its strength comes from its simplicity, speed, and expressiveness, capable of extending linear least squares like methodologies to non-linearly separable data. However it's power was not fully realised until researchers formed models of l layers feeding into each other, drastically increasing the complexity of functions it could model. Linking successive layers together allows for freedom in their size, allowing for models to increase or decrease the dimensionality, or hidden size, depending on the usage. In modern applications, these are known as linear or fully-connected layers, called such as the output a_j^k generated by weight j in layer k, w_{ij}^k, is dependent on all weights in the previous layer:

$$a_j^k = \rho(\sum_i a_i^{k-1} \times w_{ij}^k + b_j) \tag{1.2.2.2}$$

This architecture linking of the output from each previous layer of neurons to the next, led to these also being known as feed-forward layers. These regularly form the processing units of large models, fitting to data first reduced to a suitable dimensionality by other layers able to better respect the underlying data modality. Substituting such a network into Eqn. (1.2.1.10) leads to the commonly quoted form of backpropagation while avoiding the long derivation. The expressiveness of such a network can be seen by taking one of the simplest activation functions, ReLU:

$$\text{ReLU}(x) = \max(0, x) \tag{1.2.2.3}$$

While at first glance the simplicity may seem a poor basis for building powerful modelling systems, consider that with simply 2 inputs this transform, by removing some weights, can begin to build NAND or NOR logic gates required for functional completeness. Extending this to deeper models with more layers, and this begins to add support to the great modelling capabilities of NNs.

Feed-forward networks showed promising initial results on simple datasets, such as MNIST or early attempts at language modelling, however they struggled

to scale to the many practical uses practitioners had hoped for. Even with the increasing computational speed, the number of parameters scaled linearly with the size of the data, meaning tasks on larger, more complex datasets were computationally infeasible. Computer vision was an early problem where progress began halting due to this issue- using models to process images to perform tasks such as segmentation, object detection, and classification. Small images of 400×400 pixels required $160,000$ weights for simple scalar images, and as such this fast became the computational bottleneck. Factoring in full 3-colour images required at least tripling this computational load as the tensors representing images had additional channels, for example a 800 pixel square image being represented by a $(800, 800, 3)$ tensor, or $1,920,000$ elements. Additionally the fully-connected nature of these networks quickly became incredibly inefficient and posed more issues, local pixels were highly correlated and the variance in their size and location of objects of interest meant linear layers struggled to identify images under basic linear transforms.

Biology provided sufficient inspiration, and the groundwork for a new type of layer specific to image tasks emerged in the 1980's before being further refined in the 1990's[106,107]. These layers used a convolution operation to process images in small regions using a sliding window approach, initially formulated to model how visual cortices operated in local neuron patches[108]. By specifying a filter shape K as a tuple representing the size of a rectangular window, this can slide over the image to produce local outputs. The number of weights can then be greatly reduced to the product of the size of this filter and the number of channels by operating over small regions of the data, while encouraging convergence by only considering local relationships. Numerically, the convolution can be implemented as an element-wise multiplication at each possible window location, leading to a simple output equation when considering a flattened 2D-image with C channels:

$$\hat{y}_j = \rho\left(\sum_{k=1}^{C} \theta_{jk} * x_k + b_j\right) \tag{1.2.2.4}$$

The output shape can be controlled using 3 additional parameters, the stride s, padding p, and dilation d, yielding an output size of:

$$\dim(\hat{y}_j) = \left[\frac{\dim(x_j) + 2p_j - d_j(K_j - 1) - 1}{s_j} + 1\right]$$

These additional parameters control how the filter moves across the image, and which parts of the image multiplication is performed on. The index j specifies the output channel, which denotes the number of separate filters required. This allows each layer to have multiple independent filters, each capable of attuning to different features of the data and running in parallel on GPU hardware. Convolutional layers allowed researchers to iteratively reduce the size of images until they could be processed by additional linear layers[109].

Networks utilising these layers are known as Convolutional Neural Networks (CNNs), and quickly began to outperform human-expert designed algorithms at computer vision tasks. To aid this approach on especially large images, computer vision problems also introduced the now widespread pooling layers[110]. These further reduce the dimensionality using a sliding window approach to perform simple differentiable operations such as averaging, summing, or taking the maximum, taking their applicability to new scales. These approaches led to computer vision becoming one of the first areas which saw a true AI revolution, with image generation and classification now seeing widespread usage outside of the scientific community. These developments were strongly dependent of the speed of algorithms increasing with the introduction of GPUs[111]. Additionally, the convolution operator used is remarkably simple to extend to any dimensions, leading to convolutional layers seeing usage across problems involving $1D$ time series up to $4D$ spatiotemporal data and beyond.

The power of convolutional networks can be shown by the shift equivariance they display[112]. This means a discrete translation of CNNs output by some pixels is equivalent to the output of the CNN when the input undergoes the same translation. While it may seem a trivial fact, this is related to the underlying symmetries of the data. This can be seen as an explanation as to why CNNs are capable to generalise and identify components of an image across a collection of disparate images once trained with enough examples. In fact the true power

of CNNs was not fully clear until the distribution of ImageNet, a huge dataset consisting of 14-million captioned images[113]. The dataset catalysed the computer vision community, leading to some of the first major breakthroughs in modern ML[114]. This highlights not only the importance of respecting the underlying modality of the data, but also the importance of scale when achieving state-of-the-art results.

The next major breakthrough came in text and semantic processing, where researchers had struggled for years to numerically model the notion of meaning within language. Processing text became a difficult topic for various reasons. Early attempts at translating words to numerical vectors needed an effective hash-table of word embeddings, quickly found to require large dimensionality to represent the large space of potential word meanings[115]. Processing a sentence as a sequence of these large embeddings then suffered from many of known issues NNs had with sequential data: struggling with length for long sequences and redundancy for short. Attempts to circumvent this by using LSTM and GRU based Recurrent Neural Networks (RNNs), which processed a sequence element by element, introduced a memory vector accounting for previous elements[116]. In practice these suffered from instability when training, and introduced a preferred direction to processing text as sentences could only be considered sequentially causing too much emphasis on later elements[117]. To combat this, RNN researchers began to look towards methods capable of processing a whole sentence at once by attenuating to each element individually while avoiding spiking memory requirements. While early approaches applied this concept as an addition to existing RNN methodologies, soon the techniques began to come into their own[118]. With the seminal paper *"Attention is all you need"*, it was shown that solely using such layers could significantly improve performance[119]. Attention layers proposed that networks could learn how to attenuate during the learning process, by using 3 linear layers to project the data into separate vector spaces known as the query $\mathbf{Q} \in \mathcal{R}^{m \times d_k}$, key $\mathbf{K} \in \mathcal{R}^{n \times d_k}$, and value $\mathbf{V} \in \mathcal{R}^{n \times d_v}$. The now commonplace self-attention mechanism utilises the dot-product of $\mathbf{K}$ and $\mathbf{Q}$ to learn a weighting mask, analogous to the filter in

a CNN, over an entire input sequence $\mathbf{V}$:

$$\mathbf{Q} = \rho(\mathbf{X}\mathbf{W}_Q^T) \tag{1.2.2.5}$$

$$\mathbf{K} = \rho(\mathbf{X}\mathbf{W}_K^T) \tag{1.2.2.6}$$

$$\mathbf{V} = \rho(\mathbf{X}\mathbf{W}_V^T) \tag{1.2.2.7}$$

$$\hat{\mathbf{y}} = \text{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \in \mathcal{R}^{m \times d_v} \tag{1.2.2.8}$$

Here $\mathbf{X}$ is a sequence of vectors evaluated, and the scaling factor of d_k was found to encourage stability during training. The softmax function converts the output into a probability distribution via the transform:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{1.2.2.9}$$

These layers allow models to interpret sequential data at run time by prioritising specific elements relevant to the task. Attention layers allow NNs to effectively process sequences requiring only 3 times the parameters of a basic linear layer on a single entry. Similar to how convolutional layers can be parallelised into different channels, attention layers admit a similar strategy via Multi-Head Attention, where the results of α different layers are concatenated together and weighted:

$$\text{Multi-Head}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\text{Att}_1(\mathbf{Q}, \mathbf{K}, \mathbf{V})|| \dots ||\text{Att}_\alpha(\mathbf{Q}, \mathbf{K}, \mathbf{V})] \cdot W_C \tag{1.2.2.10}$$

Where $[\cdot||\cdot]$ denotes concatenation Models built upon this type of layer are commonly known as transformers, and have shown a wide-range of sequence modelling applications outside of text processing[120–122]. Using attention-heads in this way is an incredibly effective way to increase the modelling capabilities, and has shown incredible scaling properties becoming the backbone of modern LLMs. In the majority of these models, the layers utilise skip connections where the output of layer l, $\hat{y}^l$, is an incremental difference on the previous layer:

$$\hat{y}^l = \hat{y}^{l-1} + \text{Multi-Head}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \tag{1.2.2.11}$$

This allows for incredibly deep models to make small differences each layer, and is key to why transformer architectures scale so well. After being introduced in the 1980s in residual NNs, this type of layer connection now sees widespread usage. Utilising this along with the inherently parallel nature of multi-head approaches, and it becomes clear why these models regularly approach billions of parameters.

An equally important property of attention layers is their permutation equivariance. Under a permutation represented by the matrix A, given by any valid reordering of the identity matrix:

$$\mathrm{Att}(A\mathbf{Q}, A\mathbf{K}, A\mathbf{V}) = A\mathrm{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$$

In fact due to the softmax function also being permutation equivariant, any reordering of $\mathbf{K}$ and $\mathbf{V}$ cancel as $A^T = A^{-1}$. This means any change in the query rows changes the output similarly, a property key to effectively modelling sequences where the underlying relationship is built from the sequence as a whole, not any individual element. This avoids the pitfalls of LSTM-based approaches, where their auto-regressive nature introduces a preferred direction by processing vector elements one-by-one. This again is an approach which found success through respecting to the underlying data modality, as the sentiment of a sentence is often only clear from the whole sequence in context. However it does introduce a separate issue, as ordering is often key to meaning, such as a sentence featuring two adverbs, or a post-positive adjective common across romance languages. To alleviate this, positional encoding was introduced to add additional information related to a words location within a sentence and there are several viable methods to do so[123,124]. Using a different input rather than x to calculate $\mathbf{Q}$ in Eqn. (1.2.2.5) allows modulation of a sequence with respect to another data source, known as cross-attention. This has been successfully applied to tasks depending on several key interlinked data sources, for example in protein sequence prediction where layers predicting atom locations can attend to sequence data[125].

Graph Neural Networks

Clearly much of the success of ML comes from using models capable of processing data in ways which respect its modality. When faced with issues pertaining to

networks and $3D$ point-cloud data, research again pointed to the development of layers capable of respecting arbitrary physical geometries. In doing so, they began to generalise NNs to graphs- perhaps the most prevalent data structure of all[126–129]. This led to the development of Graph Neural Networks (GNNs), which in turn were realised to encompass many of the layer types previously considered. As the majority of data used across biophysics, and therefore this thesis, consists of examples of graphs these layers form a critical part of this work. A graph G with N nodes can be described as a set of vertices $V = \{v_1, ..., v_N\}$ and edges $e_{ij} \in E$, which are often represented as a binary adjacency matrix, $\mathbf{A}$, where:

$$A_{ij} = \begin{cases} 1, & \text{if } e_{ij} \in E \\ 0, & \text{otherwise} \end{cases} \tag{1.2.2.12}$$

Each node, v_i, can be associated with a d_{feat} dimensional feature vector, F_i, similarly admitting a matrix representation as a $(N \times d_{\text{feat}})$ matrix, $\mathbf{F}$. These features can be derived from the underlying data, generated at run time in the case of a featureless graph, or omitted entirely. Additionally in some applications features for edges can be considered, although these are omitted for brevity here as they are not used in this work[130]. This makes it possible to define a general graph structure as a tuple:

$$G = (V, E, F) \tag{1.2.2.13}$$

In practice, a complete specification of the graph structure can be achieved solely through the matrices $\mathbf{A}$ and $\mathbf{F}$.

Applying NNs to this data type can be achieved by adopting any differentiable mapping operating on the features of a locally connected neighbourhood specified by $\mathbf{A}$. Several schemes have attempted to consolidate all GNNs into a simple framework, including the message-passing formalism[131]. Here a GNN-based encoder maps a node v_i in G into a context-aware embedding using T separate layers via an iterative process[132]:

$$\begin{aligned} m_i^{t+1} &= \sum_{j \in \mathcal{N}(v_i)} M_t(h_i^t, h_j^t, e_{ij}) \\ h_i^{t+1} &= U_t(h_i^t, m_i^{t+1}) \end{aligned} \tag{1.2.2.14}$$

where $t \in \{1, ..., T\}$ denotes the current network depth, $\mathcal{N}(v_i)$ is the local neighbourhood of all nodes connected to node v_i by edges e_{ij}, and h_i^t represents the hidden representation of node v_i at depth t. For consistency, the initial node feature vectors F_i are represented by h_i^0.
The functions M_t and U_t are commonly implemented as neural networks, known as the message-passing function and vertex update function respectively.
This general schema can be used to represent other types of layers by appropriately constructing the adjacency matrix and choosing the form of the message-passing and vertex update functions. For example, the similarities between this and the CNN layers are clear when considering all pixels to be nodes and the column $\mathbf{A_i}$ to represent the local neighbourhood around pixel i. Similarly considering a fully-connected matrix A and implementing 3 linear layers followed by appropriate dot-products for M_t, and attention layers come under this remit.
This is an unsurprising consequence of the ubiquity and expressiveness of the graph data structure. In fact, one may consider graphs to be one of the most primitive structures known, and there is significant theoretical work drawing the links between these type of networks and differential geometric principles.

The exact formula of M and U in Eqn. (1.2.2.14) depends on the type of graph layer used. GNNs were first popularised in the seminal work on the subject, which looked to generalise the convolutional operator of CNNs to graph data[133]. It proposed the Graph Convolutional Network (GCN), one of the simplest examples of a GNN. It relies upon the graph Laplacian L, defined as:

$$L = I_n - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \tag{1.2.2.15}$$

Where D is the matrix where element D_{ii} is the degree of node i and zero everywhere else. The message passing function then takes the form:

$$\begin{aligned} M_t(h_i^t, h_j^t) &= L_{ij} h_w^t \\ U_t(h_i^t, m_i^{t+1}) &= \rho((W^t)^T m^{t+1}) \end{aligned} \tag{1.2.2.16}$$

GraphSAGE layers built from this to introduce additional freedom in the message-passing function, linking the networks with standard NN techniques[134]. They proposed either the sum or mean, and using a vertex update function which

concatenates these expressions and feeds them into a feed-forward neural network directly:

$$\begin{aligned} m_i^{t+1} &= \begin{cases} \frac{1}{n}\sum_{j\in\mathcal{N}(v_i)} h_j^t, & \text{if mean} \\ \max\limits_{j\in\mathcal{N}(v_i)} \sigma\Big(\mathbf{W}_{\text{pool}} h_j^t + b\Big) & \text{if max} \end{cases} \\ h_i^{t+1} &= \rho\Big(\mathbf{W}^{\mathbf{k}}[h_i^t || m_i^{t+1}]\Big) \end{aligned} \tag{1.2.2.17}$$

where $[\cdot||\cdot]$ denotes concatenation. This uses separate weight matrices for the pooling operation $\mathbf{W}_{\text{pool}}$ and final layer output $\mathbf{W}^{\mathbf{k}}$ which can be learnt independently.

Graph Attention Network (GAT) layers replace this with a formulation inspired by scaled dot-product attention seen in transformer layers[135]. This differs by performing the aggregation step individually for each node in a neighbourhood. The GATv2 layer improves upon the original layer by changing its attention mechanism to be dependent on the query node, further extending the expressiveness of the model[136]. This differs by attempting to give the network the ability to attune to the presence of specific nodes that may interact with the current node by calculating a context-aware attention mask, which is then used to perform a weighted vertex update:

$$m(h_i^t, h_j^t) = \underset{j\in\mathcal{N}(v_i)}{\text{softmax}}\Big(\mathbf{a}^T \rho(\mathbf{W}\cdot[h_i^t||h_j^t])\Big) \tag{1.2.2.18}$$

$$h_i^{t+1} = \sigma\Big(\sum_{j\in\mathcal{N}(v_i)} m(h_i^t, h_j^t)\cdot \mathbf{W} h_j^t\Big) \tag{1.2.2.19}$$

where $\mathbf{a}$ are an additional set of learnt weights, and the process can be generalised across multiple attention heads by concatenating messages.

1.2.3 Practical ML

Now that the relevant theory is covered, the discussion can move towards the pragmatic side of using ML techniques for modelling tasks. Overtime, a collection of general strategies have emerged to combat many of the empirically observed

issues that arise when training these models. The task of training is commonly separated into 3 different paradigms- supervised and unsupervised learning, and generative modelling. The boundary between these is not necessarily distinct and terms are often used together or interchangeably. The goal across all is ubiquitous- for the model to learn a mapping from inputs to outputs that generalizes well and is optimised for some criteria.

Supervised learning classically refers to approaches where the model is trained on a labelled dataset, meaning each input comes with a corresponding ground truth label which may require substantial human effort to collect[137]. Common supervised learning tasks can be separated into classification, where the goal is to assign inputs to predefined categories, and regression, where the goal is to predict continuous values. Supervised learning forms the foundation for many practical applications where labelled data is abundant, such as image recognition, medical diagnostics, and speech recognition. In computational biology, it encompasses common tasks such as protein structure prediction where labels take the form of experimental protein structures, predicting point mutations, and regression of complex calculation results.

Unsupervised learning involves training a model on data without any explicit labels[138,139]. These models form the basis for latent space models introduced in Section (2.2), where their utility is as a dimensionality reduction or compression tool to identify complex patterns within the data. It is common in such usages for the target outputs to be identical to the inputs, and instead the network architecture is organised to compress the necessary information into as small a representation as possible. This also covers non-standard tools such as the clustering networks in Case Study 3, which are provided with only a criterion rather than any explicit target. Unsupervised learning is particularly useful when labelled data is scarce or too expensive to obtain, or when datasets are too large for human experts to manually label. It is often employed in exploratory data analysis or feature learning[140].

Generative modelling is the newest paradigm to emerge from ML, popularised with the advent of GANs in the mid-2010s and more recently with the state-of-the-art results achieved by diffusion models in image generation and protein structure prediction[141–143]. Such models are identifiable by their usage of noise in both

the training and inference process, forcing models to effectively generalise over larger regions of their latent space and enabling advanced sampling techniques. While such models are not considered in much detail here, it is important to note these paradigms are not discrete. The models considered in Case Study 1 are both unsupervised and generative, although their end purpose is for what would commonly be considered a unsupervised task of dimensionality reduction. Similarly the prediction networks used in Case Study 5 use a GNN-based encoder network to map to a latent space of potential molecules before using a feed-forward decoder to produce a prediction for a regression task. These labels are a simplification based more on the intention of the model rather than its internal architecture.

Once the task has been identified, and the necessary NN structure decided upon, the next task is to build a model. In frameworks such as PyTorch, models are constructed as child-classes which inherit from the `nn.Module` abstract base-class. This parent class provides the necessary structure to interact with various other members of the PyTorch tooling system. The code in Listing 12 shows an example of a basic 4-layer NN constructed using a custom linear layer implementation for a given input, output, and hidden dimension.

Both classes take `nn.Module` as a parent and call their constructor in order to setup the necessary class methods. This includes defining the abstract `forward` method which requires overloading for this class to function. The method is called by the `__call__` operator using the syntax `model(data)`, and allows users to specify the forward pass of the model. Due to the aforementioned flexibility of autodiff gradients this method can implement almost any control sequence available in Python, and as long as it is comprised of basic PyTorch functions the library will automatically generate the backward pass when needed. In the `LinearLayer` class the weight and bias of a linear layer are generated by constructing tensors with the `requires_grad` attribute. This signifies to Torch this will be later be used by autograd, and causes the engine to record the sequence of operations. This is important as although autograd is very fast, this is mostly due to the innate parallelism of the library and this task, as at the backward pass all necessary values are already calculated and can be done in parallel. The process of following a tensor and caching the operations in order can become very expensive on vital RAM resources, so this should only be done when necessary.

```python
import torch
import torch.nn as nn

def initialise_w(parameter):
    return nn.init.ones_(parameter)

def initialise_b(parameter):
    return nn.init.zeros_(parameter)

class LinearLayer(nn.Module):
    def __init__(self,
                 input_size: int,
                 output_size: int,
                 use_activation: bool=True):
        super().__init__()

        w = torch.empty((input_size, output_size), requires_grad=True)
        self.weight = nn.Parameter(initialise_w(w))

        b = torch.empty((output_size), requires_grad=True)
        self.bias = nn.Parameter(initialise_b(b))
        if use_activation:
            self.activation = nn.ReLU()
        else:
            self.activation = nn.Identity()

    def forward(self, x):
        return self.activation(x @ self.weight + self.bias)


class NeuralNetwork(nn.Module):
    def __init__(self,
                 input_size: int,
                 hidden_size: int,
                 output_size: int,
                 n_layers: int=4):
        super().__init__()
        self.layers = nn.ModuleList()
        self.layers.append(LinearLayer(input_size, hidden_size))
        for _ in range(n_layers-2):
            self.layers.append(LinearLayer(hidden_size, hidden_size))
        self.layers.append(LinearLayer(hidden_size, output_size,
                           use_activation=False))

    def forward(self, x):
        for layer in self.layers:
            x = layer(x)
        return x
```

Listing 12: Simple 4-layer NN constructed using PyTorch.

The functions `initialise_w` and `initialise_b` then set the weight and bias to all 1's and 0's respectively. This is not a reasonable initialisation, but is included as it's an often overlooked part of ML design, as it shall be seen in Case Study 3 how large an impact initialisation can have on a model's performance. There are several different initialisation schemes[144,145]. The call to the initialiser is wrapped in a call to `nn.Parameter`, which tells Torch to class this tensor as a parameter of the parent class. This means it will be found by a call to `NeuralNetwork.Parameters()`, which shall be the input to the optimizer.

Many different optimisation strategies are available, but the specific implementations of these differ very little from the basic gradient descent form specified in Eqn. (1.2.1.3). These algorithms are central to the training process and are natively implemented in most tensor libraries. While simple methods such as Stochastic Gradient Descent (SGD) are still widely used, advanced algorithms like Adam and RMSProp have become popular due to their adaptive learning rates and faster convergence[146–148]. All of these adaptations are still based on the same principle of gradients, but look to improve training in different areas through introducing general concepts such as weight-decay, momentum, and various fixes for numerical stability. Adam is capable of adjusting learning rates for each parameter based on the first and second moments of gradients. RMSProp on the other hand, is designed to handle non-stationary objectives by normalizing gradients based on a moving average of recent magnitudes. These algorithms have been proven effective for training deep learning models, and many of these additions help when dealing with noisy or sparse gradients. Knowing which is best to use, and the most effective parameters, is often a trial and error process depending on the problem.

Listing 13 shows an example of training the previously defined NN using PyTorch, iterating the parameters for every example across a dataset. It also shows the simplicity of utilising GPU-accelerated code in a tensor library. Simply using the `.to(device)` method on any PyTorch object sends its data to the specified device, in this case a CUDA GPU. Any subsequent calculations performed between tensors on the same device will be GPU-enabled, and any operations between two tensor residing on different devices will raise an error.

```
from torch.optim import Adam

device = "cuda"
model = NeuralNetwork(4, 8, 1).to(device)
optimizer = Adam(model.parameters(), lr=1e-5)
for (sample_data, target) in dataset:
    sample_data.to(device)
    prediction = model(sample_data)
    error = loss(sample_data, target)
    error.backward()
    optimizer.step()
    optimizer.zero_grad()
```

Listing 13: An example of an elementary training scheme in PyTorch using the Adam optimizer and GPU acceleration.

The optimisation process is consistent for all optimizers implemented in PyTorch. In the constructor, the model's parameters are set to be iterated with an initial step size of 10^{-5}. The `optimizer.step()` function simply makes this change depending on the current gradients, and `optimizer.zero_grad()` resets all gradients to 0.

In the optimization example, the model parameters are iterated for each training example. Training ML algorithms is always a trade-off between computational speed, accuracy, and generalisation capability. An ideal model is often unattainable, and instead a compromise must be made. This notion has been about since the early days of statistical modelling, under the name the bias-variance trade-off[149,150]. Training on each example individually means early examples can push the network into a poor region of parameter space for generalisation, leading results to be heavily dependent on the ordering of the dataset. This can result in a common error known as overfitting, where the network loses its generalisation capability instead focusing on reproducing the results of the training set. It has been shown that for stability and inference accuracy, it is better to instead group the data into subsets and take averaged gradients in a process known as batching[151–153]. On the other end of the scale, memory requirements or very disparate gradients can prevent batch sizes from becoming too large, and it is often not feasible to process a dataset as one. Again, this is data and problem dependent, and different schemes have shown different levels of success for various problems

Often the final objective of training is to get a model able to interpolate over new examples, therefore it is critical to be able to effectively quantify its performance on new or unseen data. To be able to better assess this capability, the dataset is partitioned into three disjoint subsets: the training set used for fitting the model, the test set reserved for evaluating its performance, and a validation set[154,155]. A proper test-training-validation split ensures that the model's performance is evaluated on unseen data, and as such reflects its ability to model rather than memorize the training data. Popular ratios for this split include 80/10/10 or 70/15/15, but the choice can vary depending on the size and nature of the dataset.

As this section has shown, there are few practical considerations in ML which have a definite answer applicable to all use cases. In order to effectively select a batch size, optimizer, learning rate, and model dimensionality, thorough testing must be performed. This process is known as hyper-parameter optimization and is performed at the early stages of model development, where the settings and parameters of a model and training scheme are tuned to better performance[156–159]. Unlike the model parameters which are learned during training, hyper-parameters are any predefined quantity and can significantly affect model performance. Methods such as grid search, random search, and more sophisticated techniques like Bayesian or evolutionary optimization aim to efficiently explore the hyper-parameter space to identify the optimal configuration[160]. All these techniques share a similar philosophy of repeating the training process while varying the parameters, and comparing the accuracy of the model. This is where the validation set comes into use, as it provides a distinct dataset for hyper-parameter validation while reserving some data for final verification. Using the data this way avoids building models specialised to the test set, providing more support for a trained models capability to generalise to new data.

1.3 Molecular Dynamics

In the computational study of biological systems, Molecular Dynamics (MD) has become the mainstay methodology as the workhorse underpinning investigating the dynamical nature of biomolecular systems[161–163]. At its core, it is a deterministic algorithm for computing the classical kinetic behaviour of atomistic systems over short time-scales. As a term it encompasses both the simulation theory and tooling that has developed for these aims. MD software is packaged into different engines, which utilise their own collection of numerical implementations to achieve common simulations goals. Behind all these different facades the theory remains fairly consistent, despite the many different interpretations and implementations of the various processes required. Perhaps its greatest strength is its wide applicability as a general simulation engine, lending itself to studying a variety of systems in biology, chemistry, and physics. These simulations allow a number of statistical and physical quantities to be estimated, and there is a large formal body of literature relating to rigorously defining these and how to best estimate them from simulation results.

Drug design and structural biology are perhaps the fields which have most adopted MD, using it consistently across countless studies and systems[164,165]. Here there is a great focus on the $3D$-structures produced by these algorithms, as researchers look to elucidate the fundamentally dynamic functional processes of proteins. MD also provides the theoretical basis for many advanced methods which look to calculate binding or catalytic rates, such as those considered in Case Study 4 and 5. Such methods find wide usage, and these will be touched on throughout the work when relevant. Instead this section looks to introduce the topic and cover the basics of MD. These principles underpin much of the simulation techniques used throughout this work, and provide the basis for the structural viewpoint common throughout research in the area.

1.3.1 General Theory

The central concept of MD is to calculate the classical time development of a system on an atomic scale given an initial configuration. It aims to do so by solving the Newtonian equations of motions for a collection of N atoms of mass

m_i and positions $\mathbf{r}_i$ by evaluating the force $\mathbf{F}_i$ upon each atom. Indexing by the variable $i = \{1, \ldots, N\}$, Newton's second law yields the following equation for the force:

$$\mathbf{F}_i = m_i \frac{d^2\mathbf{r}_i}{dt^2} \tag{1.3.1.1}$$

By considering the system to be closed it is possible to express this alternately using the Lagrangian formalism of classical mechanics.

Associating the positions $\mathbf{r}_i$ with the generalised coordinates, the Lagrangian for the whole system is given by the difference between the kinetic and potential energy, expressed in terms of r, the time t, and the positional time derivatives $\dot{r} = \frac{d\mathbf{r}}{dt}$:

$$\mathcal{L}(\mathbf{r}, \dot{\mathbf{r}}, t) = \sum_{i=1}^{N} \frac{m\dot{\mathbf{r}}_i^2}{2} - U(\mathbf{r}) \tag{1.3.1.2}$$

The important quantity, here and throughout the topic, is the form of this potential energy function U.

Using the Euler-Lagrange equation, the equation of motion is given by:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{r}}_i}\right) = \frac{\partial \mathcal{L}}{\partial \mathbf{r}_i} \tag{1.3.1.3}$$

Substituting Eqn. (1.3.1.1) into this provides the relationship between U and F_i:

$$\mathbf{F}_i = -\frac{\partial U(\mathbf{r}_i)}{\partial \mathbf{r}_i} = -\Delta U(\mathbf{r}_i) \tag{1.3.1.4}$$

This is the basic equation whose solutions are explored and solved using MD engines and techniques. Several explicit forms of $U(\mathbf{r}_i)$ are available, with some suited for specific modelling tasks or derived using different techniques through a process known as parameterisation.

Equivalent derivations exist in terms of the Hamiltonian $\mathcal{H}$, defined as the sum of kinetic and potential energies in terms of generalised coordinates $\mathbf{q}$ and momenta $\mathbf{p}$:

$$\mathcal{H}(\mathbf{p}, \mathbf{q}, t) = \sum_{i=1}^{N} \mathbf{p}_i \dot{\mathbf{r}}_i - \mathcal{L}(\mathbf{r}, \dot{\mathbf{r}}, t) \tag{1.3.1.5}$$

This is equivalent to the Legendre transform of Eqn. (1.3.1.2).

1.3.2 Forcefields

While a chosen engine provides the numerical and computational techniques used to simulate a dynamical system, the Force-field (FF) is of equal importance as it dictates the inter-molecular interactions between the atoms of the system. FF refers to the particular functional form of the potential energy function U coupled with a particular set of parameters which are used as constants in the force calculations. Calculations based on just these forces are often referred to as Molecular Mechanics (MM). The development and empirical derivation of FFs is a state-of-the-art science, and they come in several different flavours suited for different applications[166–168]. These include variations in how atoms are considered, leading to all-atom, united-atom, coarse-grained, and more recent NN-based approaches.

For speed considerations coarse-grained FFs consider groups of atoms as single entities, united-atom approaches accommodate non-polar hydrogen's into the terms for the atoms they are bonded to, and hybrid approaches mix the two depending on necessity. ML approaches take a different approach by learning a FF directly from empirical data using NNs trained on ab-initio QM results, looking to extrapolate accurate techniques beyond their usual capacity. None of these extended approaches are used in this work, but provide differing approaches to increase the size, speed, or accuracy of simulations depending on the task. Significant literature and several reviews are available summarising their theory and applications[169].

Instead this work focuses on harmonic all-atom additive, or class I, FFs, which are the most commonly encountered FFs in biomolecular simulation. They calculate a value for U by summing over all contributions from every atom in the system, differentiating atoms and bonds into discrete types utilising different constants. Developing FFs is an iterative procedure, with many variants being released and refined over time. Most notable amongst these are the CHARMM and AMBER families of FFs, two very popular examples originating from early work in the 1980s[170–173]. They are still used over the aforementioned more efficient approaches as physical accuracy is a major concern. The computational load is viewed as a worthy compromise when the results can be supported by clear

physical reasoning and backed up with experimental results. Having per-atom data also enables a greater degree of analysis when it comes to parsing the results, and so makes a clear choice when investigating drug design targets.

The functional form of the potential energy in this category separates contributions between bonded and non-bonded interactions:

$$U_{\text{total}} = U_{\text{bonded}} + U_{\text{nonbonded}} \tag{1.3.2.1}$$

These may be further divided based upon the individual forces they look to model. The bonded contributions may be decomposed into 3 main parts:

$$U_{\text{bonded}} = U_{\text{bond}} + U_{\text{angle}} + U_{\text{dihedral}} \tag{1.3.2.2}$$

The harmonic part of the conventional name for this type of FF comes from the way these forces are modelled using harmonic terms derived from the equation for the energy stored in a simple harmonic oscillator:

$$U_{\text{SHO}}(x) = \frac{1}{2}k(x - x_0)^2 \tag{1.3.2.3}$$

For effective parameterization, this only requires two values: a force constant k and equilibrium deviation x_0 at which the energy contribution is 0. The first term in Eqn. (1.3.2.2) relates to the length of a bond d, and increases the potential energy with the displacement from an equilibrium length d_0:

$$U_{\text{bond}}(d) = k_b(d - d_0)^2 \tag{1.3.2.4}$$

This leads to a restorative force towards equilibrium, and in practice during the force calculation stage is summed between all bonded pairs of atoms. A similar thought is used to model the angle of bonds θ as they deviate from θ_0:

$$U_{\text{bond}}(\theta) = k_\theta(\theta - \theta_0)^2 \tag{1.3.2.5}$$

This term is calculating over all triplets of bonded atoms.
It is important to note while these simple approximations are effective when the

deviation from equilibrium is not too great, they struggle to accurately model more strenuous situations. In these circumstances they can be supplemented with more accurate formulas, such as the Morse potential which are common in anharmonic FFs[174]. Forms of U which use these extended terms are known as Class II and III FFs, and offer a higher accuracy in non-standard situations at the expense of efficiency.

Due to their 2π symmetry, torsional contributions from the bonds are commonly modelled using a Fourier series:

$$U_{\text{dihedral}}(\varphi) = (k_\varphi)_n(1 + \cos n\varphi - \gamma_n) \tag{1.3.2.6}$$

which enables an effective parameterization while preserving the periodicity. These are calculated from the dihedral angles φ, formed at the intersections of the two planes formed bonded quadruplets of atoms. Note here the subscript n which denotes some FFs may associate varying force constants k_φ and phase constants γ to different values of n for a more accurate modelling of complex potentials. Many FFs additionally separate torsional contributions into improper and proper torsion. This is based on whether the 4 atoms taken are consecutively bonded or are a set of 3 bound to a central atom. Separating the two is used to enforce planarity for example in ring-like structures. Different FFs use either Eqn. (1.3.2.6), for example in the AMBER family where improper torsions are handled implicitly, or using a form like Eqn. (1.3.2.3), such as in CHARMM FFs where they are calculated using a separate term.

Additionally, class II and III FFs often include coupling terms which further model the relationships between the different bonded elements[175]. These are constructed to better represent the underlying potential energy surface, which may for example have a different sensitivity to bond stretching at certain valence angles[176]. Such terms often take non-harmonic forms. Additional terms require more computation, alongside more data to accurately derive accurate parameters for all atom types. Although the different classes of FF have found considerable use in the study of small molecules, the performance aspect has led to most of the biomolecular modelling community being hesitant to adopt anything but class I FFs. Instead, they look to further optimize the simple forms already mentioned via

more accurate parameterization procedures or the inclusion of simple corrective terms which seek to additively modify energy contributions. The CHARMM FF family introduce a couple of terms to their bonded contributions to model bond-angle dependence and more accurately simulate secondary structures[177,178]. A harmonic term known as the Urey-Bradley potential is introduced to estimate the coupling between the angle and bond length. It does so by calculating the distance between the first and third atoms δ in each consecutive triplet of bonded atoms:

$$U_{\mathrm{UB}}(\delta) = k_{\mathrm{UB}}(\delta - \delta_0)^2 \tag{1.3.2.7}$$

While this is functionally the same as a bonded term, this distance is heavily dependent upon both bonds angles and distances. Therefore it allows for inclusion while keeping all terms importantly harmonic and not requiring any prior evaluations of bonded terms.
In addition to this, since 2002 the CHARMM FF has included several correctional terms for the protein backbone structure, known as grid-based energy Correction Maps or CMAPs. Parametrised from the results of ab initio QM simulations of common secondary structures, these terms are a function of the protein backbone dihedral angles. They are a continuous bicubic interpolation of simulation results taken over 15 degree intervals, and model the unincorporated QM effects that often lead to the stability and formation of secondary structures. These corrections have been shown to corroborate better with experimental results than the unmodified CHARMM FF, leading to their recent inclusion in the AMBER family in 2020 with the release of the ff19SB version[179,180].

The nonbonded contributions to the potential energy in Eqn. (1.3.2.1) can be similarly split into an Electromagnetic (EM) contribution U_{EM} and one from van der Waals (VdW) forces U_{VdW}:

$$U_{\mathrm{nonbonded}} = U_{\mathrm{EM}} + U_{\mathrm{VdW}} \tag{1.3.2.8}$$

The EM term is calculated using the classical Coulomb potential:

$$U_{\mathrm{EM}}(d_{ij}) = \frac{q_i q_j}{4\pi\epsilon_0 d_{ij}} \tag{1.3.2.9}$$

For evaluation this requires the the distance between two pairs of atoms d_{ij} along with two parametrized fixed partial charges q_i and q_j. This term is problematic for several reasons. Modelling the long-range effects causes significant issues in the early days of MD before Ewald methods, discussed in Section (1.3.5), provided a functioning workaround. Unresolved in class I FFs however are the modelling deficiencies of the point charge model. The fixed point charge model is unable to account for polarization and the additive treatment fails to fully incorporate screening effects. Class III and ML based FFs have evolved to attempt to better model this behaviour, and find significant usage for nucleic acid modelling tasks where point charge treatments fail[181,182]. However many ways to incorporate these effects are beyond the reach of class I FFs without a significant reworking of their core functionality.

The VdW force is a conglomerate of complex inter-molecular interactions used to model behaviours that are typically not considered in classical Newtonian theory, but which are still weakly present at the energy scales considered in MD. This includes attractive forces caused by temporarily fluctuating dipoles or permanently polar molecules and repulsive forces caused simply by close proximity, but falls short of any realistic polarization modelling. These interaction are nearly always represented using a Lennard-Jones (LJ) 12-6 potential, named after the powers in its defining equation:

$$U_{\mathrm{VdW}}(r_{ij}) = 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] \tag{1.3.2.10}$$

This results in a force that is repulsive at short distances and attractive over medium distances, before fading quickly to zero at long range. The σ term relates to the radius of the atom considered as a solid sphere, corresponding to the distance at which repulsion takes place. Whereas ϵ models the strength of the induced dipole of the atom, controlling the depth of the attractive well. Despite the double index on each parameter, the constants σ_{ij} and ϵ_{ij} are derived from individual values associated with each atom type- σ_i and ϵ_i. This means each atom type will require only 2 LJ parameters. They are related using equations are known as the Lorentz-Berthelot rules, which for two atoms of type i and j

has the form:

$$\sigma_{ij} = \frac{\sigma_i + \sigma_j}{2} \tag{1.3.2.11}$$

$$\epsilon_{ij} = \sqrt{\epsilon_i \epsilon_j} \tag{1.3.2.12}$$

Other ways to combine parameters are also available, but the above rules are significantly more widely used despite some concerns for their accuracy[183–185]. While several extensions to the LJ potential have been proposed, again the biomolecular modelling community has been somewhat hesitant to widely modify these terms given the ease of parameterization, computational efficiency, and experimental correspondence[186,187].

Each of these equations depends upon a set of constants which are also provided by the FF, quantifying the numerical contribution of each force. It is common for each FF family to release several variants concurrently, each of which is suited for a particular type of modelling. The process of parameterization encompasses a diverse range of techniques which are constantly updated and refined. They are generally rated based on how well they can reproduce key experimental or theoretical results, and as such can differ greatly based on the data used to empirically fit the constants. For example some FFs are based on reproducing measured dielectric constants, thermodynamic properties, or spectroscopic data, while others aim to replicate the results of ab initio QM calculations through methods such as force matching. An overview of the methods used to parametrise FFs is beyond the scope of this document, and instead interested readers are pointed towards available reviews on the subject[188–191].

1.3.3 Solvent

As the inside of the cell is an aqueous environment, to be accurate when modelling any protein the system must account for the effects and properties of water molecules[192]. Many studies highlight the critical importance of water, for all biological processes including protein conformation and ligand coordination[193–195]. It is impossible to separate biological systems from the environment in which they evolved, and as such there are several strategies to computationally model

water. They are separated into two main branches depending on whether the model considers explicit or implicit water molecules. Implicit modelling attempts to circumvent the computationally expensive problem of calculating individual forces and updating positions for the each of the many water molecules required. It achieves this by splitting the potential energy contributions into those that come from internal interactions in the protein U_{Protein}, as if the system was in a vacuum, and those coming from the solvent U_{solvent}, equivalent to the energy required to then solvate the system:

$$U_{\text{total}} = U_{\text{Protein}} + U_{\text{solvent}} \tag{1.3.3.1}$$

Implicit water modelling attempts to estimate U_{solvent} using a variety of continuum techniques, including classical electromagnetism and mean-field treatments. Through these methods, they achieve a significant speed advantage over explicit models, however the large number of assumptions required leads to a significant loss of accuracy[196]. As such these models are not considered or used in this work for general simulation purposes, although they find application in Chapter (3.1) to estimate solvation energy differences when assessing ligand binding.

Considering explicit water molecules is significantly more common when simulating proteins, and has shown excellent accuracy approximating experimental results. Since the beginning of research in the area there have been a number of different approaches considered, leading to a large variety of models available with over 45 being considered in one 2002 review[197]. Despite this, few have found their way into general usage. The Transferable Inter-molecular Potential (TIP) models have become some of the most widely used, including for modelling tasks throughout this work[198]. This is due to their computational simplicity while still corroborating with experimental data[199]. Unlike other molecules in MD simulations, these cast the bonds within water as rigid and as such only interact with the rest of the system via contributions to the non-bonded term $U_{\text{nonbonded}}$:

$$U_{\text{TIP}} = U_{\text{EM}} + U_{\text{VdW}} \tag{1.3.3.2}$$

The functional implementations of these models come in several versions denoted as TIP-nP. Here n is an integer representing the number of points used to represent a single molecule of water. TIP-3P intuitively has 3 points, each representing a different atom of a water. TIP-4P supplements this with an additional point along the bisection of the HOH angle to place the negative charge towards the hydrogen atoms to decrease the induced dipole moment. Higher values of n better model the charge distribution, however this can come at great cost depending on the number of molecules considered. It is important to note there are other options which have been shown to more accurately model the properties of water. These include SPC models, which supplements the energy with a polarization term and allow for bond flexibility, and the CC-pol family which is fitted to the results of ab initio QM calculations[200,201]. It is the decision of the researcher as to whether the system in question requires such an accurate representation of water properties, as it always to the detriment of the speed of the calculation. As such the simplicity of the TIP models make them difficult to beat due to their ease of calculation, and hence they find the widest usage.

It is often useful to additionally introduce ions into the system alongside water molecules. This can be to negate any net charge or to model more complex solvents with a higher salt concentration, for example in membrane simulations[202]. It shall be shown a charge neutral system is necessary for advanced methods to deal with long-range electrostatic calculations outlined in Section (1.3.5)[203]. Most commonly sodium and chlorine ions are used due to their biological availability, although in theory any ions with a net-charge can be used. There are several strategies that can be used to place the ions within the solvent, but the most common is to replace water molecules through a Monte-Carlo process[204].

1.3.4 Integrators

No widely used force-field accepts an analytical solution, and as such researchers turn to numerical methods to iteratively solve these equations over short discrete time-steps Δt. The exact equations used is known as the integrator, and there are a large number of choices available[205]. These range from simple methods building from the most basic Euler methods, to more complex approaches using

higher-order Runge-Kutta methods and Lie group integrators for example[206,207]. There are many good reviews and comparisons on this topic, and at its core this theory coincides with the greater discussion about finite-difference methods used across physics. Most methodologies are derived from Taylor expansions of Eqn. (1.3.1.4).

One of the simplest methods which still finds a significant amount of general usage today is the leap-frog integrator. This is the default in the popular MD engine GROMACS and a slightly altered version is used in the AMBER engine[208,209]. It is a second-order method while requiring the same number of evaluations as the simple Euler method. The positions for each atom $\mathbf{r}_i$ are updated every $k\Delta t$ while the velocities $\mathbf{v}_i$ are updated every $(k+\frac{1}{2})\Delta t$, leading to the staggered update rules:

$$\mathbf{v}(t+\frac{1}{2}\Delta t) = \mathbf{v}(t-\frac{1}{2}\Delta t) + \frac{\Delta t}{m}\mathbf{F}_i(t) \tag{1.3.4.1}$$

$$\mathbf{r}(t+\Delta t) = \mathbf{r}(t) + (\Delta t)v\left(t+\frac{\Delta t}{2}\right) \tag{1.3.4.2}$$

Interpolating the velocities over the half time-step allows for the greater accuracy with no added computational overhead.
While this method produces a suitable level of accuracy for many applications, many engines, such as NAMD, instead default to the popular third-order velocity-Verlet algorithms as the default[210]. This instead evaluates the velocities and positions at the same time-step using the update rule:

$$\mathbf{r}(t+\Delta t) = \mathbf{r}(t) + \Delta t\mathbf{v} + \frac{\Delta t^2}{2m}\mathbf{F}_i(t) \tag{1.3.4.3}$$

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t) + \frac{\Delta t}{2m}\Big(\mathbf{F}(t) + \mathbf{F}(t+\Delta t)\Big) \tag{1.3.4.4}$$

In addition to an integrator, a suitable time step for the simulation must be chosen. The usual computational step required for stability is of the order of femtoseconds. Writing each step to disk would not only create an inordinate amount of very similar data but quickly become a performance bottleneck, therefore it is common to only write frames after a predetermined number of steps. Determining these parameters is system dependent, corresponding to the timescales of the pro-

cesses studied and what simulation length is available given the size of the system. Current MD techniques have made the microsecond scale regularly achievable for systems consisting of up to tens of thousands of atoms[211]. This allows unbiased simulations to approach the time scales required to simulated complex processes such as protein folding, although thorough observations of these are only available on the best hardware and for smaller systems[212,213].

1.3.5 Boundary Conditions

Now that a general framework for the numerical approach to solving the dynamical equations is outlined, there are several physical considerations which still require further work before simulations can be performed. The most prominent issue not yet discussed is how to consider the boundary conditions of the simulation. This posits two clear problems, the first being how to deal with interactions at the boundary of the system, and the second, discussed more in Section (1.3.6), concerning how to deal with its development from a thermodynamic perspective. Early attempts at molecular simulation suffered from large non-physical forces at the edges of the box caused by the interaction of the system and the surrounding vacuum- colloquially known as "edge effects". The classical way to minimise these is by applying Periodic Boundary Conditions (PBCs) to the system[214].

Here the system considered is encapsulated within a unit cell and replicated a theoretically infinite amount of times in all 3-dimensions, simulating a crystal lattice type structure. The central cell is the only one for which the positions and velocities are updated, and during each force-calculation stage distances are measured to copies of this system placed across each face. The shape of the unit cell dictates its volume and hence how many water molecules will be required. This makes it of vital interest when simulating proteins, as for optimizing performance one wishes to minimize the number of explicit molecules considered. There are 5 shapes capable of forming a tessellation of 3D Euclidean space, known as the parallelohedra[215]. All of these can be represented using a set of 3 basis vectors $\hat{\mathbf{n}}$:

$$\hat{\mathbf{n}} := (n_a, n_b, n_c) \tag{1.3.5.1}$$

Contrary to a standard basis set, relaxing orthogonality constraints when specifying $\hat{\mathbf{n}}$ makes it possible to represent all parallelohedra using this coordinate system. When using PBCs, it is common to obey the Minimum Image Convention by truncating all non-bonded interactions above a maximum distance cut-off R_c which satisfies:

$$R_c < \frac{1}{2}\min(\|\mathbf{n_a}\|, \|\mathbf{n_b}\|, \|\mathbf{n_c}\|) \tag{1.3.5.2}$$

Ensuring the value of R_c obeys this criterion ensures only one copy of each atom is included during each force calculation step, meaning it will always have to consider at most $N-1$ other atoms. This in turn reduces the infinite tessellation of the periodic box, such that 26 separate copies of the system are required to pack 3D space directly around the central cell and ensure the boundary is sufficient.

While obeying Eqn. (1.3.5.2) and introducing a cut-off to long-range interactions is necessary for computational purposes, it introduces a systematic error when considering the non-bonded interactions considered in Eqn. (1.3.2.10) and Eqn. (1.3.2.9). This is mostly insignificant for the LJ potential which rapidly decays to zero, but for Coulombic interactions this can result in a significant deviation from expected results[216]. The theoretically infinite lattice combined with the long range of electrostatics provides a non-negligible contribution to U, and hence this requires specific treatment. A logical initial strategy is to simply extend the Coulombic term over the repeated crystal lattice:

$$\sum U_{\mathrm{EM}} = \frac{1}{2}\sum_{L}{}' \left[\sum_{i=1}^{N}\sum_{j=1}^{N} \frac{q_i q_j}{4\pi\epsilon_0(d_{ij} + L\hat{\mathbf{n}})}\right] \tag{1.3.5.3}$$

The outer sum is performed over all vectors L, which point towards the origin of every periodic cell considered. It is primed to indicate the term $i = j$ is ignored when considering the initial cell to avoid self-interaction, and the factor of one-half prevents double counting. While this equation is naive, it can be proved the sum is conditionally convergent depending on the order of the computation[217]. One of the conditions of the convergence of the potentially infinite outer series is that the net charge inside the cell must be 0- otherwise it will tend to infinity. This is one of the main justifications for introducing ions to the solvent discussed previously in Section (1.3.3).

Taking this equation towards another suitable cut-off where the interaction become negligible is a theoretically viable strategy, albeit one that is very computationally inefficient. To combat this, researchers looked towards similar methods which had found success in theoretical physics calculations, most notably the Ewald methods initially developed for the study of ionic crystals and calculation of the Madelung constant[218]. There are many full derivations available in several degrees of rigour, therefore here we shall summarise the key results and provide some intuition for one such derivation[219].

Ewald methods aim to calculate the electrostatic energy of a system by splitting the contributions into 2 terms:

$$U_{\text{Ewald}} = U^S + U^L \tag{1.3.5.4}$$

where U^S consists of the contributions from real space as considered in Eqn. (1.3.2.9) taken up to a chosen cut-off R_c, while U^L encompasses the long-distance interactions from beyond this radius. The Gauss error function $\text{erf}(x)$ and it's complement $\text{erfc}(x)$, share similar properties to what is desired, and can be used to model this interaction as an integral. They can be defined as:

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \tag{1.3.5.5}$$

$$\text{erfc}(x) = 1 - \text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_x^{\text{inf}} e^{-t^2} dt \tag{1.3.5.6}$$

Using these it is possible to represent any function dependent on the reciprocal of an arbitrary variable x as the sum of two terms with an appropriate choice of the constant β:

$$\frac{1}{x} = \frac{\text{erf}(\beta x)}{x} + \frac{\text{erfc}(\beta x)}{x} \tag{1.3.5.7}$$

The first term here is constant as $x \to 0$ but has a long tail as $x \to \infty$, while the second is asymptotically convergent as $x \to 0$ and vanishes as $x \to \infty$. These properties make the first term convergent in it's Fourier series while the second is convergent in this form in real space. Noticing this displays the exact properties needed to for a better form of Eqn. (1.3.5.3), this was applied to calculating the electrostatic field in a crystal lattice structure. Defining the reciprocal lattice

vectors to $\hat{\mathbf{n}}$ as $\hat{\mathbf{m}}$, these terms can be expressed using the structure factor $S(\hat{\mathbf{m}})$ standard in solid state physics literature. It is defined using a complex sum:

$$S(\hat{\mathbf{m}}) = \sum_{j=1}^{N} q_j \exp\{2\pi i \hat{\mathbf{m}} \cdot \mathbf{r}_j\} \tag{1.3.5.8}$$

Using this it is possible to express Eqn. (1.3.5.4) using three terms: the convergent short-range real space sum U_R, the convergent Fourier sum U_F, and a corrective term U_0:

$$U_{\text{Ewald}} = U_R + U_F + U_0 \tag{1.3.5.9}$$

$$U_R = \frac{1}{2} \sum_{L}^{\text{`}} \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{q_i q_j}{d_{ij} + L\hat{\mathbf{n}}} \text{erfc}(\beta(d_{ij} + L\hat{\mathbf{n}})) \tag{1.3.5.10}$$

$$U_F = \frac{1}{2\pi V} \sum_{\hat{\mathbf{m}} \neq 0} \frac{\exp\{-\pi^2 \hat{\mathbf{m}}^2 / \beta^2\}}{\hat{\mathbf{m}}^2} S(\hat{\mathbf{m}}) S(-\hat{\mathbf{m}}) \tag{1.3.5.11}$$

$$U_0 = -\frac{1}{2} \sum_{(i,j) \in X} \frac{1_i q_j}{d_{ij}} \text{erf}(\beta(d_{ij} + L\hat{\mathbf{n}})) - \frac{\beta}{\sqrt{\pi}} \sum_{i=1}^{N} q_i^2 \tag{1.3.5.12}$$

Where V is the volume of the unit cell. The corrective term U_0 cancels out interactions for pairs in the list X which accounts for interactions considered by the bonded terms of the FF, and are additionally excluded from the primed sum. It can be shown this energy is invariant with respect to the choice of β, which controls the rate of convergence of each term. In other formal derivations, this parameter is also related to the width of a chosen Gaussian distribution used to screen the charges.

Choosing an appropriate value for β then determines whether the Fourier space or real space term converges first without affecting the accuracy, therefore this can be decided based on computational reasoning. However this means a greater number of structure factors defined in Eqn. (1.3.5.8) are required. This led to several developments in the computations of Eqns. (1.3.5.4), and in practice the basic Ewald method is rarely used in favour of Particle Mesh Ewald (PME) methods[220]. These methods use splines, grids, or similar constructs to quickly evaluate many structure factors, allowing for a smaller cut-off to be used. Using an appropriate choice it is possible to reduce the computational cost from $\mathcal{O}(N^{\frac{3}{2}})$

to $\mathcal{O}(N \log N)$[221,222]. The nature of these calculations, relying on independent evaluations across grids using the same topology, make it a clear target for GPU acceleration[223,224]. This has been found great use across MD simulations, and has led to a significant boost in speed for large systems where the cut-off can now be made smaller and computational load on the CPU reduced. It was the first use for GPUs in molecular simulation to attain widespread adoption in the late 2000s, enabling fast electrostatics calculations and making microsecond MD simulations routine.

1.3.6 Ensembles

Now there is an overall strategy to solve Eqn. (1.3.1.4), the focus can turn to making simulations meaningful by enhancing their ability to replicate realistic systems. While the methods so far have discussed in detail how a system interacts internally, there has been no mention of how to deal with its thermodynamics properties. This poses no change to the physical theories used so far, but has grand implications to the overall behaviour of the system as it develops alongside the evaluation of physical quantities. Every model used in thermodynamics has a natural scope, after which the system under investigation ends and is taken to interact with a theoretical black-box describing the greater world outside. Simulating biomolecules is no exemption from this, and in order to corroborate with experimental results significant attention must be given to replicating physical conditions.

Any MD simulation can be performed with different controls applied to the system's development through time designed to emulate how it interacts with the outside world[225]. The set of controls used are referred to as the ensemble, as they are based on the concept of the thermodynamic ensembles the simulation aims to replicate. Ensembles may be referred to by an established name or by a sequence of letters signifying the quantities taken to remain constant during the simulation. The choice of an ensemble leads to a different expressions for system state variables at equilibrium, such as entropy and the partition function, and has drastic consequences on how a system develops through time.

The most basic ensemble used is the Microcanonical ensemble or NVE. Here the number of particles (N), the volume of the system (V), and the overall energy (E) are all conserved. This intuitively corresponds to if a simulation was performed using the theory discussed so far with no further consideration of thermodynamic properties, and hence all the initial energy calculated in U were to equalise between the potential and kinetic energy of the system. The key quantity of the ensemble is the number of microstates $\Omega(E)$ which may be used to express many related quantities. Under these constraints the probability of the different states is assumed to be equal, and as such the equation for the entropy reduces to the Boltzmann entropy formula:

$$S_{NVE} = k_B \ln \Omega(E) \tag{1.3.6.1}$$

The entropy is the main thermodynamics state function in this ensemble and provides the link between the statistics and thermodynamic properties. It also leads to simple form for the partition function:

$$Z_{NVE} = \frac{1}{\ln \Omega(E)} \tag{1.3.6.2}$$

While this is clearly computationally simple it is not often used for experiments, simulations, or theoretical applications, despite being the starting point of many introductions to thermodynamics. The value for $\Omega(E)$ can take an inordinate amount of time to evaluate with any statistical certainty, making it difficult to estimate any physical quantities from this ensemble. The ensemble leads to non-physical results as biomolecules naturally fluctuate in energy, and the process of mapping out the energy landscape is often the key reason for performing simulations in drug discovery tasks. A starting structure with a particularly large value for U can lead to very large solvent temperatures, while a particularly low value of U leads to potentially very low temperatures. With no way for energy to leave or enter the system these issues can lead to a poor sampling of biologically relevant computational space.

The Canonical ensemble, or NVT, preserves the temperature of the system T over time, making the state space explored now an isothermal surface $\Omega(T)$.

It finds widespread usage for both simulations where a specific temperature is required or during the pre-production steps to be outlined in Section (1.3.7). The partition function in this ensemble can be expressed in terms of a integral over state space of the potential energy U:

$$Z_{NVT} = \int_{\Omega(T)} \exp\left\{-\frac{U(\mathbf{r})}{k_B T}\right\} d^3r \tag{1.3.6.3}$$

where h is Planck's constant.
The key state function of this ensemble is the Helmholtz free energy[ii] F:

$$F = -k_B T \ln Z_{NVT} \tag{1.3.6.4}$$

In comparison to the NVE ensemble, the system now has the ability to transfer kinetic energy outside the system fixing the potential issues arising from energy dependence on the starting structure. The contribution to the partition function is now inversely related to the energy of a state, meaning higher energy contributions can be assumed to influence calculations less. This allows for the estimation of quantities with more accuracy in less time, circumventing the need to evaluate a large degree of state space.

To implement this ensemble the temperature must be controlled, and this is achieved using a variety of techniques known as thermostats. Approaches to thermostats can be described as strong, weak, extended-system or stochastic methods. All methods look to push the current system temperature T towards a target value T_0 with differing degrees of force[226]. Strongly coupled temperature scaling is the simplest, and looks to rescale the velocities v_i at each time-step or after a predetermined number of steps by a factor λ:

$$\mathbf{v}_i^{\text{rescaled}} = \mathbf{v}_i \cdot \lambda \tag{1.3.6.5}$$

Strong methods rescale or reassign the velocities to ensure the system stays at the correct temperature. This can be achieved by using the quotient between the

[ii] Often denoted using A instead in chemical literature.

current and target temperature:

$$\lambda = \sqrt{\frac{T_0}{T}} \tag{1.3.6.6}$$

Weakly coupled methods are closely related, but instead look to couple the temperature to an external heat bath of fixed target temperature. Therefore the natural heat flux between the two will be proportional to the temperature difference. The most well-known example is the Berendsen thermostat, which additionally introduces a scaling parameter τ[227]:

$$\lambda = \sqrt{1 + \frac{\Delta t}{\tau}\left(\frac{T_0}{T} - 1\right)} \tag{1.3.6.7}$$

By doing so this thermostat simulates the system being coupled to an external heat-bath at the specified temperature:

$$\frac{\partial T}{\partial t} = \frac{1}{\tau}(T_0 - T) \tag{1.3.6.8}$$

Neither of these are capable of correctly sampling the NVT ensemble and are not used for production quality MD runs. Instead they find widespread usage during equilibration stages to be described in Section (1.3.7) due to their ability to quickly change the temperature. Under these circumstances constraints are applied to the protein, so the aggressive nature of the temperature control can be mitigated.

Extended system methods encompass the widely used Nose-Hoover method and its many derivatives, where the general concept is to modify the defining equation of the dynamics Eqn. (1.3.1.1)[228]. The method achieves this by introducing a frictional term ζ_{ext} which is proportional to the velocity of the system:

$$\mathbf{F}_i = m_i \frac{d^2\mathbf{r}_i}{dt^2} + \zeta_{\text{ext}} m_i \dot{\mathbf{r}}_i \tag{1.3.6.9}$$

The coefficient ζ_{ext} changes through time and this dependence means it is able to decelerate or accelerate the system smoothly depending on the temperature. Nose-Hoover and it's extensions find wide-usage as they are capable of correctly sampling

both the NVT and NPT ensemble. As it fundamentally changes the equations of motion it requires some significant changes to the system and techniques used, requiring direct changes to the integrators used.

The final class of thermostats involve the usage of random numbers to control the temperature[229]. There are many widely used stochastic methods, such as the Langevin thermostat used throughout this work. This thermostat can be viewed as a hybrid stochastic and extended system method, introducing both a stochastic parameter ζ_{LD} and a random process $X(t)$ to Eqn. (1.3.1.1):

$$\mathbf{F}_i = m_i \frac{d^2\mathbf{r}_i}{dt^2} - \zeta_{\mathrm{LD}} m_i \dot{\mathbf{r}}_i + \sqrt{2 m_i \zeta_{\mathrm{LD}} k_B T} \frac{dX(t)}{dt} \tag{1.3.6.10}$$

The dynamics are governed by both these parameters. The damping constant ζ_{LD} defines the inertia of the system, and in the high-ζ_{LD} limit these equations become equivalent to Brownian motion and a protein's innate dynamics become somewhat negligible. To avoid this the constant must be carefully chosen, however it is important to not choose too small values which could lead to the system heating. The random process $X(t)$ is commonly a time-invariant Gaussian process, satisfying the conditions:

$$\langle X(t) \rangle = 0 \tag{1.3.6.11}$$

$$\langle X(t) \cdot X(t') \rangle = \delta(t - t') \tag{1.3.6.12}$$

where $\delta(\cdot)$ is the Dirac delta function and angled brackets denote the expectation value. Again using such a thermostat requires a change to the integrator, and there are several efficient implementations capable of doing so with good performance. The Langevin method is an effective way to control the temperature, but does so by inevitably dampening some of the dynamics which take longer than $1/\zeta_{\mathrm{LD}}$. While these thermostats are capable of correctly sampling an ensemble, this damping can disturb correlated movements and slow down some kinetic processes[230]. For drug discovery uses where the proteins large-scale dynamics are the most important this may not necessarily be a negative, however it should be noted if short timescale processes like water dynamics are of critical importance.

While the first 2 ensembles considered are of theoretical importance, it is the Isothermal-Isobaric or NPT ensemble which finds the most usage in modern simulations and throughout this work[231]. In modern MD this is considered the most desirable ensemble, and has become the de facto ensemble used due to its physical accuracy[232]. It is the most flexible of those considered as it is able to exchange thermal energy using the aforementioned thermostat algorithms, alongside altering the unit-cell vectors defined in Eqn. (1.3.5.1) in order to maintain pressure. This is the most similar to experimental conditions, and as aligning with experiments is the main goal in drug discovery tasks all production simulations in this work are performed using this. The partition function can be related to that of the NVT ensemble defined in Eqn. (1.3.6.3) through introducing an additional integral over the now varying volume V:

$$Z_{NPT} = \int_0^\infty \exp\left\{-\frac{pV}{k_B T}\right\} Z_{NVT} dV \tag{1.3.6.13}$$

The key state function is now the Gibbs free energy:

$$G = -k_B T \ln Z_{NPT} \tag{1.3.6.14}$$

In addition to specifying a thermostat algorithm, simulating in this ensemble requires specifying a pressure-altering algorithm known as a barostat[233]. Barostats maintain a specified pressure P_0 by fluctuating the volume of the box. The pressure of a simulation can be estimated using the Virial equation:

$$P = \frac{N k_B T}{V} + \frac{1}{3V} \left\langle \sum_{ij} r_{ij} F_{ij} \right\rangle \tag{1.3.6.15}$$

This can be efficiently calculated after the force calculation step.
The simplest way to couple again is to link the volume to a scaling constant λ as done with the temperature in Eqn. (1.3.6.5):

$$P^{\text{rescaled}} = P \cdot \lambda \tag{1.3.6.16}$$

Strongly coupling the pressure leads to hard rescaling of the simulation box which causes significant unwanted changes to the dynamics. Therefore they find little

usage, and instead the simplest forms found are weakly coupled methods. The Berendsen barostat is very similar to its thermostat counterpart specified in Eqns. (1.3.6.7) and (1.3.6.8), in that it couples the pressure to another system with a specified pressure[234]:

$$\frac{\partial P}{\partial t} = \frac{1}{\tau}(P_0 - P) \tag{1.3.6.17}$$

Leading to a rescaling constant equivalent to:

$$\lambda = \left(1 - \frac{\Delta t}{\tau}\left(P_0 - P\right)\right) \tag{1.3.6.18}$$

Much like the thermostat equivalent, this does not correctly sample the ensemble but is widely used for equilibration steps where quick scaling is required. There is also room for vector valued τ which can cause the scaling to be anisotropic.
Similarly there are many extended system barostat methods that find widespread usage and correctly sample the ensemble. They are fundamentally based on the same principles as the temperature coupling methods by extending the equations of motions using a fictitious mass which can instead be interpreted as a piston compressing the system. These include the Parrinello-Rahman barostat, the Nosé-Hoover barostat, and the Martyna-Tuckerman-Tobias-Klein barostat[235–237]. Such methods introduce additional degrees of freedom for the box vectors $\hat{\mathbf{n}}$ to develop over time depending on the current pressure calculated from Eqn. (1.3.6.15). As these fluctuations affect the whole simulation, they can be best viewed as modifying Eqn. (1.3.1.5) to $\mathcal{H}'$. For example when using the Parinello-Rahman barostat:

$$\mathcal{H}'(\mathbf{p}, \mathbf{q}, t) = \mathcal{H}(\mathbf{p}, \mathbf{q}, t) + \sum_i P_{ii} V + \sum_{i,j} \frac{1}{2} W_{ij} \left(\frac{d\hat{n}_{ij}}{dt}\right)^2 \tag{1.3.6.19}$$

where $\mathbf{W}$ is a matrix defining the strength of the coupling between box vectors, often referred to as the barostat mass, and the pressure has been decomposed across each dimension.
In this scheme, the box vectors obey the equation of motions defined by:

$$\left(\frac{d\hat{\mathbf{n}}}{dt}\right)^2 = V W^{-1} \hat{\mathbf{n}}^{-T} (P - P_0) \tag{1.3.6.20}$$

where $\cdot^{-T}$ represents the matrix inverse transpose. Extended system barostats are widely used, but have significant difficulty in situations deviating far from equilibrium. Stochastic barostats provide a more robust option, based on the successes of similar thermostats as described in Eqn. (1.3.6.10)[238,239]. These operate very similarly to the temperature coupling methods, however the Gaussian process is now applied to the box volume:

$$\frac{d^2V}{dt^2} = \frac{1}{\mathbf{W}}\left(P - P_0\right) - \zeta_{\mathrm{LB}}\frac{dV}{dt} + \sqrt{2m_i\zeta_{\mathrm{LB}}k_BT}\frac{dX(t)}{dt} \tag{1.3.6.21}$$

Again the pressure-coupling friction coefficient ζ_{LB} must be adequately chosen. The combination of Langevin pressure and temperature coupling has consistently risen in popularity over the past decade, and is the combination used throughout in this work.

1.3.7 Minimisation and Equilibration

Before beginning a simulation there are two more physical considerations. The first is to note that the structure may not be in a starting conformation capable to effectively sample its realistic dynamics. Often programs used to solvate the periodic box generate the water positions with no consideration of the solute position, leading to waters having unnatural locations. Additionally there is no way to determine how much an experimentally found structure has been impacted by the crystallization procedure or affected by errors introduced when processing the electron density maps. In many scenarios when beginning MD from an experimental structure, early visualisation of the results can make it clear that the system is failing to find a natural energy well of the FF. This can be confirmed by investigating the calculated forces in the system- as finding particularly large values points toward the presence of an unfavourable and potentially non-physical interactions. To resolve issues such as these it is possible to perform a period of energy minimization before commencing the full simulation, where gradient strategies similar to those posited to optimise the parameters of NNs in Section (1.2) are used to reduce unfavourable interactions.

These algorithms are often very simple and work by optimising all positions at step n $\mathbf{r}^n$ with respect to the total force vector $\mathbf{F}$. As this step looks to reduce

unfavourable interactions and not perform any significant alteration, methods such as this often look to iterate the positions until the magnitude of the proposed next step is below a specific tolerance ϵ or a predetermined number of steps has been performed. For example, the default algorithm in the MD engine GROMACS is based on a general steepest descent paradigm[240]:

$$\mathbf{r}^n = \mathbf{r}^{n-1} + \frac{\mathbf{F}^n}{\max(|\mathbf{F}^n|)} \cdot h^n \tag{1.3.7.1}$$

Here h^n is the constant analogous to the learning rate, which can too be optimised over time.

The second, and final consideration, is how to initialise the velocities and ensure the simulation is performed close to thermodynamic equilibrium. Although experimental structures provide positions for atoms, this has no indication of how they may oscillate at a chosen temperature T. Considering the initial minimised structure as physical conformation, care now needs to be taken when beginning the dynamics. Failing to do so, or doing so too aggressively on a sensitive system, could result in poor sampling of biologically relevant conformational space. There are several different strategies, and is it common to do it as a staggered procedure where the temperature gradually increases over several stages while protein atoms are subjected to physical constraints. This initially severely limits the initial movement of the protein as the water molecules heat up, and the slow process of reducing these constraints ensures kinetic energy is distributed realistically over the solute. Initial velocities can be chosen from the Maxwell-Boltzmann probability distribution:

$$p(\mathbf{v}_i) = \left(\frac{m_i}{2\pi k_B T}\right)^{\frac{1}{2}} \exp\left\{\left(-\frac{m_i \mathbf{v}_i^2}{2k_B T}\right\}\right) \tag{1.3.7.2}$$

As T iterates over several stages, the velocities are reassigned until the desired final temperature is reached. Both minimization and equilibration are commonly performed in the NVT ensemble.

Chapter 2

Target Identification & Investigation

2.1 Introduction

Any drug design process is a long endeavour, and requires the work of a variety of different researchers across a spectrum of fields. The first stage is identifying and characterising the biological targets involved in a disease whose modulation would yield therapeutic benefit. This requires an in-depth knowledge of both disease pathology and systems biology[241–245]. Building and analysing datasets to this aim is the focus of the first part of this thesis, focused on identifying protein targets and performing further investigations into their dynamics and function. Here researchers look to determine the key drivers of the diseases processes, to better understand what underlying biological activities cause the observed symptoms.

Once this is understood, the next step is to pinpoint which areas are viable candidates for therapeutic intervention. This involves an in-depth understanding of the disease, leveraging approaches from areas such as genomics, proteomics, and bioinformatics, to build a network of biological activities causally linked to a pathological condition. Not all involved processes are suitable for modulation, with many being vital to other critical functions. Once a suitable target has been identified, work can begin on accruing data to investigate its structure, function, mechanism, and dynamics, with the downstream goal of verifying whether it has the potential to be modified by binding a small drug-like molecule. Or to put it succinctly, is "druggable"[246].

After this stage is the entry point to drug-discovery for this work. Exploring techniques to identify and collect useful data about protein targets which already known to be disease modifying or critical for pathological development. This task can be split into two main threads, focusing on either locating relevant structural data or generating and parsing dynamical data. The first, coming under the identification topic, involves working with sequence and structural databases in order to pinpoint what information is available on a target or how much relevant information can be extrapolated from studies on closely related proteins. Often a useful starting point for this type of work is determining and ranking relevant information by using alignment tools to locate similar sequence and structural motifs. The second thread is the investigation side, where the task shifts to generating and parsing large datasets of structural and dynamical information for various uses. These include finding structures key to the protein function or particularly suitable for downstream modelling tasks such as docking, elucidating the dynamic mechanisms to aid future drug design work, or simply parsing the high-dimensional data into more intuitive formats.

After introducing the relevant common methodology for these tasks in Section (2.2), the focus moves to studying 3 specific case studies showcasing how these techniques can be used or developed upon to identify and investigate early-stage biological targets. Section (2.3) looks at BRAF, a target with a mutated form highly prevalent in breast cancer cases. This provides a good example of using alignment programs to collect relevant data across publicly available biological databases, and highlights the benefit ML tools can bring to identify potential dynamical pathways key to function. In Section (2.4) turns to elucidating the functional mechanism of NSP13, the SARS-CoV-19 helicase critical to virus replication. This begins with a basic dynamical analysis of several MD runs across different force-fields, displaying the critical role of analysis for identifying behaviours that differ or coincide across methods. It then moves to the early stages of a full functional simulation, with the aim of observing the translocation process suing MD to provide insight into how this may be modulated. Finally, Section (2.5) introduces a new methodology for interpreting dynamical data as Markov processes. It develops a new methodology for using GNNs as an optimization

framework to cluster these faithfully by preserving a key dynamical quantity known as Kemeny's constant.

2.2 Methodology

2.2.1 Target Identification

Given a viable protein target, the first stage of research is to identify and collect relevant data which may be useful to the study. This involves searching the considerable amount of publicly available biological data which is open for researchers to use. Aggregators such as UniProt and GenBank provide easy access to gene sequences and their CDS translations, while the PDB provides the central repository for experimentally derived structural data[247,248]. As the rate of generation of biological data has been steadily increasing, the scale of these repositories is now beyond the capabilities of many standard search methods. Effectively navigating them provides a great challenge, and requires both specialised techniques and substantial computational resources.

Alignment algorithms provide one possible approach to mining useful information from these databases by finding similar proteins using similarity metrics to quantify how related they may be, or their homology[249–251]. Most aim to produce a 1-to-1 sequence alignment from 2 or more proteins, identifying whether individual residues or sections are similar evolutionary, functionally, or structurally. They also commonly provide a single numerical interpretation of this value, useful for then ranking the homology over a large dataset. There are a plethora available methods which require different data to use for comparison and look to assess based on different criteria, and this work focuses on two distinct approaches using either primary sequence or structural data. Performing these alignments across vast datasets enables researchers to collate and rank data which may be useful to the further study of a potential target. This provides the basis for a large proportion of the preliminary work in drug design, such as in Case Study 1 where these datasets form the core dataset used to characterise the conformational space of a conserved protein loop.

Sequence Alignment

The primary sequences of proteins is the language of cellular activity. While it defines its function, structural properties, and evolutionary history, decoding this relationship is one of the major ongoing struggles in science and still proves difficult with many competing approaches. Sequences are widely distributed in the simple text-based FASTA shown in Fig. (2.1). This format consists of a single sequence or a collection of primary sequences, with identifiers specified on lines beginning with a >-character followed by the residue sequence using one-letter abbreviations. The simplicity allows for quick and easy storage, distribution, and parsing of protein sequences and both it and its extensions are in widespread usage across proteomics and genomics research.

```
>SEQUENCE 1
AVGACVLCNSQTSLRCGACIRRPFLCCKCCYDHVISTSHKL...
```

Figure 2.1: An example of FASTA format file.

While this has now become the de-facto primary sequence file format, its origins are as the input format to the homonymous sequence alignment program[252,253]. Alignment of primary protein and DNA sequences has been a topic of great research since the field of genomics first emerged, as it became quickly evident effective comparison was beyond human capability. The fundamental problem of determining the correspondence between two sequences has then become establishing a computable similarity measure between any two sequences. Such a problem is highly non-trivial leading to most approaches being heuristic, and the original FASTA algorithm is just one approach. Early attempts split between aligning entire sequences or focusing on discovering conserved small motifs, commonly referred to as global or local alignments respectively[254]. In general, global alignments can yield a measure for evolutionary homology between full proteins, while local alignments can be used to find conserved functional regions such as compound-specific pockets.

The most impactful early approaches used expensive dynamic programming approaches, most notably the Needleman-Wunshch and Smith-Waterman algorithms[255,256]. Although they provided effective results and still find usage today, they struggled

to scale to modern database sizes, but current developments using GPU accelerated calculations have shown potential to heavily increase performance[257,258]. These algorithms lay the foundations for the majority of approaches that are in use today, establishing the core concept of using a pairwise comparison between two sequences as the basis for a scoring metric. These dynamic programming approaches compare two sequences at a time using a substitution matrix, which provides a numerical value to each potential pair.
Given two protein sequences A and B:

$$A = a_1a_2a_3 \dots a_n \quad B = b_1b_2b_3 \dots b_m \tag{2.2.1.1}$$

A score between each individual element is assigned using the score matrix $\mathbf{S} = s(a_i, b_j) = S_{ij}$, where the indices i and j cover all the unique characters in any sequence- usually the proteinogenic Amino-acids (AAs). The elements of the matrix S_{ij} define the value added to the alignment score if a pairing of elements (i, j) is considered. The potential variability in $\mathbf{S}$ allows for many different systems to emerge, and the wide variability for different uses was discussed even from the first paper introducing the Needleman-Wunsch algorithm in 1970. As a simple example, consider the basic scoring system:

$$s(a_i, b_j) = \begin{cases} 0 \text{ if } a_i = b_j \\ 1 \text{ if } a_i \neq b_j \end{cases} \tag{2.2.1.2}$$

This is equivalent to the Levenshtein metric, and would be appropriate when considering how homologous a collection of closely related sequences are. Using this metric S takes the form of a matrix where all values are 1 except the leading diagonal which is zero, and the smaller this value the more similar the sequences are. In practice scoring matrices for protein alignment are derived by applying statistics to datasets of known conserved protein sequences.

Such an example commonly used for protein sequences are the BLOSUM matrices[256]. These were constructed using a set of over 2000 conserved protein regions consisting of the 20 most common AAs. For each pair of AAs observed in each region of the the dataset, an observed probability q_{ij} was calculated from

the frequency f_{ij} using:

$$q_{ij} = \frac{f_{ij}}{\sum_{i=1}^{20} \sum_{j=1}^{i} f_{ij}} \tag{2.2.1.3}$$

From this, the expected probability of occurrence for each residue in a pair p_i can be calculated:

$$p_i = q_{ii} = \sum_{j \neq i} \frac{q_{ij}}{2} \tag{2.2.1.4}$$

Similarly, this defines the expected probability of observing each pair e_{ij} as:

$$e_{ij} = \begin{cases} p_i p_j \text{ if } i = j \\ 2 p_i p_j \text{ if } i \neq j \end{cases} \tag{2.2.1.5}$$

The ratio of observed probability to expected probability can be viewed as a measure of how much the observed frequencies deviate from what would be expected. A BLOSUM matrix is defined as the logarithm of these values multiplied by a scaling factor of 2:

$$s_{ij} = 2 \log_2 \left(\frac{q_{ij}}{e_{ij}} \right) \tag{2.2.1.6}$$

There are several different BLOSUM matrices used depending on the level of homology a search requires. Sequences above a certain identity threshold can be clustered by weighting all included pairs such that they contribute the same as a single sequence. The sequence identity cut-off r considered defines the BLOSUMr matrix. This weighting procedure means as r increases, more sequences with a smaller evolutionary distance are considered.

Smaller cut-off values tend to be used for assessing more dissimilar proteins, for example providing an effective score when looking for comparisons between protein families. Larger values, giving more weight to similar sequences, are beneficial for assessing closely related proteins, for example studying the variation within a protein family. An example of the commonly used BLOSUM62 matrix is shown in Fig. (2.2). As a general rule, identities and replacements which conserve properties of the protein are scored positively, while unlikely substitutes are scored negatively. This results in a statistically significant measure between two protein sequences which is able to quantify the degree of homology between them. The other most widely used scoring matrix is the PAM family, which were some of

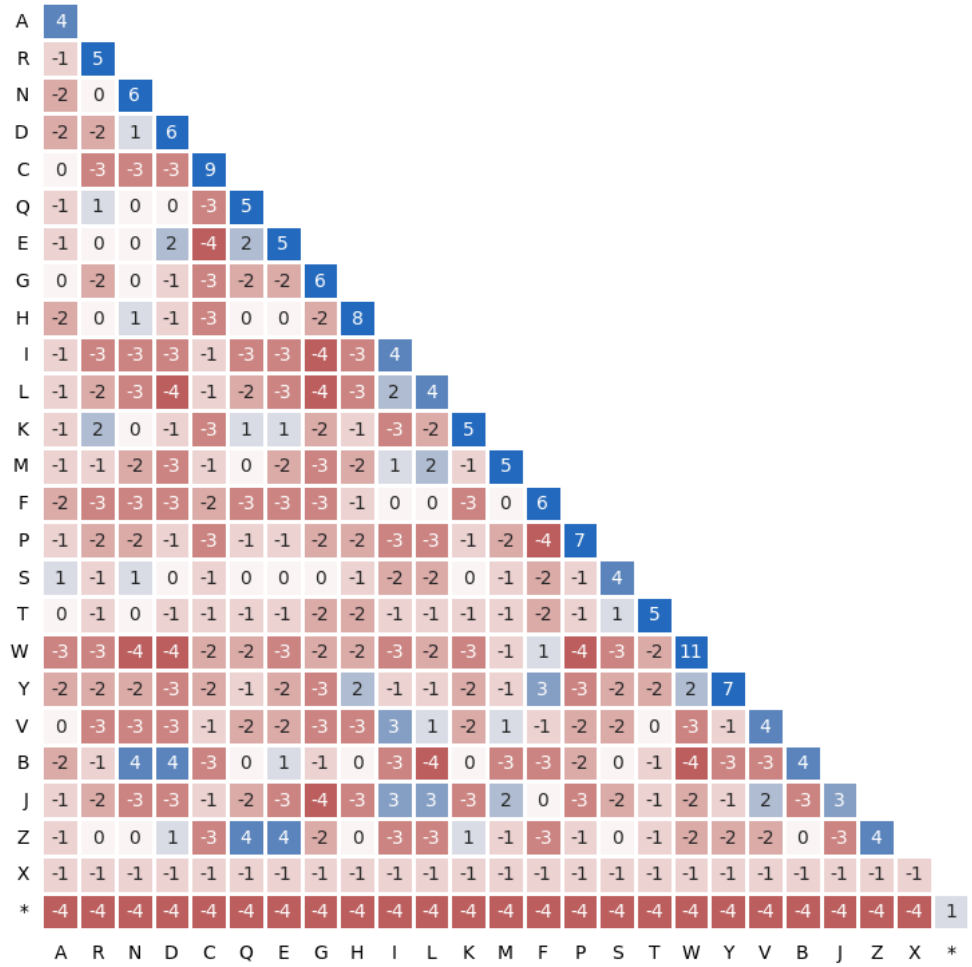


Figure 2.2: The values of the BLOSUM62 matrix.

the first scoring matrices introduced[259,260]. They are constructed in a similar statistical fashion, instead deriving values from the frequencies of point mutations across 1572 proteins from 71 different families. Choosing the correct matrix is search dependent, and it has been shown to have a dramatic impact on search results.

As indels and point-mutations are common in biology, these alignment algorithms were also built to accommodate for possible gaps in the alignment. This is commonly implemented using an affine scheme:

$$G_k = lk + o \tag{2.2.1.7}$$

Where G_k is the score of introducing a gap of length l, parametrised by a penalty for each unaligned element k and a base penalty for introducing a gap o. This is commonly written as a tuple (o, k).

Both algorithms use a score matrix and a gap penalty to calculate a score for all possible individual pairings or gaps. Once this is stored, it is recursively traced through to determine the best possible alignment score. The main difference between determining a local or global alignment, is the global begins from the end of the sequence and traces back to the beginning, whereas the local begins from the maximum scoring pair and works backwards until the score is unchanging. By considering all alignments these algorithms guarantee the best possible alignment is found. However using a dynamic programming strategy quickly becomes very costly in both memory and time, especially when covering potentially millions of sequences. For these reasons, these algorithms are often used only on small databases or when it is absolutely necessary to know the best possible alignment, and other related algorithms have become more commonplace for high-throughput searching tasks.

In place of heavy-duty alignment algorithms, k-mer based methods have emerged as the dominant strategy for performing large-scale local alignment searches across large protein sequence databases[261]. These methods prioritise computational efficiency over accuracy, instead looking to provide a quick reference value useful for determining if there is any potential for similarity between two sequences. This can be used to efficiently determine which sequences are unrelated and may be disregarded, which is most likely a majority of the queried database, providing a quick and easy method to screen sequence data before using more accurate methods if needed. Their unifying strategy is to instead base alignments initially of short identical matches between the sequences of length k, which can then be extended using a scoring strategy to give a final score. By not evaluating every possible alignment, these methods are significantly faster than their predecessors at the cost of reliability and accuracy. There are many such examples of these programs, however most rely on estimating some simplification of the above dynamic programming methods at their core.

BLAST

Of these methods, one of the most popular is the Basic Local Alignment Search Tool (BLAST), first published in 1990[262]. The method is used across both protein and nucleotide sequences, and has many available local implementations and an online servers for easy usage[263,264]. The BLAST method produces a set of local AA alignments between a single given sequence known as the query, and a usually very large set of sequences known as subjects. For a given word length k, a default of $k = 3$ for protein searches, the method splits the query sequence into individual sequences $\omega = \{\omega_1, \ldots, \omega_k\}$ using a sliding window approach. For a query sequence of length m, this generates $m - k + 1$ words, each of which are then scored by assuming an identical match, calculating using the score matrix:

$$\text{score}(\omega) = \sum_{i=1}^{k} s(\omega_i, \omega_i) \tag{2.2.1.8}$$

The BLAST algorithm then utilises two threshold values T_w and T_s. The first is the word pair threshold, and only words scoring above this as calculated by Eqn. (2.2.1.8) are considered as seeds, or hits, to begin the alignment process. The default value is $T_w = 13$, and each subject sequence is searched for all identical words scoring above this. When a matching word is found, the algorithm then begins extending the match in both directions until it's score stops increasing or decreases 20 below the highest value already found. This ensures matches with a very low probability (≈ 0.001) of scoring as highly are terminated early. Once all matches between the query and a subject sequence are found and extended the algorithm terminates.

The second cut-off parameter T_S, is the minimal score required for an alignment between two proteins to be considered significant. Only alignments scoring above this threshold will be considered, and it is an important usage specific parameter as higher values are more useful when considered more homogeneous sequences. This is often expressed as a probability known as the Expect-Value (e-value), which gives a statistical estimate of how likely this match is to be by chance. In this form it is calculated using two parameters, λ and K, which were derived in early work on the statistics of sequence distributions. Beginning with a score S,

the first step is calculating the normalized score S', known as the bit-score, which enables a statistical comparison between scores using different systems:

$$S' = \frac{\lambda S - \ln K}{\ln 2} \tag{2.2.1.9}$$

The number of times this can be expected to appear is dependent on the database size. Taking $N = mn$, where N is the length in residues of all subject sequences, the e-value can then be expressed using:

$$E = \frac{N}{2^{S'}} \tag{2.2.1.10}$$

Inverting this equation allows one to define a score as the function of an e-value:

$$S' = \log_2 \left(\frac{N}{E}\right) \tag{2.2.1.11}$$

This means a cut-off T_S can be defined using either a numerical value in a specific system, or as a probabilistic condition. The latter is advantageous for several reasons, as it allows for direct comparison between scoring systems and is considered more intuitive as a probability based scheme. This is the default in BLAST, with notable matches being those with $E < 0.05$.

While the initial BLAST algorithm focused on finding the maximum scoring alignment segment between two sequences, it was later extended to include gaps alongside several performance enhancements. This utilises an affine scheme as introduced in Eqn. (2.2.1.7), with default parameters $(11, 1)$. To calculate the gapped alignment, this uses a variation on the dynamical programming method limited to the local section between two high scoring aligned regions. This version of BLAST has found widespread acceptance in the bioinformatics community, and is available both locally and via a web-service increasing it's accessibility. In both cases, the default parameters specified above can be fine-tuned for usage-specific needs, and the original paper and later studies provide plenty of empirical examples of what may be suitable for a given task.

Structural Alignment

Primary sequences provide a good starting point for assessing the homology between different proteins, however they can miss some of the more nuanced relationships between less homologous systems. One of the key tenets across structural biology is the correspondence between structure and function. Structural alignment algorithms aim to exploit this by prioritizing the structural similarities between proteins. Protein structures can be found distributed on the PDB, providing experimental 3D-point data for any atoms resolved using experimental techniques.

Databases such as SCOP, CATH, and FSSP, provide a useful entry point to PDB data[265–267]. To provide an assessment across diverse proteins, they cluster proteins into similar families and superfamilies. On primary sequence data, this is often performed using probabilistic interpretations of the sequence data using hidden Markov models, a particular case of the Markov chains worked on in Case Study 3. However it has been shown that proteins with wildly differing sequences can exhibit significant structural and functional similarity[268]. Therefore these classification methods also rely heavily on structural similarity methods, which aim to quantify the structural homology between proteins.

Such alignment procedures take protein structures in a PDB or similar format, and again look to produce a 1-to-1 correspondence of residues instead based on structural criteria. For comparison between protein models the Root-Mean Square Deviation (RMSD) is often used as the simplest similarity metric, finding applications across structural biology. Given two structures, both with n atoms represented using $3D$-coordinate vectors $A = (\mathbf{a}_1, \ldots, \mathbf{a}_n)$ and $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$, the RMSD is defined as:

$$\mathrm{RMSD}(A, B) = \sqrt{\frac{1}{N}\sum_{i=1}^{n}(\mathbf{a}_i - \mathbf{b}_i)^2} \tag{2.2.1.12}$$

While this is incredibly useful when comparing structures using the same models, for example over the course of a MD-run, it is incapable of directly comparing structures with a differing number of atoms. This is a severe limitation when comparing different proteins. Furthermore, the lack of a canonical basis between

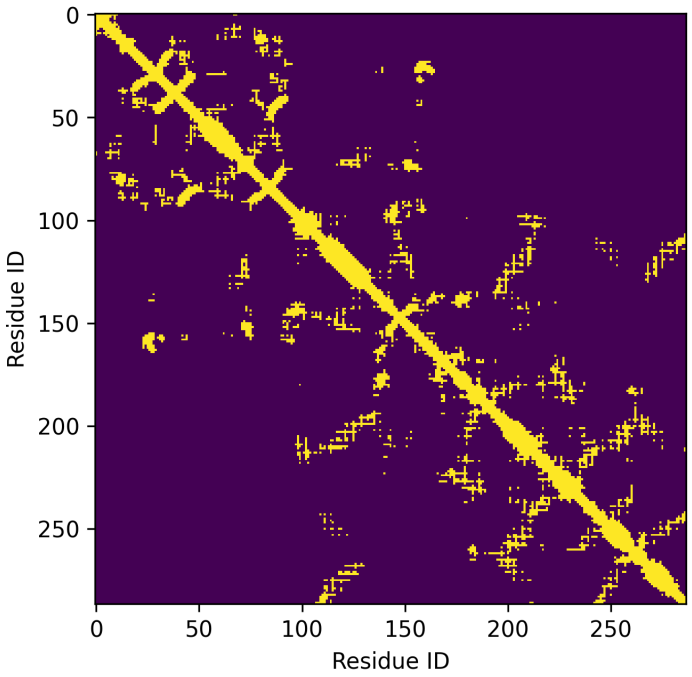


Figure 2.3: Example of a discretized protein contact map generated from MD simulations of BRAF performed for Case Study 1. Each value at index (i, j) takes a value of 1 (yellow) if the closest non-hydrogen atoms between residue i and residue j are separated by a distance of $\mathring{4}$ or less, and 0 otherwise (purple).

experiments makes direct comparison between Euclidean coordinates ineffective, as there is no guarantee they will be modelled in a similar space or orientation. There are two main approaches to bypass this issue.

Firstly, it is possible to remove any reliance on a basis set at all by converting to an internal coordinate scheme. In this, an N-atom proteins atomic coordinates are converted from a $(3, N)$-shape array to a square (N, N)-matrix representing all inter-residue distances present. The distances are most commonly measured using only the C_α atoms for each residue, the closest non-hydrogen atoms, or all atoms. This allows structures to be represented visually in a form known as a contact map, and an example is shown in Fig. (2.3). These representations have found particular usage across structural biology due to its utility in converting 3D representations into coordinate invariant 2D images, providing particular success in early protein structure prediction attempts[125,269]. The task of generating these is a simple Euclidean distance calculation, which is ample for GPU acceleration, and moving to $2D$ figures allows for the applications of CNNs such as in the original AlphaFold

implementation. For structural alignment tasks, they provide a way to avoid any coordinate dependence while enabling comparison between structures of various sizes, and are the basis of several methods such as the widely used DALI and its extensions[270]. DALI-based methods segment two protein contact maps into smaller matrices representing similar local substructures, and pair the associated elements within these matrices. The partitioning aims to maximise the additive similarity score S_{DALI} for each segmented region:

$$S_{\mathrm{DALI}} = \sum \sum_{i=1}^{K} \sum_{j=1}^{K} \phi(i,j) \tag{2.2.1.13}$$

Where the indices i and j enumerate the K matched residue pairs contributing to the alignment, and ϕ is some similarity measure between the paired residues where a larger value denotes a more similar structure. This score has no upper-bound. The structures A and B are converted into distance matrices composed of elements d_{ij}^{A} and d_{ij}^{B}, the inter-atomic distance between atom i and j. The original implementation introduced two explicit forms of ϕ, the simplistic rigid similarity score:

$$\phi^{R}(i,j) = \theta^{R} - |d_{ij}^{A} - d_{ij}^{B}| \tag{2.2.1.14}$$

where θ^{R} is some scalar, $\theta^{R} = 1.5$Å in the original implementation, which denotes a zero level of similarity. Distances below this indicate a good pairing, and thus contribute to building the score, whereas values above this denote a poorer fit and reduce the overall score. It also introduced the elastic similarity score ϕ^{E}:

$$\phi^{E}(i,j) = \begin{cases} \left(\theta^{E} - \frac{|d_{ij}^{A} - d_{ij}^{B}|}{d^{A+B}ij}\right) \exp\left\{\frac{-d_{ij}^{A+B}}{\alpha^{2}}\right\} \; i \neq j \\ \theta^{E}, \; i = j \end{cases} \tag{2.2.1.15}$$

with $d^{A+B}ij = \frac{1}{2}(d_{ij}^{A} - d_{ij}^{B})$ being the average distance between both contact maps. The exponential term in the first branch is an envelope function designed to give less weight to paired residues which are far apart, and the default value is $\alpha = 20$, empirically observed to be the the approximate size of an average domain. The underlying concept is that structurally similar domains should have similar contact map patterns, and therefore by optimising pairings using

Eqn. (2.2.1.13) it is possible to find the most geometrically similar alignment. Initialising the method requires a starting segmentation scheme along with an optimisation method. Originally this used a basic 5-AA segmentation scheme which merged areas with very similar contact maps, along with a Monte-Carlo scheme to propose improvements, although this was later expanded to include more efficient schemes[271,272]. This method is the basis of the FSSP database.

Another approach to structure comparison is to treat the protein as a rigid point-cloud and utilise techniques from geometry to find the transform which minimises Eqn. (2.2.1.12). The atomic positions of two crystallised structures consisting of N-atoms can be represented using the $(N, 3)$ vectors $\mathbf{x} = \{x_{ij}\}$ and $\mathbf{x}' = \{x'_{ij}\}$, where $i = \{1, \ldots, N\}$ indexes the atoms and $j = \{1, 2, 3\}$ indexes the dimensions of $\mathcal{R}^3$. The translation which best minimises the RMSD corresponds simply to translating each point-cloud such that its centroid corresponds with the origin:

$$x_i = x_i - \frac{1}{N}\sum_{i=0}^{N} \tag{2.2.1.16}$$

In what follows this initial step is assumed to be performed for both $\mathbf{x}$ and $\mathbf{x}'$.

Finding the subsequent rotation which minimises Eqn. (2.2.1.12) is a more complex procedure, and several approaches have been used including the widely used Kabsch algorithm[273]. The most robust, stable, and compact way to implement finding this transform is by using quaternions, the extension of the complex numbers to $3D$-space[274]. An arbitrary unit quanternion can be written as $Q = (q_1, q_2\mathbf{i}, q_3\mathbf{j}, q_4\mathbf{k}) = (q_1, \mathbf{q})$ where $\mathbf{i}, \mathbf{j}, \mathbf{k}$ are the canonical basis vectors of Euclidean space. The composition rule for two quaternions Q and P is:

$$QP = (q_1 p_1 - \mathbf{q} \cdot \mathbf{p},\, q_1\mathbf{p} + p_1\mathbf{q} + \mathbf{q} \times \mathbf{p}) \tag{2.2.1.17}$$

The cross product means this is non-commutative.
Any $3D$-rotation of a vector $\mathbf{v}$ can subsequently be expressed using the conjugate $Q^{-1}\mathbf{v}Q$ of a quaternion, which ensures it results in a real quaternion. Amongst their other uses, the quaternions of unit norm provide a representation of $SU(2)$ and therefore a double-cover for the Euclidean rotation group $SO(3)$, which can be viewed as an effective parameterisation of any Euler rotation with an added

degeneracy of $Q^{-1}\mathbf{v}Q = (-Q)^{-1}\mathbf{v}(-Q)$. Using this latter view, they can be viewed as a Euler rotation with a vector representing an axis $\mathbf{a}$ and an angle θ about which the points are rotated about:

$$\mathbf{a} = (a_1, a_2, a_3) = \frac{(q_2, q_3, q_4)}{\sqrt{q_2^2 + q_3^2 + q_4^2}} \tag{2.2.1.18}$$

$$\theta = 2 \arccos q_1 \tag{2.2.1.19}$$

This allows them to represent rotations effectively while remaining numerically stable and avoiding common problems, such as gimbal lock and improper rotations, which are commonly found when using other techniques[275]. The optimal transform Q, rotating $\mathbf{x}$ onto $\mathbf{x}'$ results in the residual real quaternion $E = (0, \mathbf{e})$:

$$E = (0, \mathbf{x}') - Q^{-1}(0, \mathbf{x})Q \tag{2.2.1.20}$$

Contact can be made with Eqn. (2.2.1.12) by first left multiplying through by Q, then squaring, summing:

$$Q(0, \mathbf{e}) = Q(0, \mathbf{x}') - QQ^{-1}(0, \mathbf{x})Q = Q(0, \mathbf{x}') - (0, \mathbf{x})Q$$
$$\therefore Q^2 \sum_{i=0}^{N} \mathbf{e}_i^2 = \sum_{ij} \left[Q(0, x'_{ij}) - (0, x_{ij})Q \right]$$

This can be further simplified by introducing $\Delta^m = (0, \mathbf{x}' - \mathbf{x})$, $\Delta^p = (0, \mathbf{x}' + \mathbf{x})$, and multiplying out using Eqn. (2.2.1.17). Alongside this, introducing a Lagrange multiplier to restrict solutions to the unit quaternions, results in the final expression for E^2:

$$E^2 = \sum_{i=0}^{N} (Q\Delta_i)^2 + \lambda(1 - q_1^2 + q_2^2 + q_3^2 + q_4^2) \tag{2.2.1.21}$$

By using this to express the products, then differentiating with respect to the components of Q and setting to zero, the problem may be expressed as an eigenvalue problem of a symmetric matrix:

$$\mathbf{B} = \begin{pmatrix} \sum (\Delta_{i1}^m)^2 + (\Delta_{i2}^m)^2 + (\Delta_{i3}^m)^2 & \sum \Delta_{i2}^p \Delta_{i3}^m - \Delta_{i2}^m \Delta_{i3}^p & \sum \Delta_{i1}^m \Delta_{i3}^p - \Delta_{i1}^p \Delta_{i3}^m & \sum \Delta_{i1}^p \Delta_{i2}^m - \Delta_{i1}^m \Delta_{i2}^p \\ \vdots & \sum (\Delta_{i1}^m)^2 + (\Delta_{i2}^p)^2 + (\Delta_{i3}^p)^2 & \sum \Delta_{i1}^m \Delta_{i2}^m - \Delta_{i1}^p \Delta_{i2}^p & \sum \Delta_{i1}^m \Delta_{i3}^m - \Delta_{i1}^p \Delta_{i3}^p \\ \vdots & \vdots & \sum (\Delta_{i1}^p)^2 + (\Delta_{i2}^m)^2 + (\Delta_{i3}^p)^2 & \sum \Delta_{i2}^m \Delta_{i3}^m - \Delta_{i2}^p \Delta_{i3}^p \\ \vdots & \vdots & \vdots & \sum (\Delta_{i1}^p)^2 + (\Delta_{i2}^p)^2 + (\Delta_{i3}^m)^2 \end{pmatrix}$$

$$\mathbf{B}\begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q4 \end{pmatrix} = \lambda \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q4 \end{pmatrix} \tag{2.2.1.22}$$

Where all the summations are implicitly over i, omitted for clarity.
This technique is used across different software libraries, and is a basic technique implemented in visualisation programs, MD software, and small molecule similarity techniques[276–278]. Over the course of any MD simulation, there are significant translational and rotational movements which must be removed before any comparisons can be made. Using Eqn. (2.2.1.22) assures researchers the structures aligned are at a minimal superposition. Its reliance on parallel linear algebra techniques have made this a key target for GPU techniques, which have sped this up 260-fold and made it viable to perform across whole genomes[279].

However an unresolved issue persists in how to deal with the reliance on N. When dealing with structures of differing size, how best to choose the atoms to compare is paramount to the success of the method and is implementation dependent, as there is no clear best strategy. Many techniques use a sequence alignment to first match N similar residues, giving a pairing between C_α atoms. Others rely on more complex methods, and the following section shall discuss how one in particular circumvents this issue.

MUSTANG

The MUSTANG algorithm is one approach to generating a structural alignment across a database of many different structures which utilises a mixture of the aforementioned approaches[280]. The algorithm works on the C_α coordinates of N separate structures consisting of $\mathbf{L} = \{L_1, \ldots, L_N\}$ residues to produce an alignment represented by a $(N \times l)$-matrix $\mathcal{A}$. Each element of $\mathcal{A} = \{a_{ij}\}$ is indexed by a structure label $i = \{1, \ldots, N\}$ and a corresponding residue index $j = \{1, \ldots, l\}$. The index l ranges from $\max(\mathbf{L}) \leq l \leq \sum \mathbf{L}$, as each element can be either a residue identifier from the alphabet of proteinogenic AAs or a gap

represented by a hyphen −. It produces a Multiple Sequence Alignment (MSA) which is useful for assessing the homogeneity across the whole database.

Contrary to DALI based methods, here contiguous regions of proteins are detected via RMSD minimising transform before a similarity score is assessed using the closeness of the paired contact map pattern. For each pair of proteins in the database, the algorithm begins by determining the largest section of the proteins which are able to be aligned with an RMSD below a predefined threshold ϵ. To do so, it assesses all segments of κ-residues able to be formed beginning from the N-terminus. For every section the RMSD-minimisation transform is then calculated using a scheme based on on Eqn. (2.2.1.22), and if the found value is below ϵ the algorithm attempts to extend the region through adding further consecutive residues to the C-terminus. This process is repeated for all residues $1 \leq j \leq L_i - \kappa + 1$ until the largest regions, or maximal fragment pairs, are found for all pairs of structures. The algorithm truncates every maximal fragment pair which ends between a helix or a beta strand towards the start of the secondary structure, to avoid empirically observed mismatches in this region. It does so by predicting the secondary structure from the C_α positions[281]. This is the only part of the algorithm which uses any secondary structure prediction or sequence data. For this part, the algorithm looks to align regions that are below $\epsilon = 1.75$Å in terms of RMSD and at least $\kappa = 6$ residues long.

For each MSP scores are then tentatively assessed using a contact map comparison based off the DALI elastic score shown in Eqn. (2.2.1.15). The end strategy of the algorithm is to optimise this alignment using dynamic programming to produce a MSA alignment across the database which best optimises this score. Therefore once this initial pairing is found, it is theoretically possible to proceed directly to the final step. However similarly to the dynamic approaches in sequence alignment, the time cost of this approach scales very poorly- with the squares of the sequence lengths being compared. This quickly becomes prohibitively expensive, and as such the algorithm uses several strategies to first assess equivalence classes between closely aligned regions before using the dynamic approach. Further implementation details of these can be found in the original paper.

The MUSTANG algorithm therefore provides a useful way to assess the homogeneity across a database of proteins. The produced MSA allows users to view

regions of a protein database which are more likely to be structurally related, helping to pick up on relationships that may be missed by sequence-based alignment algorithms. It is also able to be run on individual pairs, giving a structural assessment in a similar format to BLAST by quantifying how homologous all proteins in a database are to a reference structure. By basing the assessment on purely structural information it is able to pick up on mutations that may be functionally irrelevant yet stable.

2.2.2 Target Investigation

The data gathered during the target identification stage outlined in Section (2.2.1) is useful for determining the starting point for an investigation but is often insufficient. While it can provide a significant number of sequences and structures, potentially enough to identify a target's function, family, and some conformational diversity, the reality is that more data or further analysis is needed to begin downstream tasks. This data may not be publicly available, and the analysis may require a system specific methodology.

In cases where further conformational data is required, the attained information will have to be supplemented with simulated data to asses whether it is suitable for pharmaceutical intervention. Determining this will require an in-depth understanding of the process which is to be modulated, for example a critical conformational change, catalytic reaction, formation of a binding pocket, or a quaternary structure formation. Whatever form this process takes, its dynamics and energy are now the principal property of interest. This is a area with a significant number of approaches, and no single definitive methodology. Even to experts in the field the large dimensionality and magnitude of data proves beyond an intuitive comprehension, and as such researchers are always looking for new ways to generate, process, or display biological data. For the data-orientated in this field this provides an opportunity to perform large analyses across gigantic datasets, and consists of some of the most varied and numerical work in the field.

The most common way of investigating targets is by performing MD simulations outlined in Section (1.3). Fundamentally, these simulation methods probe a systems dynamical properties through solving it's equations of motions. Under a more state-based statistical mechanics based view, they can also be seen as producing time-dependent samples of a systems Free-energy Surface (FES)[282]. The value of a proteins internal energy ε_i associated with a particular conformation i, is intrinsically linked to the probability of that state occurring p_i and the partition function Z introduced in Section (1.3:

$$p_i \propto \frac{1}{Z}\left(\exp\left\{-\frac{\varepsilon_i}{kT}\right\}\right) \tag{2.2.2.1}$$

In all but the simplest circumstances this equation is impossible to evaluate, especially on systems as complex and large as biomolecules. Yet the abstract geometry this defines is arguably the main object of interest when studying dynamical systems such as proteins. Catalytic reactions and important conformational changes, key targets for treatment, can be viewed as paths along this surface.

Transition state theory provides the underlying framework that high-energy barriers separate reaction pathways into reactant and product states[283]. Interpreting this for proteins implies the abstract FES of a biomolecule dictates its likely conformations alongside their catalytic potential. Proteins reside in stable low wells for long periods and rarely transition to higher energy states or through large barriers, which often separate the functionally interesting states researchers look to observe. While this provides a useful abstract viewpoint, utilising it suffers from two main issues.

The unbiased simulations that have so far been discussed suffer heavily from residing in relatively uninteresting wells. Most biological phenomena, especially for larger systems, occur at timescales that are infeasible to simulate even with the benefit of modern GPU hardware[212]. Enhanced sampling methods have emerged to combat this fundamental issue, and have achieved a large level of success in providing statistically rigorous methods capable of observing rare phenomena in biomolecules[284–288]. In this work, only a simple implementation of biasing is used in Case Study 2, and so in the interest of relevance it shall be introduced there. However it is important to note that these methods form an important part of the modern study of biomolecules, and no overview of the subject would be complete without an exploration of their uses and benefit.

The second main issue, and the focus of this section, is forming an effective understanding of the geometry of the FES. This is a difficult endeavour due to the high-dimensionality of all biomolecular systems, making it infeasible to gain any intuition of its structure. It is often unclear from observation what is hindering a protein from performing a functional transition, or even what such a transition looks like. Achieving an understanding of these processes can require significant system specific expertise. This section looks into general dimensionality reduction methods used to interpret the datasets gathered during the target investigation stage. It looks at both classical and modern ML-based approaches, showcasing

how researchers are able to compress large datasets into easier to understand formats.

Dimensionality Reduction

Dimensionality reduction is a key facet of interpreting the results of any biomolecular simulation. It is the task of projecting a dynamical system down to its most important quantities, and even before this term was commonplace across general science it was well understood as an integral task in the area[289,290]. In modern ML parlance, this is often referred to as feature engineering and requires expert-level domain knowledge. Chemical reactions occur with a number of degrees of freedom impossible to visualise or efficiently sample from, therefore for interpretation or further investigation the dimensionality must be reduced. Its significance in the field can be traced back to the early chemical notion of Reaction Coordinates Reaction Coordinates (RCs)[291]. These can be identified as projections η from the conformational space of a system down to a one-dimensional space capable of separating the reactants and products of a chemical reaction:

$$\mathrm{RC} = \eta(\mathbf{x}) \in \mathcal{R} \tag{2.2.2.2}$$

Examples include key distances, volumes, or angles, often requiring some level of chemical knowledge about the reaction considered. In their current usage, they are not specific to just chemical reactions and can instead be used to separate interesting protein conformations, such as active or inactive states.
With the advent of more complex reactions and biomolecular simulations, it was often seen one coordinate was not enough to fully describe many processes. This resulted in the concept of a Collective Variable (CV), which simply serves as a contraction of a set of RCs into a singular function ζ[292]:

$$\mathrm{CV} = \zeta\{\mathrm{RC}_i(\mathbf{x})\} \in \mathcal{R}^n, \quad i = \{0, \ldots, n\} \tag{2.2.2.3}$$

Throughout the history of this area, finding effective CVs and RCs has been considered a difficult task requiring significant system-specific expertise. These were decided via analysis of theorized chemical reactions or catalytic functions,

and this is still a viable approach, finding usage in the previous work Case Study 5 looks to build upon. An ineffective choice of a CV leads to a poor estimate of the FES, leading to bad estimates of any free energy estimations or process rates. Therefore identifying relevant functions is of great importance to any downstream uses of simulation data.

Moving forwards, there is a shift towards automated approaches capable of parsing large datasets and finding the most interesting or relevant kinetic quantities. This can be seen as a response to the scale of current drug discovery tasks, where relevant kinetic quantities and target potential must be assessed as quickly as possible. Classically, there are several approaches which have been adopted by the field. Simple correlation based analysis can be performed at a large scale to highlight quantities which may be heavily related. The Pearson Correlation Coefficient (PCC) is a popular method for calculating such relationships between two scalar sequences x and y[293]:

$$\mathrm{PCC}(x,y) = \frac{\sum_i (x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\sum_i (x_i - \mu_x)^2 \sum_i (y_i - \mu_y)^2}} \tag{2.2.2.4}$$

where $\mu_.$ denotes the mean.
It produces a simple value bounded between $[-1, 1]$, with lower values implying an inverse proportionality and higher values a direct relationship. The standardization in Eqn. (2.2.2.4) means this is unaffected by the overall magnitude of the two sequences, allowing for comparison between two dramatically different values. Performing large scale PCC calculations provides an easy method to numerically identify potentially important relationships within data. Pairwise calculations can be performed either across sets of distances, or between distances and important calculated quantities such as pocket volumes, ligand interactions, or ion coordination. While the method is computationally very simple, in can be scaled even to the largest databases via modern GPU implementations[294].

Dynamic cross-correlation (DCC) map analysis built upon this to provide a way to generalise correlations to vector valued sequences[295–297]. This allows it to be applied to raw atomic coordinates, providing a method to evaluate the extent to which domains move together. It is commonly represented as a matrix,

calculated using:

$$C_{ij} = \frac{\langle \Delta r_i(t) \cdot \Delta r_j(t) \rangle_t}{\sqrt{\langle |\Delta r_i(t)|^2 \rangle_t} \sqrt{\langle |\Delta r_i(t)|^2 \rangle_t}} \tag{2.2.2.5}$$

where $r_i(t)$ is the i^{th} atoms coordinates at time t, $\langle x \rangle_t$ denotes the time ensemble average of the quantity x, and $\Delta r_i(t) = r_i(t) - \langle r_i(t) \rangle_t$. This equation again yields a scalar quantity C_{ij} for each pair in the range 1 to -1. The closer the value to 1, the more the displacement of atom i is correlated to that of j. Similarly, a negative value indicates an inverse correlation between the two displacements, and a zero value indicates there is no correlation. The dot product in the numerator means unlike PCC values this value is dependent upon the respective orientation of the two displacements. This allows for identification of distinct regions of movement in protein simulations, such as the opening and closing of pockets.

Moving beyond correlation based methods, Principal Component Analysis (PCA) is the next most common classical strategy used for general dimensionality reduction[298,299]. The method relies on calculating the cross-correlation matrix from a $(N \times M)$ data matrix $\mathbf{X}$, interpreted as N measurements of M different features. It is common to first remove the mean from each column of $\mathbf{X}$ and optionally divide by the variance, to weight features regardless of their magnitude. In application to MD or structural protein data consisting of S samples of P atoms, the $(S, P, 3)$ positions of the atoms of a protein can be reshaped to form a $(S, 3P)$ shaped data matrix. Due to memory requirements, this can be performed just for C_α atoms.

PCA looks to find the linear combinations of the M columns which represent the greatest proportion of the observed variance across the N samples[300]. Most historical derivations of PCA rely on calculating either the first projection, explaining the most variance, explicitly and subsequent projections from this, or using the covariance matrix derived from $\mathbf{X}$. Modern implementations instead identify these combinations with the singular value decomposition of the data matrix[301]:

$$\mathbf{X} = \mathbf{P}\boldsymbol{\Delta}\mathbf{Q}^T \tag{2.2.2.6}$$

The $(M \times \min(N, M))$ matrix $\mathbf{Q}$ here is known as the projection or PC matrix. Its values are the linear coefficients of the combinations of features which best explain

the observed variance. The columns are projection operators which when applied to **XQ**, yield the values of the projections for each sample and are ordered in decreasing order of the amount of variance they represent. **PΔ** gives the amount of variance explained by each combination, or PCA scores.

In its full form, PCA is a simple change of basis on the data space into a coordinate basis maximising the variance. It is important to note that this relies upon the assumption that the dimensions with the greatest variance best explain the dynamical process being investigated, and this is not guaranteed or often the case. For true dimensionality reduction, it is commonplace to take a subset of the rows of **Q** which explain the greatest degree of variance. Doing so allows researchers to ignore potentially irrelevant or small motions, such as the vibrational modes of hydrogens or uncorrelated loop motions. This allows PCA to function as a kinetic screening step, highlighting just the major correlated modes. Determining the number of coordinates to take is often done by visually comparing the explained variance using a scree plot, providing a simple and intuitive description of the explained variance per PC. The explained variance can be calculated using:

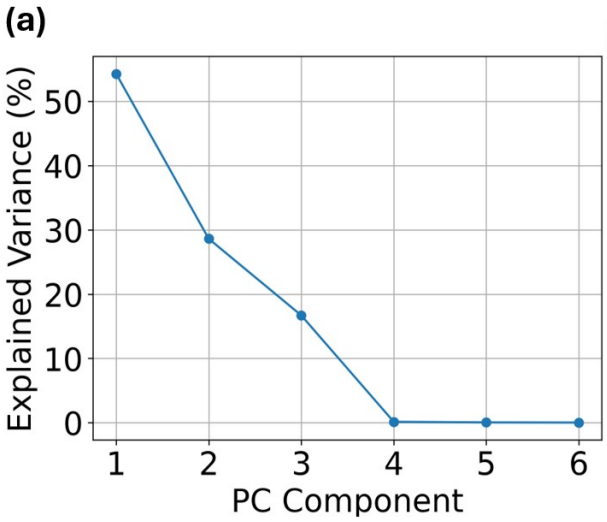
$$\lambda_i = \sum_{k}^{M} (\mathbf{P\Delta})_{ki}^2 \tag{2.2.2.7}$$

where λ_i is equivalent to the eigenvalues of the associated covariance matrix obtained in the alternate derivation.
An example of a scree plot is shown in Fig. (2.4). Once a necessary number of PCs are decided upon, their projections of the original data can be visualised to observe whether there is any interesting separation that can provide an avenue for further investigation.

Additionally the contribution of the i^{th} feature to the j^{th} PC, Y_{ij}, can be calculated via:

$$Y_{ij} = \frac{(\mathbf{P\Delta})_{ij}^2}{\lambda_i} \tag{2.2.2.8}$$

Comparison between contributions for a single PC can subsequently be used to highlight how much an individual feature contributes to a component. This can be visualised to potentially identify the key features driving any motion. When



(b)

PC Component	Explained Variance (%)
1	54.26
2	28.61
3	16.71
4	0.14
5	0.05
6	0.03

Figure 2.4: An example of a PCA generated from MD data on the BRAF system studied in Case Study 1. Prior to calculation all frames are aligned using RMSD-minimisation, and only the C_α coordinates are used. ***(a)*** *shows a scree plot displaying the percentage of the overall variance explained by each component. The first 3 components in this example account for over 99% of the observed variance. The kink in the plot showing a large discrepancy between consecutive PCs is commonly observed in these plots, and gives a good indication of how many PCs to use.* ***(b)*** *is a table showing the numerical values for each component.*

performed on structural data, instead the weighted PCA-RMSD modes Z_{ij} can be used in a similar fashion to obtain information about an individual residues contribution to a PC[302]. Doing so requires reshaping the PC matrix $\mathbf{Q}$ such that each column is now the sum of all features corresponding to a single residue $\mathbf{Q}^*$. For the i^{th} residue, its contribution is then the product of the eigenvalue and the per-residue PC matrix:

$$Z_{ij} = \sqrt{\lambda_i Q^*_{ij}} \tag{2.2.2.9}$$

This can again be compared for each component, and the multiplication by the square root of eigenvalue means this is in units of distance allowing for a clear structural interpretation.

While PCA has been widely applied to biomolecular simulation studies, it is limited to only modelling linear relationships. This can severely hinder the investigation of large and complex reactions or conformational changes, and as such pushed researchers to look to novel ML approaches capable of non-linear

modelling. Autoencoders are a class of latent space ML models which look to fill this gap, and have found a large amount of success in biological simulations recent years[303–305]. Latent space refers to the abstract vector space they are trained to form, capable of representing complex structures using a lower dimensionality. This is achieved via using a reconstruction loss function such as the Mean-Squared Error (MSE):

$$\mathcal{L}_{\text{RECON}} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x_i}) \tag{2.2.2.10}$$

where $\hat{x_i}$ is the models output given structure x_i.
This intuitively assess model performance as the accuracy with which it can map the data down to a lower dimensional representation, and reconstruct it. The form a latent space takes is dependent upon the model architecture, which generally enforces some kind of information bottleneck through using a lower dimensionality than the data space.

Autoencoders are one of the earliest examples of latent space models, forming a p-dimensional vector space $\mathcal{Z} \cong \mathcal{R}^p$ using an encoder-decoder architecture[306]. The simplest implementation uses an encoder network $E_{\theta_1} : \mathcal{X} \to \mathcal{Z}$ formed of a sequence of linear layers to produce an embedding $z \in \mathcal{Z}$. This embedding can then be processed by a similar decoder network $D_{\theta_2} : \mathcal{Z} \to \hat{\mathcal{X}} \cong \mathcal{X}$, to yield the final model output $\hat{\mathbf{X}}_i$. The reconstruction loss then becomes:

$$\mathcal{L}_{\text{RECON}} = \frac{1}{N}\sum_{i=1}^{N}[x_i - D_{\theta_2}(E_{\theta_1}(x_i))] \tag{2.2.2.11}$$

Both models are independently parametrised with different weights but trained concurrently, with gradients calculated from the final reconstruction configuring both weights. Using the MSE yields a model designed to both generate an effective description of the data and reconstruct it. After training, both models provide many subsequent practical applications. Not only can the encoder be used as a similar dimensionality reduction method like PCA to provide a potentially informative separation of high dimensional data, but the decoder can be used to interpolate between datasets and to generate new data points. From this perspective such models can be seen as a form of lossy compression, and when performed without non-linearities these models have been shown to span the

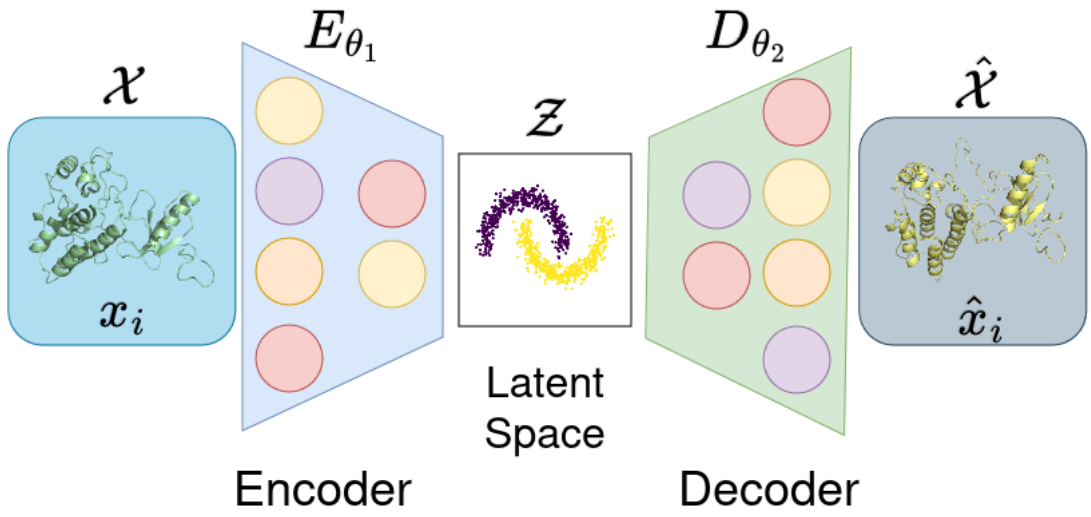


Figure 2.5: Graphical representation of an AE architecture being used for dimensionality reduction of a protein simulation. The model flows from left to right, and is trained on the MSE between original data point x_i and its reconstruction $\hat{x}_i$.

same space as PCA[307]. Similarly to PCA, the compression process is normally agnostic to the downstream task considered, although it is clear how Eqn. (2.2.2.11) can be augmented with additional differentiable terms for a specific usage. The architecture of a generic AE is illustrated graphically in Fig. (2.5).

The AE and similar models have a long history in ML, but early implementations had a number of issues with their practical implementations. Most prominent were problems with any lack of support in the data, leading to decoder interpolations to be inaccurate even when extrapolating between relatively similar data points. The non-linear nature of the mapping meant the learnt space $\mathcal{Z}$ could compress a significant amount of the configurational space between two supported points, leading to discontinuities. This also meant the model could learn to simply replicate the training data points, disregarding the structure of the underlying data manifold. To combat this Variational Autoencoders (VAEs) were developed, first introduced by Kingma & Welling in 2013[308].

While they share a common nomenclature, the AE and VAE are related in application and usage only, as the theory between the models is quite distinct. Rather than use the direct mapping from data space to latent space seen in AEs trained with MSE loss, VAEs extend these methods capabilities by looking to

model the underlying data distribution. In this approach. the data comprising a training set is thought to be produced by some unobserved process dependent on the latent variable z_i. Data generation then occurs as a two step process, where z_i is generated via the prior distribution $p_\phi(z_i)$ followed by the data being generated through the conditional distribution $p_\phi(x_i|z_i)$. This can be expressed using Bayes theorem:

$$p_\phi(z_i|x_i) = \frac{p_\phi(x_i|z_i)p(z_i)}{p_\phi(x_i)} \tag{2.2.2.12}$$

$$p_\phi(x_i) = \int p_\phi(x_i|z_i)p(z_i)dz_i \tag{2.2.2.13}$$

Evaluation of $p_\phi(x_i)$ is commonly intractable due to the integral over all of the latent space, leading to dimensionality issues even when attempting to solve this via Monte-Carlo schemes. This integral also appears when using maximum likelihood approaches, leading to researchers to find alternate approaches to using probability distributions for AE modelling.

The solution came from the field of variational inference, by introducing a model distribution $q(z) \in \mathcal{Q}$, where $\mathcal{Q}$ is a known family of distributions allowing for a simpler expression of $p_\phi(x_i)$[309]. The encoder now looks to estimate an approximate posterior distribution $E_{\theta_1} = q_{\theta_1}(z_i|x_i) \approx p_\phi(z_i|x_i)$, and the decoder the latent distribution $D_{\theta_2} = p_{\theta_2}(x_i|z_i)$. A critical result from work in pure statistics is that using such a formulation places a strict lower bound on the likelihood of observing the given data, known as the Evidence Lower Bound (ELBO):

$$\log p_\phi(x_i) \geq -\mathrm{KL}(q_{\theta_1}(z_i|x_i)||p_\phi(z_i)) + E_{q_{\theta_1}(z_i|x_i)}[\log p_{\theta_2}(x_i|z_i)] \tag{2.2.2.14}$$

By maximising this term, it is possible to obtain the best possible approximation of $p_\phi(x_i)$ without requiring an explicit formulation.
The second term in the ELBO is deterministic, defined for any sampled latent variable z_i. Therefore it can be approximated easily by Monte-Carlo sampling the output of $q_{\theta_1}(z_i|x_i)$. The first term is the Kullback-Leibler (KL) divergence,

defined for two probability distributions H and O as:

$$\mathrm{KL}(H||O) = \sum_i H(x_i) \log\left(\frac{H(x)}{O(x)}\right) \tag{2.2.2.15}$$

This is a key concept forming the basis of many variational methods, and full derivations of it are widely available but not covered here for brevity[310].
In this context, it gives a viable optimisation criteria for probabilistic VAE methods. As it is a lower bound requiring maximisation, in practice it is possible to multiply this by -1 to obtain a loss function for use with autodiff. All that is needed to evaluate Eqn. (2.2.2.14) is to decide upon the distribution used as the latent prior $p_\phi(z)$ and the family of approximate posteriors $\mathcal{Q}$.

As always, Gaussian distributions for both are considered sufficient and find widespread usage near universally in practical applications of VAEs. This is due to the clear advantages they bring to evaluating Eqn. (2.2.2.14). Firstly, the second term in the ELBO becomes directly proportional to the reconstruction loss from Eqn. (2.2.2.11). Assuming a dataset is large enough to provide a decent Monte-Carlo approximation to the decoder expectation value, this previous form can now be again used to directly optimise the reconstruction part of the VAE.
The prior is commonly modelled as a unit Gaussian with a mean of 0 and standard deviation of 1:

$$p_\phi(z) = \mathcal{N}(0,1) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \tag{2.2.2.16}$$

Additionally, the family of distributions $\mathcal{Q}$ for modelling the posteriors is generally the multivariate Gaussians with diagonal variance terms:

$$\mathcal{Q} := \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma^2}) = \frac{1}{\sqrt{2\pi\boldsymbol{\sigma}^2}} \exp\left(-\frac{(x-\boldsymbol{\mu})^2}{2\boldsymbol{\sigma}^2}\right) \tag{2.2.2.17}$$

Substituting these into the KL divergence defined in Eqn. (2.2.2.15) results in the heavily simplified form:

$$\mathrm{KL}(q_{\theta_1}(z_i|x_i)||p_\phi(z_i)) = \frac{1}{2}\left(1 + \log\left(\boldsymbol{\sigma^2}\right) - \sigma^2 - \mu\right) \tag{2.2.2.18}$$

The last consideration required before training a VAE is how to differentiate through the sampling of $q_{\theta_1}(z_i|x_i)$, as this stochastic process can introduce a

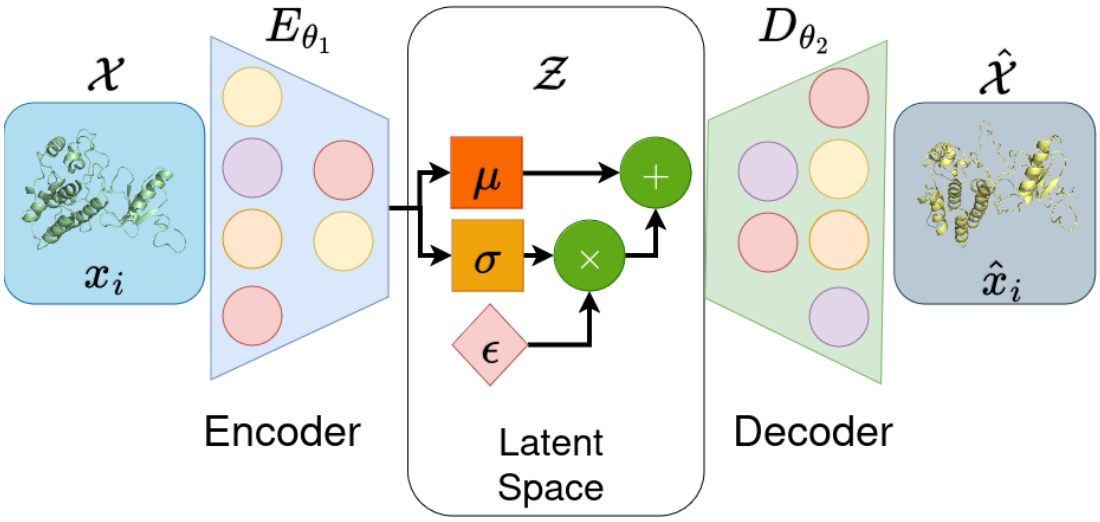


Figure 2.6: Graphical representation of an VAE architecture being used for dimensionality reduction of a protein simulation. In the centre, the autodiff schematic for the reparametrisation trick is shown. The encoder network now produces two p-dimensional vectors representing the mean μ and standard deviation σ for a given data point x_i. This is used to modulate the p-dimensional samples from a multivariate Gaussian ϵ using differentiable unit-wise multiplication and addition operations, bypassing calculating explicit gradients for any non-deterministic sampling functions.

large variance to any gradient estimator used. This is perhaps the most notable contribution of the initial paper introducing the method, and it provides a formal derivation of the reparametrization trick used[308]. However with the notation for autodiff introduced in Section (1.2), this procedure becomes quite intuitive. The general idea is to use the encoder to instead output two p-dimensional vectors representing the mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$ of q. Most commonly this is achieved by using two separate final linear layers with distinct weights. Instead of attempting to backpropagate through the sampling procedure directly, independent samples ϵ are drawn from a unit normal distribution which are then multiplied by the variance and translated by the mean. Utilising this procedure allows for a deterministic distribution to be sampled from allowing for the usage of autodiff, while capitalising on the capabilities of stochastic modelling. A graphical explanation is provided in Fig. (2.6). As the training procedure now encompasses a stochastic modelling process, a test set is commonly used during training to evaluate model generalisation.

VAEs have found widespread success in deriving non-linear CVs in the biological simulation community[311–313]. However it is important to note using such models loses a significant degree of interpretability, and cannot give the same degree of understanding as an expertly derived coordinate. Still it is currently the most promising technique for furthering the degree of structural investigation that can be automated. Many modern applications look to effectively combine these methods with enhanced sampling in order to quickly identify probable conformational changes. VAEs can also provide analysis for complex non-linear dynamics where PCA may fail to recover the most important variations.

2.3 Case Study 1: Exploring the Conformational Diversity of the Activation Segment of BRAF

2.3.1 Overview

The first case study of this work showcases how a large-scale analysis of available biological structures can be leveraged to answer questions concerning the functional dynamics of a protein. It is a clear example of the critical role target identification and investigation methodologies play in the preliminary stages of drug design tasks, and how there is still more to learn even after treatments are well established. It highlights the importance of effectively identifying relevant biological data, parsing dynamical and structural information, and how ML methodologies can improve researchers' ability to extrapolate key information from such datasets.
This project focuses on BRAF, a member of the RAF family of protein kinases which play a vital role in cell signalling. It is a relatively well understood protein which has received significant academic attention due to its prominent oncogenic mutations. Several therapeutic compounds have been identified and developed to the point of widespread usage, and BRAF is hailed as a major breakthrough in rational drug design. Despite this there are still many questions about the system, concerning paradoxical activation, suppression, and its dynamics. Due to its proximity to critical mutated residues, the activation segment, a relatively conserved kinase motif, stands out as having a clear functional significance. This region plays a key role in both the wild-type and mutated function of the protein, and a better understanding of its behaviour could play a key role in furthering our understanding and developing future treatments. The aim of this work was to characterise the conformational space available to the activation segment by studying all available crystallised structures of both it and similar proteins. This could then be used to derive a feasible kinetic pathway for a functional conformational change in the system from an inactive to an active state. Although the final pathway was never fully realised and enhanced simulations were not performed, the project stands as an example of the key work involved at this stage of biological study in the drug design pipeline.

2.3.2 Background

A signalling cascade begins when biological messenger molecules bind to receptors on the cell membrane, and these can range from peptide growth factors, cytokines, or hormones[314]. The most common kind are Mitogen-Activated Protein Kinase (MAPK) pathways, comprised of 3 to 5 layers of MAPK proteins which can either interact independently or be organised into large signalling complexes by scaffold proteins[315,316]. Signals are passed downstream in a cascade by the involved proteins sequentially phosphorylating each other[317]. This is a reversible post-translational modification of a protein where a phosphate group is cleaved from an Mg-ATP molecule and a covalent bond is made to a specific amino acid, generally a serine or threonine. This changes the proteins current conformation, in turn modifying its catalytic capabilities. Chemically the reaction is:

$$\mathrm{MgATP^{-1}} + \mathrm{R\text{-}OH} \rightarrow \mathrm{R\text{-}OPO_3^{2-}} + \mathrm{MgADP} + \mathrm{H^+}$$

where R denotes the amino acid being phosphorylated.
Kinase refers to proteins which catalyse this reaction, and those which catalyse the reverse are called phosphatases. In MAPK proteins, this addition alters the conformation from inactive to active, allowing the kinase to create complexes where further downstream phosphorylation can take place. Balancing the activation of these proteins is a critical function to maintain healthy cell activity. The reversible nature of the process allows for an effective mechanism for cells to regulate process based on the external stressors[318].

The RAS–RAF–MEK–ERK, or simply MAPK/ERK, is one of the most understood and well studied cascades[319]. It is of great interest to both molecular biologists and oncologists alike due to its pivotal role in regulating how the nucleus and its DNA respond to its environment and signals from the rest of a multicellular organism[320]. The pathway is responsible for transmitting extra-cellular signals to the nucleus, provoking responses to external stimuli such as performing mitosis, cell differentiation, and various stress responses[321]. Its role in mitosis makes it a key oncogenic factor in a variety of different cancers, as excessive activation is linked to tumour growth[322]. This has made it a key target for therapeutic intervention, and there are several viable points at which this is possible[323].

Located downstream of RAS and upstream MEK1/2 of in this cascade are the RAF family of Serine/threonine-specific protein kinases[324]. The family consists of 3 protein isoforms named ARAF, BRAF, and CRAF, and how much they overlap in function in still an open question. It is named Rapidly Accelerated Fibrosarcoma, due to its enhancing effect on the development of the disease in mice, the context in which CRAF was first discovered in 1983[325]. Despite this nomenclature, the family is highly conserved across metazoans[326]. They have been identified as one of the main targets which can cause faulty MAPK/ERK activation, and as such have received a significant amount of attention from the scientific community[327,328]. Knockout studies have solidified knowledge that the proteins plays a critical role in development and function, as mice without production of any RAF isoforms result in a lethal phenotype[329]. All have been experimentally observed as dimeric structures, and are thought to be functionally active in all 6 possible combinations of the 3. This dimerisation is required for wild-type activation, and is mediated by activated GTP-loaded RAS through a complex process resulting in membrane recruitment of the protein.[330]. Activation enables RAF to phosphorylate MEK1/2 in an ATP-dependent mechanism to propagate the signalling cascade.

All RAF isoforms share a very similar structure with 3 major conserved regions CR1-3[331]. The first of these, CR1, is formed of a RAS-binding domain and a cysteine rich domain which is capable of binding 2 zinc ions. A GTP-bound activated RAS binds directly to this region, and it is thought to also play a key role in membrane recruitment of the protein. CR2 is serine-threonine rich, containing many phosphorylation targets. This domain is known for its capability to bind 14-3-3 once activated, a regulatory protein that inhibits the active capabilities by preventing RAS binding[332]. CR3 is the main catalytic kinase domain, responsible for binding and phosphorylating MEK1/2. It has a structure heavily conserved amongst kinases, especially the MAPK-type. Its large C-terminal lobe is composed of mostly α-helices, which form the substrate binding site for MEK1/2. This is paired with a smaller N-lobe region composed of multiple antiparallel β-sheets which anchor bound ATP molecules. The active site, as is universal across kinases, is located in the cleft of these lobes. It is in this region ATP cleavage occurs,

freeing the phosphate group required for MEK1/2 activation. An overview of the domains of CR3 is shown in Fig. (2.7).

The loop-like region linking the two, known as the activation segment, plays a vital role in whether ATP can access the catalytic site. Phosphorylation of this region is the key driver in active-inactive conformational change. The segment is located between two highly conserved kinase motifs: the DFG and APE. In the inactive conformation the DFG occupies the ATP binding site preventing any catalytic activity in a conformation known as DFG-in[333]. Phosphorylation drives the conformational change to active, enabling solvent access to a hydrophobic catalytic spine capable of binding in a conformation similarly named DFG-out[334]. The residues involved in phosphate transfer vary between the isoforms, but in general precede the activation segment forming the catalytic loop (residues 574–581 in BRAF). It is in close proximity to the Walker A motif or P-loop, a key motif involved in ATP-binding. In addition, it is Fig. (2.8) shows the structural differences between these loops depending on whether the cofactors are bound. The disordered nature of these key motifs in RAF makes experimentally sampling this region difficult, and it is rare to find any fully resolved in experimental structures.

Of the RAF family, BRAF has received the largest amount of biological and clinical research since its discovery in the 1980s. It is the main target of most studies due to its role as a prolific oncogene mutations appear in over 66% of malignant melanomas, and at a lower frequency across other cancers[336]. A specific mutation, named V600E[i] is of note due to both its prevalence and efficacy in oncogenesis, accounting for 80-90% of the observed mutations in melanomas[334]. The significance can be accounted for by the fact it causes BRAF to become active in a monomeric form and without RAS, allowing for unregulated downstream activation[338]. This mutation is located on the activation segment, between the two phosphorylation targets in BRAF: T599 and S602. It alters the position of the DFG motif, moving it outside the ATP pocket without the need for dimerisation

[i]This may be specified, especially across older literature, using an alternative numbering such as V599E amongst others. This is due to a sequencing error which caused residues higher than 32 to be one less than their actual position[337]. This was first noticed in 2004, yet still easily causes confusion in the literature to this day.

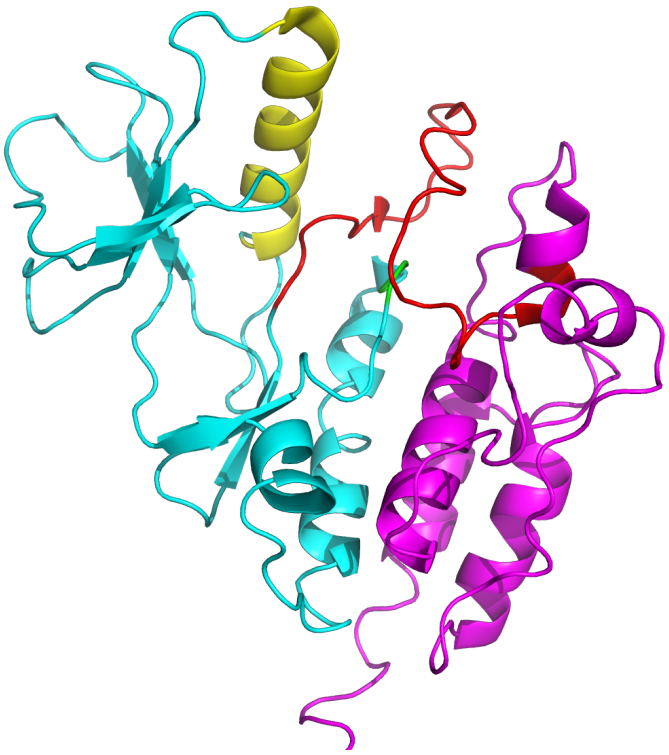


Figure 2.7: Diagram of the general conserved structures in RAF kinases CR3, highlighted on PDB ID 6UAN chain D, a rare structure featuring a fully resolved activation segment[335]. Yellow indicates the αC-helix, and green the P-loop segment. The rest if the N-lobe is coloured in cyan, to separate it from the magenta C-lobe. The activation segment connecting the two is show in red.

or phosphorylation. Search into verifying the exact structural reasons for this suffer from the poor resolution obtained in many crystal structures. Despite these challenges, drug discovery work on the V600E system has been incredibly successful.

BRAF has been the target of many drug developments, most notably vemurafenib, which served as an early success for the process of rational drug design[339]. This drug has shown through several clinical trials to improve patient outcomes in melanoma cases[340]. It is thought to competitively bind with ATP selectively on proteins featuring the V600E mutation, preventing the protein from binding ATP and propagating signals downstream. However this and similar treatments have been shown to enhance the dimerisation of wild-type RAF leading to paradoxical activation[341]. Despite the large interest in this system, and the substantial number of structures available, the physical process of BRAF monomer activation has not been fully characterised. Current theories hypothesise the introduced negative

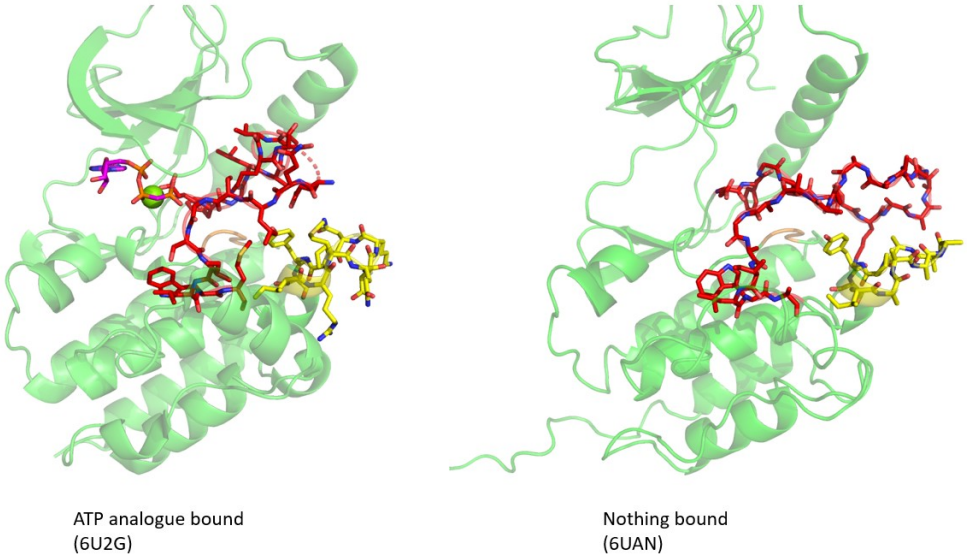


Figure 2.8: Figure showing a comparison between a BRAF structure with an ATP analogue bound (PDB ID: 6U2G) or an apo structure (PDB ID: 6UAN). The activation segment is shown in red, the catalytic loop is highlighted in yellow, and the P-loop is shown in orange. An ATP-analogue MgACP can be seen in the top-left. Dotted lines in the cartoon representation denote missing residues, unresolved during the crystallisation experiment.

charge of the glutamate mimics the phosphorylation of two nearby P-loop residues, or a salt-bridge forms between residue Lysine-507 on the αC-helix, both enhancing the stability the active conformation[342]. Alongside this the mechanism causing the paradoxical activation is still unknown. Previous work in the group focused on the categorisation of inhibitors, defining a 2D CV capable of separating ligand-bound dimeric structures into 3 distinct states[343,344]. However this did not provide a viable and interpretable path for the transition between states.

A better understanding of the conformational diversity of the activation segment would therefore yield a framework for further investigating the open questions about the transitional dynamics of BRAF and mechanism of the V600E mutation. BRAF shows the potential to be one of the most well-understood signalling proteins, and the activation segment seems to hold the answers to the few remaining questions about its signalling process. As kinases stand as one of the most studied protein families, there is a wealth of available information on related systems featuring a similar ATP-binding activation segments and mechanisms. This study aims to investigate if this large wealth of information

about similar systems can be leveraged to collate enough information to be able to inform our understanding of inactive-active conformational change for a BRAF monomer. It looks to develop a PDB database of homologous protein structures activation segments, and from this derive a suitable pathway for the transition. The high-dimensionality of the structures provides an ample opportunity to test the effectiveness of VAEs as an ML-based dimensionality reduction technique for working with distantly related atomic data. Applying a VAE-based network to this problem could provide an intuitive model of what regions of conformational space are available to the segment. Additionally interpolating over the support of this generated latent space of activation segments could be interpolated over to generate a potential transition pathway. This would allow for later enhanced sampling MD to be performed to obtain free-energy profiles for the transition. Once extended across phosphorylation states and with ligands bound, such a methodology could form an effective system for testing the many remaining questions about RAF and be used to enhance inhibitor design.

2.3.3 Methodology

The sequence chosen as the BRAF activation segment was determined using the two highly conserved motifs at either end, the DFG and APE motifs. The APE motif is observed to commonly form a small helix in many BRAF structures ending the disordered region of interest. The sequence for wild-type BRAF is the following 30 residue sequence:

DFGLATVKSRWSGSHQFEQLSGSILWMAPE (2.3.3.1)

To begin the analysis, a large-scale search of the PDB was performed to create a sequence database of homologous BRAF structures. This search tool uses the MMseqs2 algorithm to parse similar structures over all deposited structures[345]. As input to this search, the full wild-type BRAF sequence was used. To further increase the coverage for more distant kinases, this was further supplemented using the BLAST search algorithm also performed across the PDB. This search used just the activation segment as a reference, and aimed to find any more distantly related structures featuring a similar motif[346]. Due to the short length of the sequence, this BLAST search was performed using a word size of 3, using the PAM-30 scoring matrix, and gap costs of $(9, 1)$. These parameters are optimised for finding matches in queries under 35 residues[347]. This search returned all structures with an e-value below 0.05.

Every structure in this database was then processed to remove all waters, heteroatoms, and separate structures into individual chains. Additionally a SCOP family search was performed to identify the protein superfamilies these came from, and those not from a protein kinase clade were discarded[265]. From these individual chain files, primary sequences were then extracted to form a local BLAST database. To rank the proteins similarity an additional BLAST search was performed across this aiming to rank the primary sequence based relevancy of each chains structures. For this all chain sequences were compared to the full wild-type BRAF sequence. This used a word length of 5, the BLOSUM-62 score matrix, and gap costs of $(11, 1)$.

Searching across these separated chain-sequences and structures yielded an e-value for each sequence in the database, and any 0 scoring structure were labelled

as homologous to BRAF in further analysis. As variation across the activation segment was the main concern, to maintain the relevancy of the data these results were then filtered to exclude examples that did not contain motifs similar to the DFG and APE. This was evaluated from the local BLAST sequence alignment, where any chain unable to provide matches to these motifs were disposed of. Allowing some variance was important as these motifs have been shown to vary somewhat, including the isoform C-RAF which features an uncommon AAE motif to end the segment. A further structural alignment was performed using the MUSTANG algorithm to help analyse the structural conservation across this database. Cryo-EM structure PDB ID 6UAN chain D was chosen as the reference as it has a rare fully resolved activation segment, providing the best example of a realistic conformation[335].

Relating this data back to BRAF suffered from two clear issues. Much of the data is from sequences of different lengths due to frequent insertion or deletion mutations along the segment. Also as a loop, many key residues are often missing any positional data as these regions are hard to experimentally resolve leading to missing data points. This latter issue can be seen in Fig. (2.8), where the ACP bound structure is clearly missing residues. Therefore to further compare these activation segment structures, a method was developed to fit the aligned residues to a set length using a cubic fitting procedure. As the APE motif showed little variation over BRAF structures, the modelled sequence was taken to be the first 27 residues of the activation segments including the DFG. The dimensionality of each sample was made consistent by taking the C_α atoms for each activation segment and performing a cubic interpolation between them. Sampling the resultant curve 27 times gave the necessary number of positions. A schematic of the process is shown in Fig. (2.9).

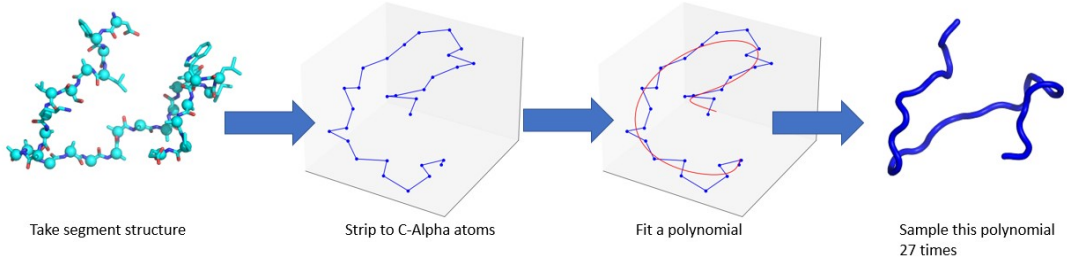


Figure 2.9: A schematic of the fitting process performed across the data.

To begin with this method was tested on the structurally aligned PDBs outputted when the MUSTANG alignment was performed. However hand-checking these often showed poor results around this region as it was unable to exactly align the very different orientations of the DFG motif and subsequent loop. To avoid this issue, the aligned structures used further were derived by matching the BLAST aligned residues to the reference structures DFG and APE motifs. Subsequently a RMSD-minimisation procedure was performed between just these 6 core C_α residues. This ensured all loops occupied a similar space. The fitting procedure was then performed for all PDB structures which had an aligned activation segment close to 27 residues in length. This cut-off was taken to be ± 10, not counting missing residues, as severe outliers greater than this were found to lead to clearly unphysical results. A PCA was performed across the fitted segments to analyse the conformational diversity in terms of residue contributions by calculating the PCA RMSD modes or PCA scores.

While the PCA provided a good linear way to project the data down to a manageable space, interpolating over the structures did not provide a consistent path. To this aim a non-linear fitting was also performed using a VAE. The segment data consisted of 27 3D-coordinates from the aligned and fitted structures. which were processed using 4 feed-forward layers in both the encoder and decoder as a 81 dimensional vector. Gaussian forms for both the priors and the approximate posterior were assumed, and the ELBO was used as the loss function. The encoder layer sizes decreased from an initial 128 dimensions in powers of 2, before an embedding was produced. The decoder used the same pattern but in reverse. A 90/10/10 training-validation-test split was used, using the validation set to perform hyper-parameter search to find the optimal dimensionality of the latent space. This tested using 3, 5, 10, and 15 dimensions for the latent space representation. The learning rate was set to 1×10^{-5} and data separated into 200 batches. Training was performed in PyTorch on an NVIDIA Quadro RTX3000 GPU. Performance on both the training and validation sets were monitored throughout preliminary runs, using early stopping to find where the model began to overfit. 150 epochs was empirically found to be sufficient for model convergence while avoiding overfitting to the training set.

In addition to verify the procedures capability to model the dynamics of BRAF, a short MD simulation was performed for 300ns using the NAMD engine[348]. This was performed using a monomer structure based on a model of the DFG-In conformation in a dimer state developed for previous work[344]. To maximise the amount of conformational space explored, this model was additionally phosphorylated on residues T598 and S601 to promote its transition to DFG-Out. It was performed using the CHARMM36 FF, with a Langevin thermostat set to 300K, and Langevin piston barostat set to 1atm. Both Langevin control methods used a damping coefficient of $1ps^{-1}$, and the time-step was set to 2fs A rectangular unit cell was used, fitted to 10Å from the protein and solvated with TIP3P model water molecules. 62 K^+ and 73 Cl^- ions were placed to neutralise charge in the system. PBCs were used with the PME method for long-range electrostatics. Production run MD was preceded by 1000 steps of minimisation using the steepest descent algorithm followed by 10ns of NVT equilibration.

2.3.4 Results & Discussion

Initial searches located 7537 individual chains from both the PDB and BLAST search. The PFAM superfamily identification showed 6827 of these were from protein kinases. This was just over 90%, showing that the sequence was majorly conserved amongst kinases. A full break down of the SCOP superfamily search is provided in Fig. (2.1) in the appendix. The local BLAST showed this database included 143 structures identified as BRAF homologues, and corresponding PDB and chain IDs are shown in the appendix Fig. (2.15).

The MUSTANG structural alignment results are shown in Fig. (2.10)(a). It highlighted the many key regions of CR3 that struggle with experimental resolution even across homologous structures. These regions are displayed in Fig. (2.10)(b). The most interesting result was the very low percentage ($\approx 20\%$) of homologous BRAF sequences which had activation segments (residues 594-623) aligned to the reference structure. This showed a more significant variation across the start of the sequence near the DFG, with APE motifs showing a very stable configuration even across non-homologous structures. The rest of the unstable regions were confined to similar loop-like segments, which is to be expected given their more disordered and flexible nature. The consistent alignment results across the rest of the database provided support that it may be possible to leverage data from a wider range of structures than just BRAF for study.

Discarding regions which had unaligned DFG and APE segments left 2702 sequences in the database. Fitting these activation segments using the outlined procedure allowed for a comparison of just this region to be made. The first four components from performing a PCA on this data is shown in Fig. (2.11), accounting for 84% of the overall observed variance across the data. The first PCA component separates the majority of BRAF sequences well, shown by the distribution of the projection in Fig. (2.11)(a). It also represents the vast majority of the variance across the dataset, shown by the magnitude of the associated PCA score in Fig. (2.11)(b). This highlights the most fluctuating area as around V599-S606, the region containing the V600E oncogenic mutation. It has a consistent fluctuation across all PCA scores apart from the second. While the first two

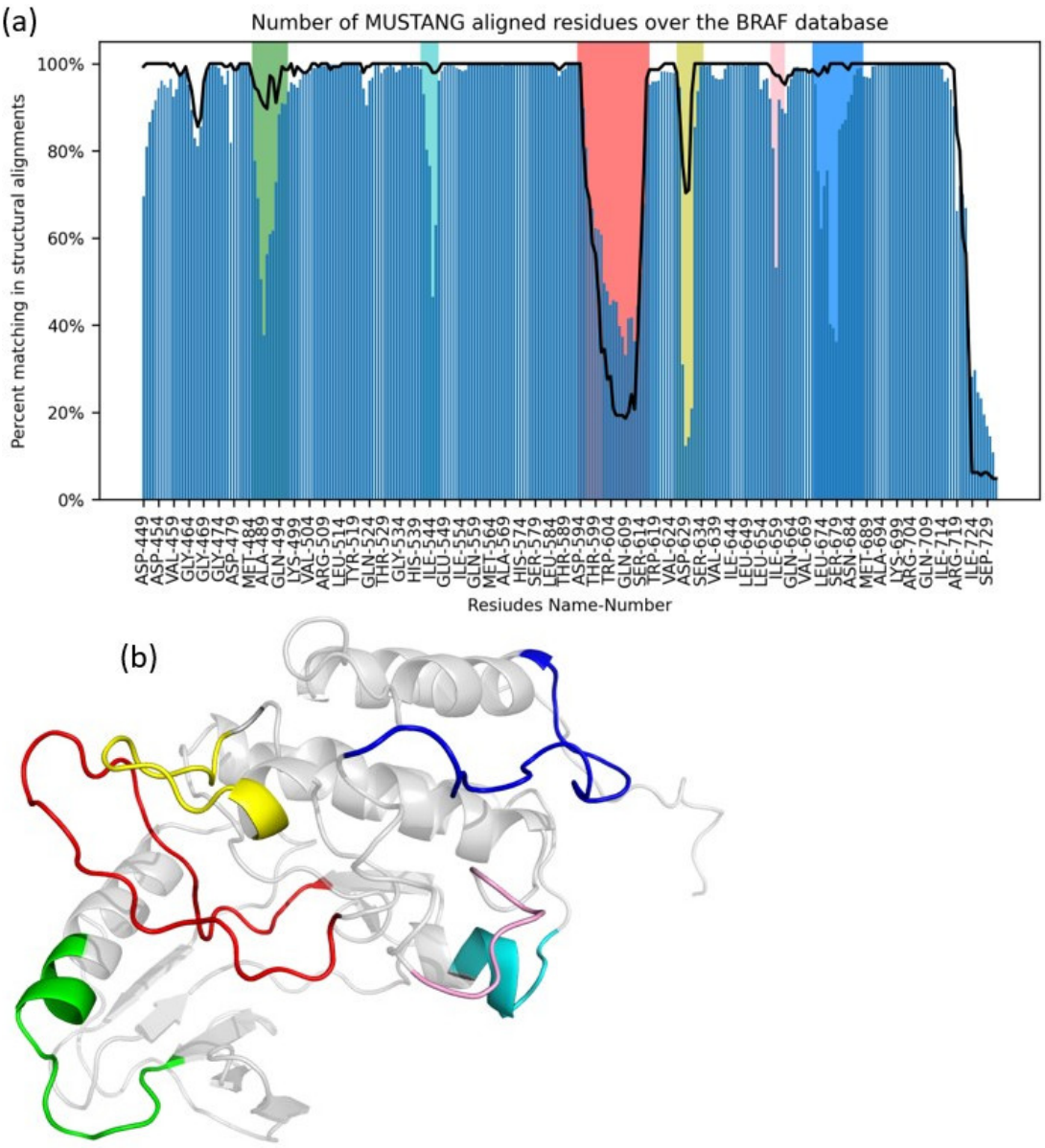


Figure 2.10: The MUSTANG structural alignment results across the PDB data. ***(a)*** *shows the number of conserved residues over the acquired PDB structures. Each structure was fitted to 6UAN, and then the results were converted into a percentage of how often each residue in the BRAF structure was matched to another sequence. The black line shows this result across structures labelled as BRAF (E-value = 0). Regions of notably poor alignment are highlighted in various colours, with* ***(b)*** *showing their position on the 6UAN BRAF chain D structure using the same colour-scheme.*

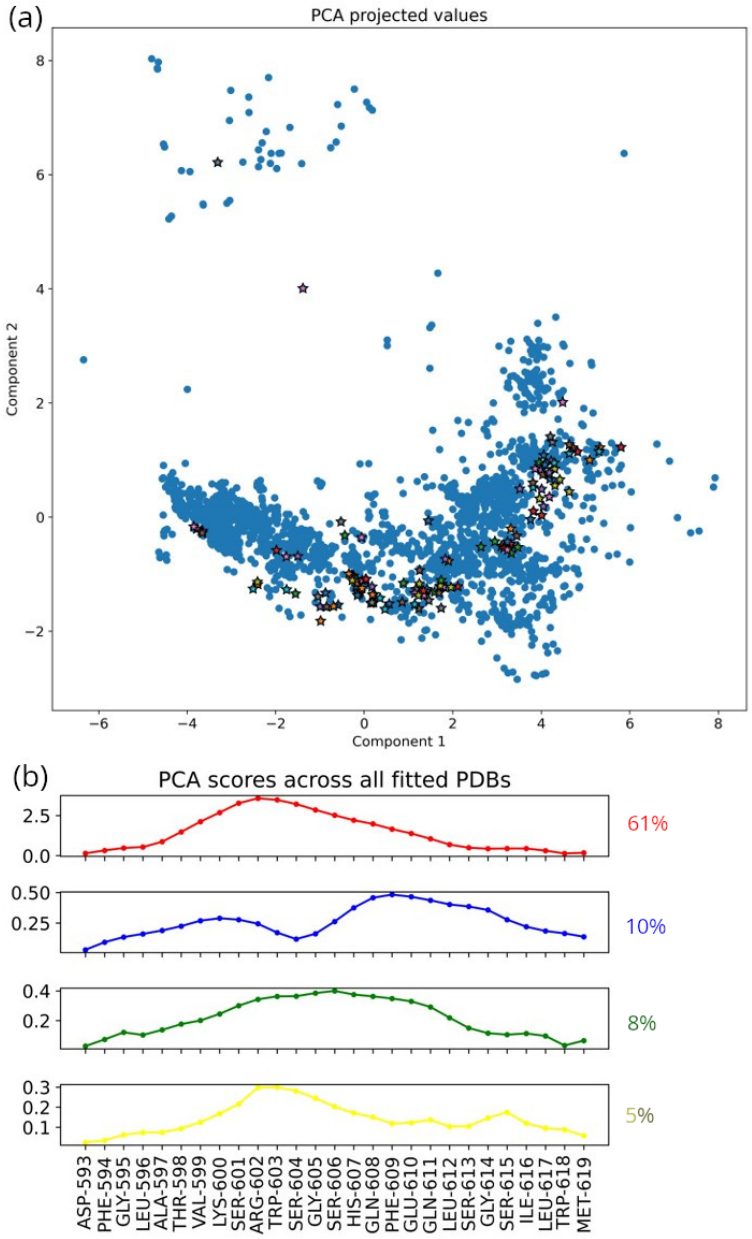


Figure 2.11: The PCA results attained over the fitted BRAF segments. ***(a)*** *shows the projections from the first two PCA component values for each structure in the database (blue), homologue structures are marked with coloured stars.* ***(b)*** *shows the corresponding scores, calculated as the weighted RMSD PCA modes.*

components were able to separate many of the PDBs, interpolated structures suffered from the sparsity of the data and were often unphysical.

All of the fitted segments are visualised in Fig. (2.12)(a), where they are coloured by the value of the first PC component. This component is shown to be correlating with the angle of protrusion of the activation segment relative to

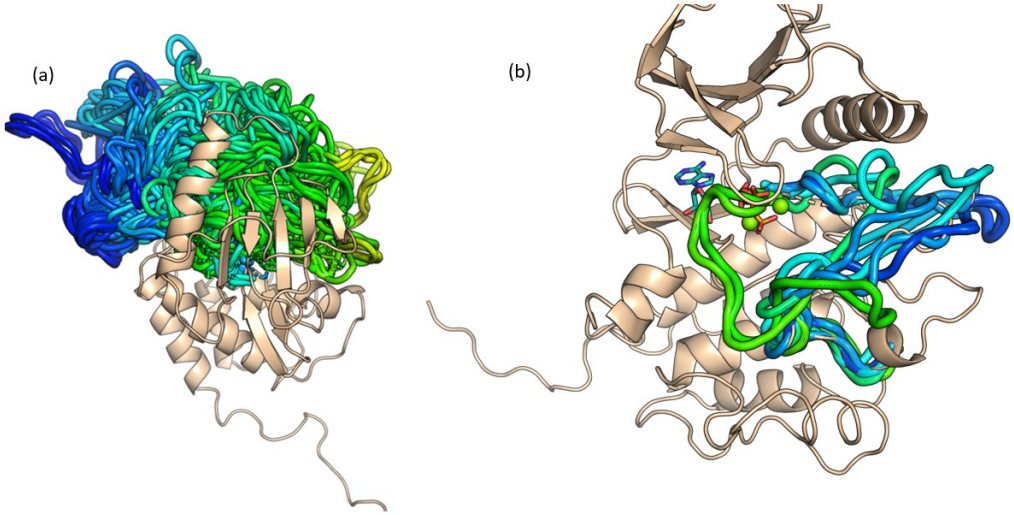


Figure 2.12: The fitted structures coloured according to their PCA component 1 values, from blue (low) to green (high). ***(a)*** *shows all fitted sequences in the database, superimposed on the 6UAN reference structure.* ***(b)*** *shows this fitting amongst selected structures labelled as BRAF, to more clearly show the structures. ATP is also shown here towards the left of the figure to show how these different PCA values represent different interactions with the binding site.*

the ATP binding site, consistent with the DFG-out/in transition. The two main outliers of the BRAF labelled structures, seen as the stars in the top left quadrant of the PCA projections, have a standard value for the first component but very different across the second. Both come from the same X-ray PDB structure 7P3V, and feature a novel inhibitor dabrafenib bound, which has been shown to force the segment into a novel conformation[349]. They are clearly visible as the very dark blue fits in the left of Fig. (2.12)(a), protruding from the main cluster. Fig. (2.12)(b) is a direct comparison of several structures sampled across the component's value. Here the segments can be seen to have both relatively unrealistic interatomic distances and meandering conformations. Despite these unphysical properties of the fitted data, it separates the locations of the DFG remarkably well.

These 2702 fitted structures were then used to train the VAE. Preliminary hyper-parameter optimisation results, shown in appendix Fig. (2.17), showed there was no clear separation in terms of reconstructive ability between any of the tested latent space dimensionalities. All models were capable of obtaining

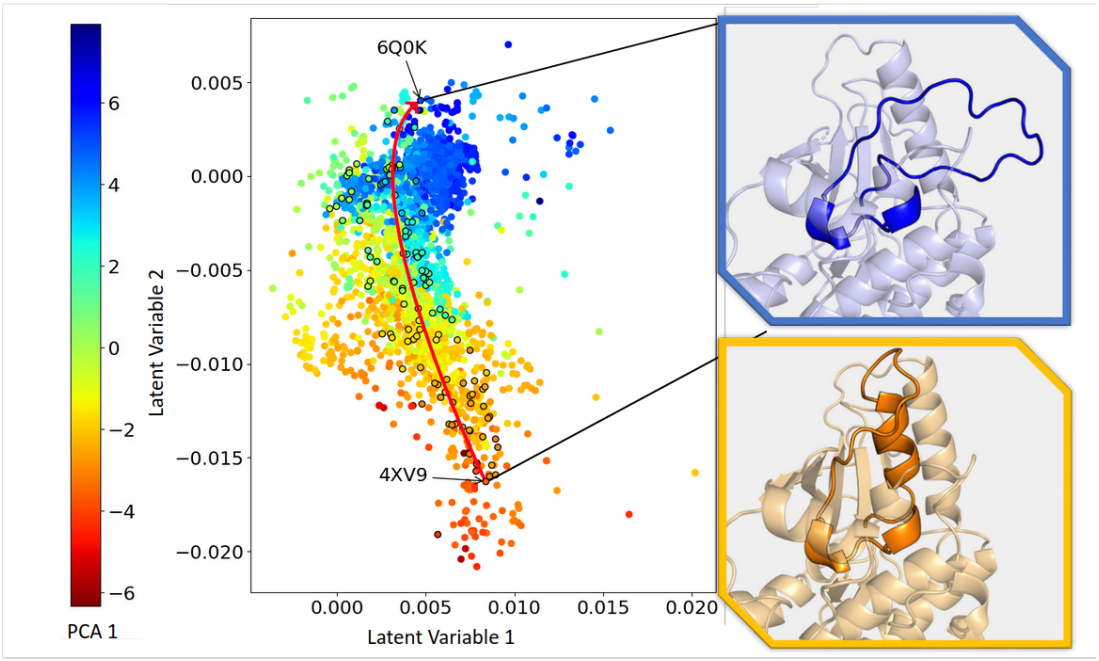


Figure 2.13: A projection of the Kinase activation segment data onto the first two dimensions of a trained auto-encoder with a 3D latent space. Values are taken from the averages μ of modelled Gaussians from each data point. Axis are dimensionless, and the points are coloured by the normalized first PCA component. Outlined points correspond to homologous BRAF structures, and labelled are the most extreme values for these structures. The associated image shows the PDB structure of the labelled activation segment.

an average reconstruction less than 5.4Å on the validation set. For this reason, the 3-dimensional model was chosen as it had fewer latent dimensions, in theory making each one more representative. This model had a reconstruction loss of 6Å across the test set. Projections onto the first two dimensions of a trained VAE, highlighting homologous BRAF structures, are shown in Fig. (2.13). It also found an additional outlier, X-ray structure 6P3D which is bound to another inhibitor ponatinib which has been shown to induce a distinct conformation through stabilization of the αC-helix[350]. All BRAF homologous structures identified as structural outliers are labelled in the VAE projection in the appendix Fig. (2.18).

Labels for the extreme values of the first coordinate are included in Fig. (2.13). This places X-ray structure 4XV9 as the lowest projected structure. It corresponds to a model in a complex with novel binder PLX5568, shown in the appendix Fig. (2.16). Designed to avoid paradoxical activation of BRAF, it enforces the DFG-Out conformation which typically is active and associated with increased ATP

binding[351]. The highest projected structure is the electron-microscopy derived model 6Q0K, which is a large structure of a MAPK complex featuring a BRAF dimer alongside MEK1 and the associated scaffold protein 14-3-3[352]. In this state of the MAPK pathway, BRAF shows its ability to auto-inhibit in the DFG-in conformation. These two functionally opposite states being separated clearly by the first latent space dimension pointed towards the separation having functional relevancy.

For the final step of this work, mapping from the developed VAE latent space back to BRAF structures was attempted. However a suitable methodology was not found. Despite this, the trained VAE displayed a promising representation of the viable regions of conformation space for the activation segment. This is shown in Fig. (2.14), where segment conformations obtained from the unbiased MD of the wild-type have been mapped to the latent space. However before performing this mapping the data was put through the fitting process. Failing to do so resulted in the trajectory data occupying a disparate region of the latent space in comparison to the structural database. This is somewhat expected, as the fitting process has been seen to have a dramatic impact on the observed conformation. The projected data does show the modelled latent space represents a much greater region of conformational space than is usually explored during a standard MD run.

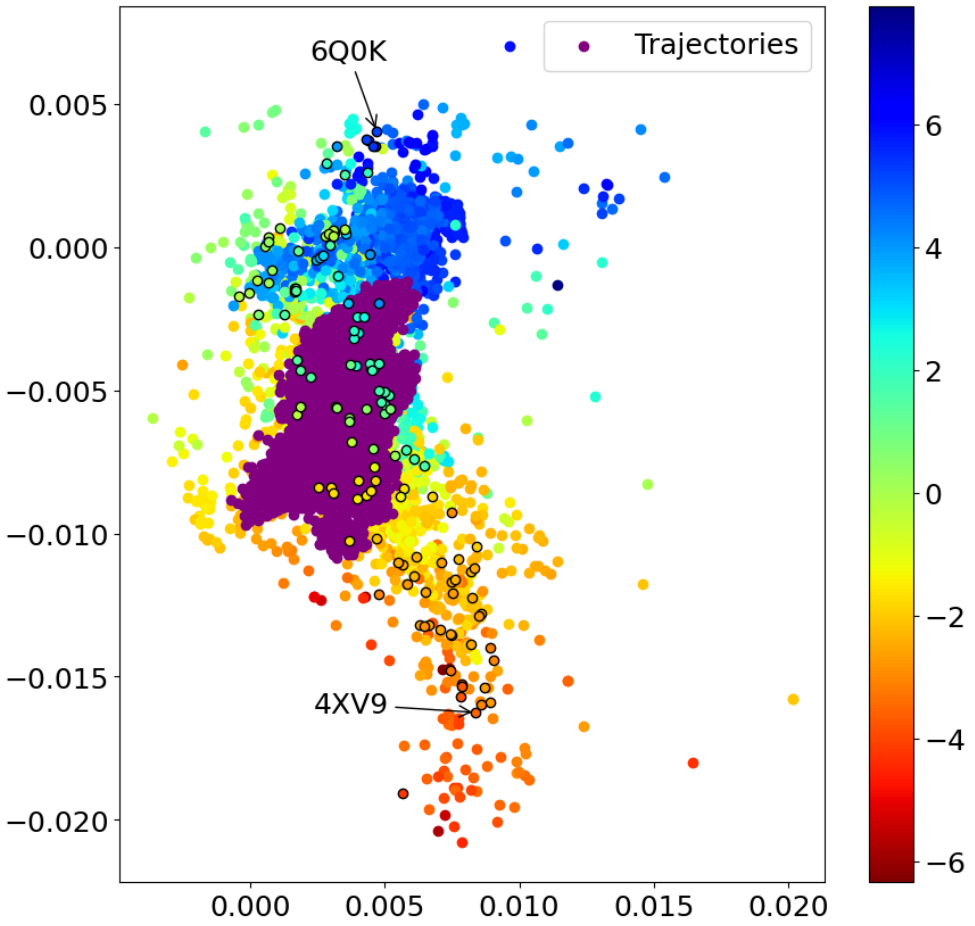


Figure 2.14: Projections of the first two VAE dimensions for the BRAF database with projections from MD data of the DFG-In trajectories superimposed.

2.3.5 Conclusion & Future Work

The progress made highlights the great potential for VAEs to be used to provide automated CVs built from experimental data. It has been shown to be a valid approach for identifying structural outliers from sequence searches and proposes an RC able to separate known functional states. In the future this could form part of a investigative pipeline for exploring the available evidence for the conformational space available to region of interest. As a general strategy for discovering CVs this could be used to make biasing simulations much easier. Requiring a large data parsing pipeline able to mine the most relevant data for the region of interest rather than expert input. In future work, this could be implemented alongside techniques such as meta-dynamics, which require a well-defined CV space to

explore and derive free energies for. However capitalising upon this for simulation purposes would require significantly more thought.

Interestingly despite the advancement in techniques, this project struggled with possibly the most fundamental problem of structural alignment. That is how to effectively compare structures of differing length. While mapping the simulation to the latent space is trivial, performing the opposite suffered from several issues. Most notably, the ends of the segments varied greatly in their final positions making swapping an activation segment for one inferred from the latent space very difficult. Fitted structures also lost any rotational information, and sometimes featured unrealistic bond lengths as they were derived from different length segments. These latter problems resulted from structures that were significantly changed by the fitting process, and improving its ability to model realistic conformations could alleviate this. Alternate solutions are to instead sample from the latent space and use this to inform a homology modelling procedure where realistic structures with a minimal RMSD are constructed. This would require significant expert attention, negating the automated features of this approach. However the efficacy of automated homology modelling approaches, such as the MODELLER software, is yet to be seen in this approach[353].

This work failed to provide a suitable transition path between the known BRAF states, but it does lay the groundwork for future projects to assess this. Activation segments sampled from the laten space could be used to form an internal distance contact map providing values to bias a simulation towards. Implementing a simple umbrella potential to direct a wild-type simulation towards the given conformation may provide a systematic approach to interpolating physical conformations from the latent space. Even if the latent space heavily effected by the fitting procedure, limiting the strength of the bias could still enable sampling of realistic physical space. However again this is a significant amount of effort and would require a lot of analysis to ensure the MD is viable. Due to time and computational constraints this was never tested.

Other, larger ML projects could also be used to alleviate some of these issues. Since the project, structure prediction networks like AlphaFold have become significantly more powerful and prominent[36]. They could provide one potential remedy to both these problems by predicting a more realistic structure around a

given activation segment. Additionally these models' internal representations could be leveraged as a starting point for our model. However this would perhaps require its own dedicated project to be feasible, but designing a usable methodology and user interface for this process could be an interesting further development. Predictions by networks such as these also suffer from the fact they simply search existing structures, meaning they regularly struggle on loop-like structures such as the activation segment. For these regions, there is a no consensus from the PDB data. A possible avenue is using networks specifically trained on disordered regions, taking inspiration from networks that already exist to predict these regions[354].

2.3.6 Appendix

Superfamily	**Proportion of PDBs** (%)
Protein kinase-like (PK-like)	90.58
Immunoglobulin	4.72
SH3-domain	0.88
Growth factor receptor domain	0.77
KA1-like	0.57
NTF2-like	0.52
PH domain-like	0.27
SMAD/FHA domain	0.19
Cysteine-rich domain	0.19
SH2 domain	0.15
ARM repeat	0.15
cAMP-binding domain-like	0.13
Ubiquitin-like	0.12
Chaperone J-domain	0.11
E set domains	0.09
Carbohydrate-binding domain	0.08
UBA-like	0.05
SIS domain	0.05
AF1104-like	0.05
Nucleoside diphosphate kinase, NDK	0.04
Thyroglobulin type-1 domain	0.04
Regulator of G-protein signaling, RGS	0.04
SAM/Pointed domain	0.04
(Trans)glycosidases	0.04
Aminomethyltransferase beta-barrel domain	0.03
Galactose-binding domain-like	0.03
Other	$<$0.001

Table 2.1: Superfamily search results using PFAM across the constructed chain database. Any superfamily with less than 100 structures is grouped into the other category.

6UAN_C	3PRF_A	3Q96_B	4XV2_B
7MFD_A	3TV6_B	3Q96_A	4XV2_A
6Q0K_B	3TV6_A	5HIE_B	4XV3_B
6Q0K_A	3TV4_B	5HIE_D	4XV3_A
6Q0J_A	3TV4_A	5HIE_C	6V34_B
6Q0J_B	3Q4C_B	5HIE_A	6V34_A
6NYB_A	3Q4C_A	4H58_B	3OG7_B
4MBJ_B	4EHE_B	4H58_C	3OG7_A
4MBJ_A	4EHE_A	4H58_A	5ITA_B
3II5_B	3PPJ_B	6N0P_B	5ITA_A
5FD2_B	3PPJ_A	4YHT_B	6P3D_A
5FD2_A	3PRI_B	4YHT_A	5JSM_B
4MNE_C	3PRI_A	4XV9_A	5JSM_D
4MNE_F	3IDP_B	5CSW_B	5JSM_C
4MNE_B	3IDP_A	5CSW_A	5JSM_A
4MNE_G	4MNF_B	4JVG_B	5JT2_B
3PSD_B	4MNF_A	4JVG_C	5JT2_D
3PSD_A	4EHG_B	4R5Y_B	5JT2_C
3SKC_B	4EHG_A	4R5Y_A	5JT2_A
3SKC_A	5C9C_B	4WO5_B	5JRQ_B
4E26_B	5C9C_A	4WO5_A	5JRQ_A
4E26_A	6PP9_A	4FK3_B	4CQE_B
3PPK_B	5CT7_B	4FK3_A	4CQE_A
3PPK_A	5CT7_A	3C4C_B	7P3V_B
4E4X_B	4KSP_B	3C4C_A	7P3V_A
4E4X_A	4KSP_A	4RZW_B	5HID_A
4PP7_B	4FC0_B	4RZW_A	5HI2_A
4PP7_A	4FC0_A	4RZV_B	
3D4Q_B	4DBN_B	4RZV_A	
3D4Q_A	4DBN_A	6U2H_D	
3PSB_B	4KSQ_B	6U2H_C	
3PSB_A	4KSQ_A	6XFP_A	
4G9C_A	2FB8_B	4XV1_B	
3PRF_B	2FB8_A	4XV1_A	

Figure 2.15: PDB IDs of all PDB structures identified as homologous by BLAST. Each structure obtained an e-value of 0 *during the search. Structures are named by their PBD ID suffixed with the corresponding chain they are derived from.*

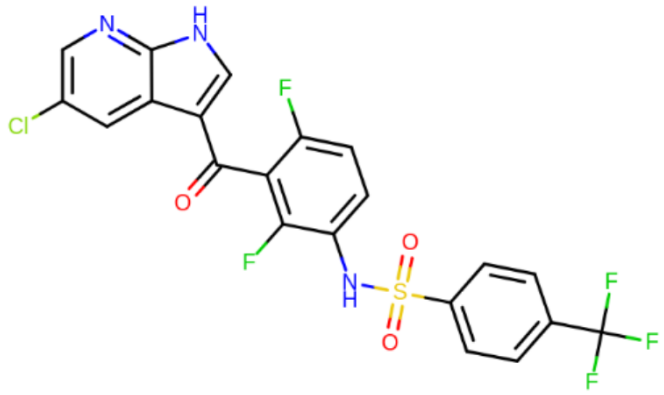


Figure 2.16: Chemical diagram of novel inhibitor PLX5568, found in PDB ID 4XV9.

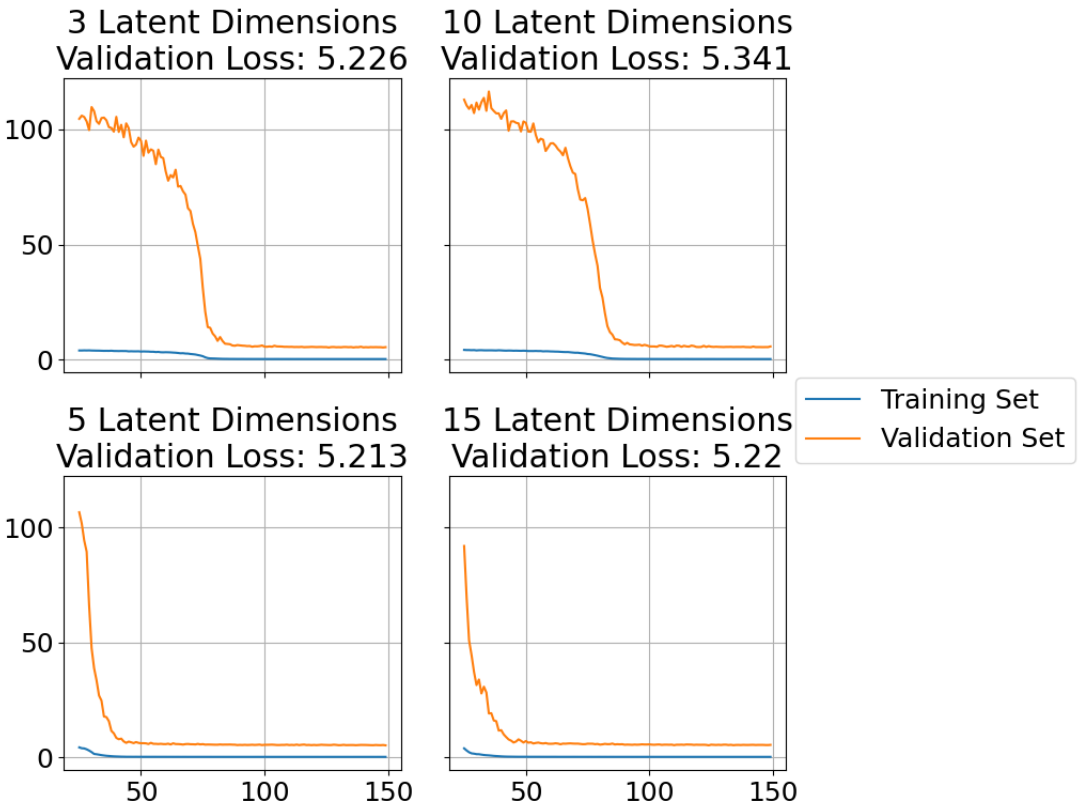


Figure 2.17: Plots of the training and validation loss for the 4 different latent space sizes trained. Loss values from the first 10 epochs are not displayed, due to their much larger magnitude, to aid comparison between the models.

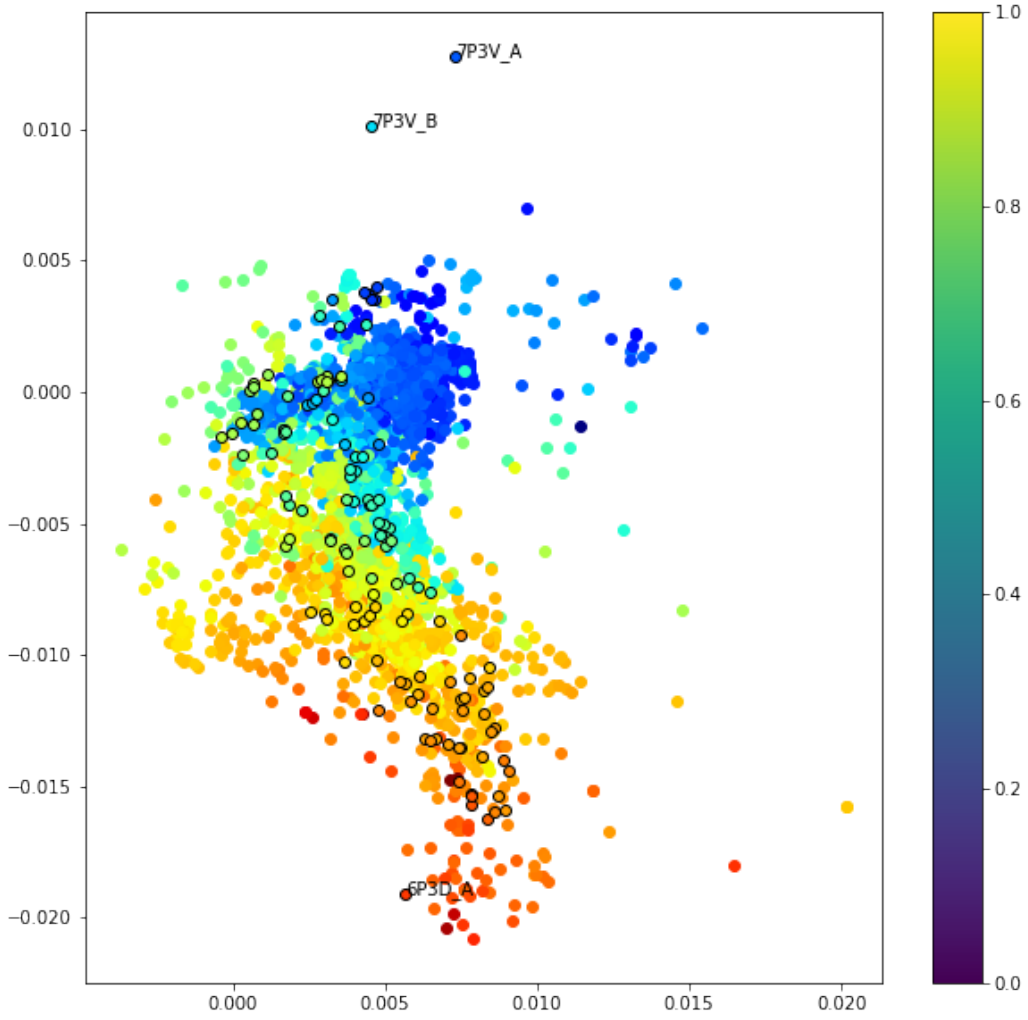


Figure 2.18: A projection of the Kinase activation segment data onto the first two dimensions of a trained auto-encoder with a 3D latent space. Homologous outliers are labelled with the corresponding PDB and chain ID for these structures.

2.4 Case Study 2: The functional dynamics of NSP13

2.4.1 Overview

The arrival of the Coronavirus pandemic in early 2020 showed the rapid mobilisation of the scientific community in the face of adversity, as many shifted their research focus to applying their skills to further understand and help combat the new disease. This work began at this time, as researchers looked at the viral genome of SARS-CoV-2 to find critical proteins suitable as treatment targets. It serves as a textbook example of applying the target investigation process to the initial stages of drug design, covering the process from initial modelling tasks to attempts at simulating biological function.

The target of the work was Non-Structural protein 13, a helicase protein vital to the replication process of SARS-CoV-2 virus. It shows significant promise as a therapeutic target based on its functional importance alongside the many similar helicase-focused antiviral treatments available. This part encompasses two distinct projects, with the initial work focusing on modelling and simulating the protein, alongside a later attempt to obtain a viable translocation pathway. The first project was focused on developing an effective protocol for developing a realistic model and simulation for both RNA-bound and unbound structures. During the course of this I was responsible for performing the kinetic analysis across the MD simulations and producing the associated figures.

After publication of these results, the project moved forwards to applying this gained knowledge in order to further study the translocation process. Here I was responsible for developing a method to attempt to observe translocation and performing the simulations. Although the final results did not culminate in a full observation of the functional process, it demonstrates a potential approach and highlights the difficulty and sensitivity of sampling functionally relevant kinetics from these systems. It serves as an example of how many of the discussed techniques can be utilised in practice. This work resulted in the publication of two papers. The first summarising the group's modelling and simulation efforts, and a later review distributing our knowledge about the helicase translocation process:

- Berta, D.; Badaoui, M.; Martino, S. A.; Buigues, P. J.; Pisliakov, A. V.; Elghobashi-Meinhardt, N.; Wells, G.; Harris, S. A.; Frezza, E.; Rosta, E. Modelling the Active SARS-CoV-2 Helicase Complex as a Basis for Structure-Based Inhibitor Design. Chem. Sci. 2021, 12 (40), 13492–13505. https://doi.org/10.1039/D1SC02775A

- Horrell, S.; Martino, S.; Kirsten, F.; Berta, D.; Santoni, G.; Thorn, A. What a Twist: Structural Biology of the SARS-CoV-2 Helicase Nsp13. Crystallography Reviews 2023, 29 (4), 202–227. https://doi.org/10.1080/0889311X.2024.2309494

2.4.2 Background

Severe Acute Respiratory Syndrome Coronavirus-2 (SARS-CoV-2) is the causative agent of COVID-19 disease and is responsible for the largest modern pandemic, with over 6 million reported deaths worldwide. Belonging to the viral family Coronaviridae, the family is separated into the genera alpha-, beta-, or gamma-coronavirus depending on evolutionary history. SARS-CoV and SARS-CoV-2 are the most notable and prevalent examples of betacoronaviruses, responsible for several human pandemic outbreaks[355]. Early study into COVID-19 benefitted from the wealth of work available for this genus pioneered during other betacoronavirus outbreaks, such as the 2003 SARS outbreak and the 2012 Middle East respiratory syndrome epidemic[356,357]. The virus is a positive-sense single stranded RNA (+ssRNA) virus, meaning it carries its genome in an mRNA form which requires no prior translation to be used by the host cells ribosomes[358,359]. The genome is around 26.0 kb to 32.0 kb nucleotides in length[360]. As with most viruses, the proteins encoded within can be separated into structural proteins which form the virus itself, and Non-Structural proteins (NSPs) which enable replication within infected cells. The mRNA payload contains the necessary code for 16 NSPs, 4 structural proteins constituting the capsid, and 6 accessory proteins[361]. These are synthesised within the hosts ribosome as long polyproteins called pp1a and pp1ab, which are cleaved into the final proteins by the main protease, or Mpro, of the virus[362]. Once expressed within an infected cell, the NSPs coalesce to form the RNA-transcription complex (RTC) responsible for all replication processes

occurring outside of the viruses capsid. Formation of this complex functionally activates the constituent proteins, and it serves as one of the main therapeutic intervention target along the replication cycle alongside the Mpro[363]. Part of this complex, comprising of the functional unit around NSP12 known as the RNA-dependent RNA polymerase (RdRp), was successfully crystallised early into the pandemic and is shown in Fig. (2.19)(a)[364].

The RdRp is highly conserved across coronaviruses, and shares a structure consistent with all known viral polymerases. Its main function is to produce copies of the viral-RNA through the relevant NTP molecules binding to its active site[365,366]. This vital function has made it a clear drug target for a large class of antiviral drugs, most notably the prodrugs remdesivir and molnupiravir used in the treatments of COVID-19. These drugs are metabolized within the cell to produce nucleotide analogues which exhibit binding to the NTP pocket[367]. The addition of a remdesivir derivative to the RNA chain causes stalling of the transcription, inhibiting replication and offering a statistically significant drop in mortality rates across patient groups[368]. Available treatments are further discussed in Case Study 4.

Found also in the crystallised structure of the RTC are two copies of the SARS-CoV-2 RNA helicase, or NSP13. Much of its function can be derived from studies into the analogous protein in SARS-CoV-1, as it differs only in a single replacement I570V. Early structures of crystallised dimers show an almost identical structure between the two, however featuring a different interface between the two monomers[369,370]. It has two known functions. Firstly it performs the first step in the 5'-capping of the viral RNA via its triphosphatase function[371,372]. This process of hydrolysing the 5'-triphosphate group is key to allowing copied RNA strands to further replicate by allowing identification by the hosts ribosomes[373,374]. Furthermore it enables RNA translocation and unwinding through an ATP-dependent mechanism during viral replication. Based on sequence classification, NSP13 belongs to the helicase protein superfamily SF1B[375]. SF1 is the second largest helicase superfamily, containing many well-characterised helicases such as PcrA and Rep and a relatively high level of conservation between its members[376 377]. NSP13 belongs to SF1B subfamily as it translocates a RNA strand in the 5' end to 3' end direction, placing NSP13 in the same category as other well-studied

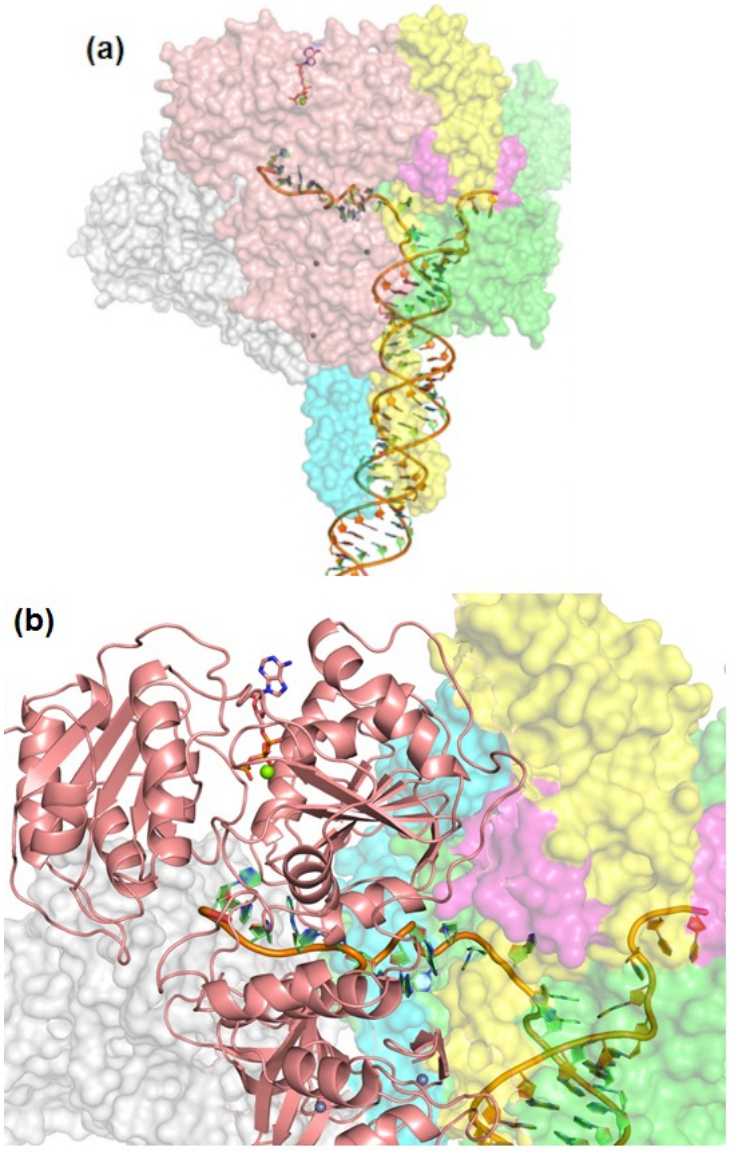


Figure 2.19: ***(a)*** *The structure of the RNA transcription complex, showing both a MgATP molecule bound to NSP13 (top) and a strand of viral RNA. The different NSPs are shown using coloured surface representations. NSP7 is shown in purple, NSP8a and NSP8b are shown in yellow and cyan respectively, NSP12 (or RdRp) in green, and the two copies of NSP13 are shown in salmon and white. In this complex the salmon NSP13 is active and bound the the ssRNA, while the white is inactive. It is unknown if this dimerisation is required for function.* ***(b)*** *A close-up of the NSP13 RNA binding cleft and ATP binding pocket. Both structures are derived from the electron microscopy model PDB ID 6XEZ*[364]*.*

replicative DNA/RNA helicases such as RecD2[378 379]. Interestingly this differs from the 3'-5' directionality of the RdRp inside the RTC structure, leading to theories of NSP13's potential role in backtracking. This is where the complex moves the



Figure 2.20: The domains of NSP13 coloured on PDB 6ZSL, an apo structure. The ZBD is green, Stalk is cyan, Domian 1B is wheat, RecA1 is orange, and RecD2 is purple.

t-RNA 5'-3' with the helicase playing a role in reverse-threading the ssRNA 3' end through the polymerase NTP binding site[380,381]. While this is a known event seen in other DNA-dependent RNA-polymerases, SARS-CoV-2 is evolutionary distant from these leading to questions about its functional significance and evolutionary development[382,383]. Backtracking has been hypothesised to be an evolutionary mechanism for aiding DNA repair in studies on the similar SF1 UvrD helicase[384]. These critical tasks and clear questions about its function make it an additional target for both pharmaceutical targeting and further research, expanding research directions across the RTC[385].

The structure of NSP13 can be decomposed into several distinct domains, highlighted in Fig. (2.20). Its sequence begins with a Zinc Binding Domain (ZBD) (residues 1–100) capable of binding 3 Zinc ions and known to aid in nucleic acid identification[386]. This is followed by 2 α-helices forming the Stalk domain (residues 100–150), linking the ZBD to Domain 1B (residues 150–260) which takes part in

RNA-binding. Like other SF1B helicases, the subsequent regions are decomposed into two RecA-like folds RecA1 (residues 261-442) and RecA2 (residues 443-601), in a structural pattern seen throughout the family. These two folds handle the majority of the functional work when it comes to the RNA translocation. The RNA binds between them in a large cleft, and the interface between the two domains forms the ATP binding pocket.

The conformational changes necessary for translocation therefore happen in the region between the bound RNA and ATP binding pocket, shown in Fig. (2.19)(b). Current work on NSP13 supports the "inchworm" type mechanism. This is where ATP hydrolysis drives tightly bound contacts in separate RecA-like domains to cyclically take turns grabbing and releasing the nucleic acid, moving it forward a nucleotide each time, analogous to a set of hands passing the nucleotide chain between them[387,388]. This process has been well characterised for similar SF1B helicases, such as RecD2 which is thought to play a role in DNA repair[379]. Mapping this process to NSP13 gives an insight into the potential mechanism. It operates through the motions of 7 highly conserved helicase motifs, forming the key contacts which constitute the basic 'motor' for translocation[389]. The motifs can be separated into 3 subgroups depending on what substrates they interact with, as motifs I, II, and IV are mainly involved in ATP binding, motif IV exclusively binds RNA, and Ia, V, and III are involved in both[390]. Key motifs I and II constitute the Walker A and B motifs, key motifs seen in ATP binding sites across many different protein families[391]. These motifs, determined through a sequence alignment with RecD2 (PDB ID: 3GP8), are highlighted in Fig. (2.21).

The inchworm mechanism is a common theory in helicase literature, first proposed for SF1A helicase PcrA before being generalised to SF1B DNA helicases like RecD2 and now NSP13. While the fundamental concepts underpinning the mechanism is the same, changes in the residues involved with the process determine the specific directionality. In RecD2, ATP binding causes RNA-protein contacts to occur with the RecA2-analogous subdomain 2A motifs IV, V, and VI, to slide along the nucleotide chain. This causes RecA1-analogous subdomain 1A motif 3 to interact with motif 1a, adopting a closer conformation to generate a pocket around a central nucleotide which attracts its base and allows domain 1A to bind tightly.

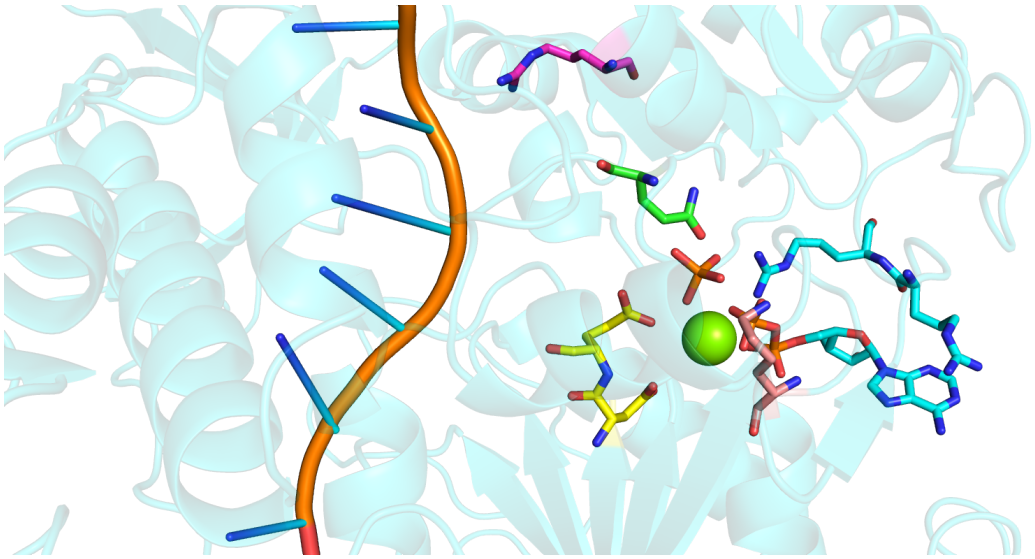


Figure 2.21: Conserved SF1B Motifs showing 5 of the conserved motifs, around the ATP-binding site with an MG atom, ADP, and H_2PO_2 molecule bound. Motifs I and Ia, II, III, IV, and V are coloured coded as blue, pink, green, magenta, and yellow. These residues interactions determine the shape, size, and functionality of the binding pocket, and the picture nearby RNA strand is translocated based on the deformation of the pocket by the binding process. Motif VI is not pictured as it lays further out the pocket and only participates in RNA binding.

Once hydrolysis is complete, the structure relaxes to its original conformation and one cycle is complete[392]. Due to the conserved nature of this superfamily, this purported mechanism can be directly mapped to NSP13 as current work supports[390,393]. A comparison between the conserved motifs is shown in Fig. (2.22).

It is important to note however, the inchworm mechanism attempts to explain the function of NSP13 seen as a functional monomer, and there is experimental evidence that NSP13 is much more active within the complex. This could be due to external forces caused by the presence of the RdRp and NSP7/8, as seen in SARS-CoV1 where this presence doubles the catalytic efficiency of its NSP13[394,395]. Other work in the area points to the importance of the stalk domain, which has been proposed to mediate interactions between the ZBD and the rest of the protein, which is necessary for NSP13 to function[396]. Therefore further investigation is needed before NSP13 can be confirmed as using the exact same mechanism as these other helicases.

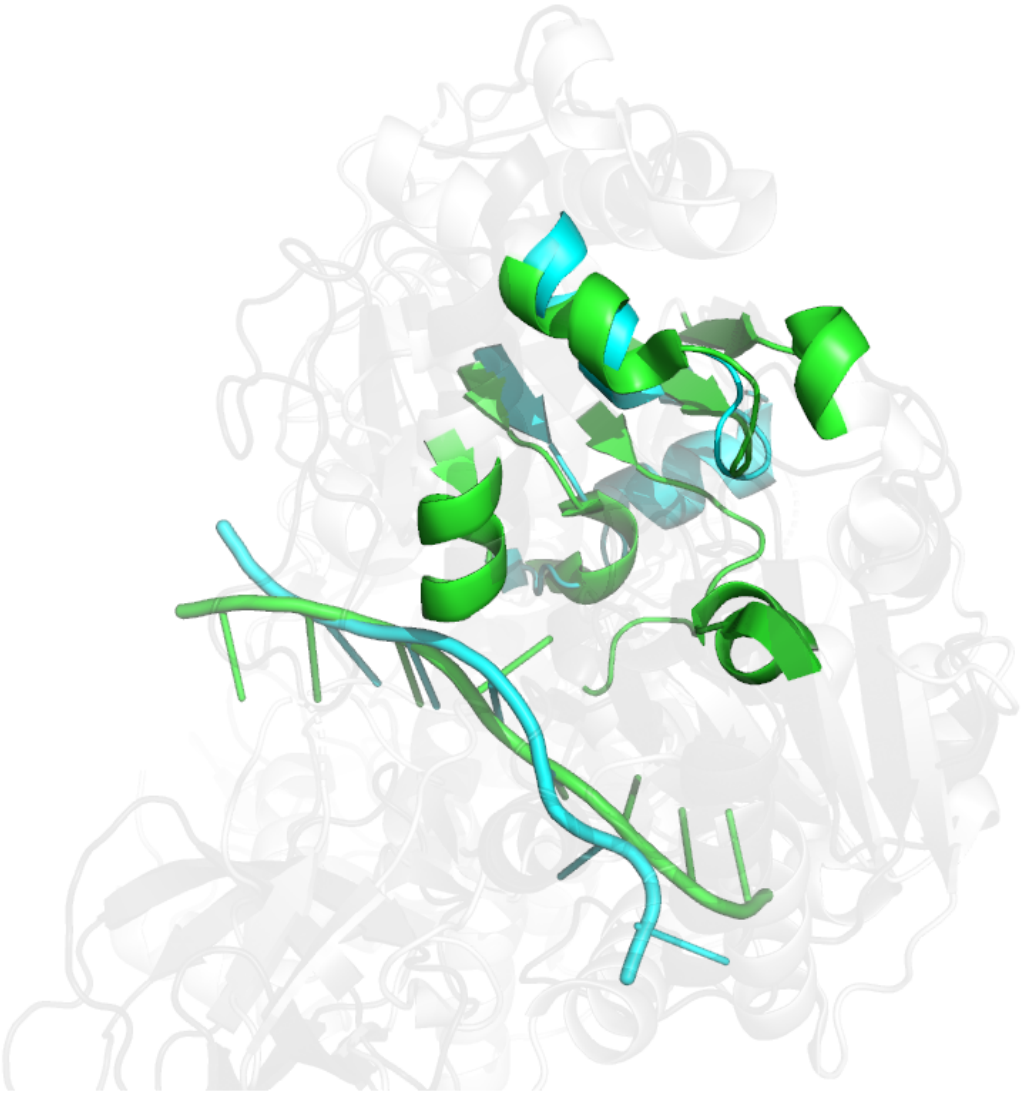


Figure 2.22: A comparison between motifs in structures for NSP13 (PDB: 6ZSL - Blue) and RecD2 (PDB: 3GP8 - Green), both type SF1B helicases. The overlap between them is clear, with an almost direct correspondence. Bound substrates are shown in both, RNA for NSP13 and DNA for RecD2. The structural similarity between the two is clear, validating considering a mechanically similar mechanism for NSP13.

This became the necessary justification for further studying the dynamical nature of NSP13 to further its potential as a drug target and elucidate its translocation mechanism. This work began early in the pandemic, and initially looked to provide a thorough modelling and kinetic analysis of the protein. Looking to provide an investigative basis for future drug design tasks, this work aimed to consolidate the relevant sequence and structural information available about the system. Following this by performing MD simulations to perform pocket and

kinetic analysis. Following the completion of this, the scope was expanded to cover the studying the dynamical nature of its mechanism. Aiming to simulate a translocation process using enhanced sampling MD, the initial modelling work provided a basis to test the existing theories surrounding the process.

2.4.3 Methodology

The first focus was on creating an accurate model of NSP13 bound with its natural substrates, ssRNA and ATP. This was developed by my colleagues from crystal structures via homology modelling, on data mainly on the SARS-CoV-1 helicase and the few existing experimental structures at the time. Additional sequence comparison was performed to find the homology of the sequence in comparison to other helicases. After this model was created, various MD simulations were performed using various force-fields to determine the natural behaviour of NSP13 in a monomeric state. For a thorough understanding across simulation techniques, 3 separate MD FF set-ups were used:

S.1 CHARMM36 combined with TIP3P water potential (CHARMM)[397]

S.2 Amber14SB for the protein, Amber ff99 + parmbsc0 + chioL3 for the ssRNA, ZAFF for the zincs, and TIP3P for water[398,399]

S.3 Amber14SB protein force field combined with ff99OL3 for the RNA, ZAFF for the zincs, and TIP3P for water[400].

Simulations were performed either with ATP and an ssRNA strand (holo) or with no cofactors (apo). Additionally simulations of NSP13 in a homogenous apo dimer were carried out in both **S.1** and **S.1** for $2\mu s$ each. This work was carried out by my collaborators, and further details on the methods used and other analyses performed on the system can be found in the corresponding paper and SI[401].

My contribution began with analysis of the resulting monomer MD simulations. Data from all performed runs were collected and separated depending on the FF setup used and cofactors bound, summarised in Table (2.2). This led to a large dataset consisting of $6\mu s$ of apo and $9.5\mu s$ of holo monomer simulations. A PCA was performed to assess the conformational changes observed in the monomer molecular dynamics simulations. Before the calculation, each protomer

Structure	Replica	Duration (μs)	FF Setup	MD Engine
Apo	1	1	**S.1**	NAMD
	2	1	**S.1**	NAMD
	3	1	**S.1**	NAMD
	1	1	**S.3**	Amber
	2	1	**S.3**	Amber
	3	1	**S.3**	Amber
Holo	1	1.5	**S.2**	GROMACS
	2	1.5	**S.2**	GROMACS
	3	0.5	**S.2**	GROMACS
	1	1	**S.1**	NAMD
	2	1	**S.1**	NAMD
	3	1	**S.1**	NAMD
	4	1	**S.1**	NAMD
	5	1	**S.1**	NAMD
	1	1	**S.3**	Amber

Table 2.2: Table displaying the amount of data for each simulation type performed on NSP13.

in every frame was aligned via RMSD minimization to a reference structure from the equilibrated holo model. The analysis was restricted to the C_α protein atoms to represent the major conformational changes observed and to reduce the dimensionality. Additionally the protein chain was truncated to limit the contribution of the long disordered tails of the protein seen to vary wildly in the monomer configuration. The PCA was performed on all apo and holo frames using the scikit-learn library[402,403] Weighted PCA-RMSD modes N_i were calculated to show the contribution of each residue to the corresponding PCA modes[298].

Dynamic cross-correlation (DCC) map analysis was also performed to determine inter-residue displacement correlations[295,296]. Calculations were performed between all C_α atom pairs in monomer simulations, producing a matrix for each simulation type. This matrix can be displayed as a 2D image, indicating which residues are displaced together and highlighting groups of residues that have a correlated motion and move as larger units. Additional pocket analysis was carried out by collaborators, measuring the volume of up to 7 distinct pockets identified in the simulations using a fork of the PyVol software[404]. To better understand the pocket formation process, correlations between measured pocket volumes and related inter-residue distances were calculated for the apo simulations. In

addition to the monomer simulations outlined, these calculations were additionally performed for the dimeric data. Defining a residue-residue distance as the shortest non-hydrogen distance between their constituent atoms, all such measurements that were below 4.5Å in 40-60% of frames were selected. For each of these distances **d**, PCC values between it and each pocket volume were calculated. Correlations were also calculated for PC coordinates.

Focus then moved onto attempting to simulate the translocation process. Determining the exact mechanism requires a full coverage of the hydrolysis process converting ATP to ADP + Pi, and the release of these products. For this, additional structures were built from the previously developed model to obtain a structure for each stage of the ATP-cleavage cycle: an ATP-bound model (ATP), an ADP-bound model both with (ADP+PO) and without (ADP) a H_2PO_4 in the binding site, and an apo model (Apo). All models featured a ssRNA strand consisting of 7 nucleotides of sequence $CCCAUGU$ consecutively numbered from $C7$ to $U12$, derived from the homology-modelled strand created in the previous work from crystal structures. The ATP binding pocket for each of these is shown in Fig. (2.23). The approach taken was to bias the backbone phosphates of the ssRNA, as these are in direct contact with much of the protein and are more importantly present in all nucleotides. Doing so would also avoid putting extra forces onto sidechains capable of rotating. Such a strategy would be easily generalised to differently coded strands, allowing for a wide a applicability in later use cases. As the work was to be performed in NAMD with FF setup **S.1**, preliminary work then focused on analysing the previously performed simulations in order to find the most frequent and stable phosphate-protein contacts. Defining distances similarly to how they were for calculating the PCCs, all simulations using this setup were analysed for their RNA-protein contact frequency f_{ij}. To calculate this, a contact was defined as a sub-4Å distance between any RNA-phosphate and heavy residue atom. Defining D_{ij} as an indicator function, taking a value 1 if any distance from residue i to RNA-phosphate j is below 4Å and 0 otherwise, this can be defined as:

$$f_{ij} = \frac{\sum^{\text{frames}} D_{ij}}{\sum^{\text{frames}} 1} \tag{2.4.3.1}$$

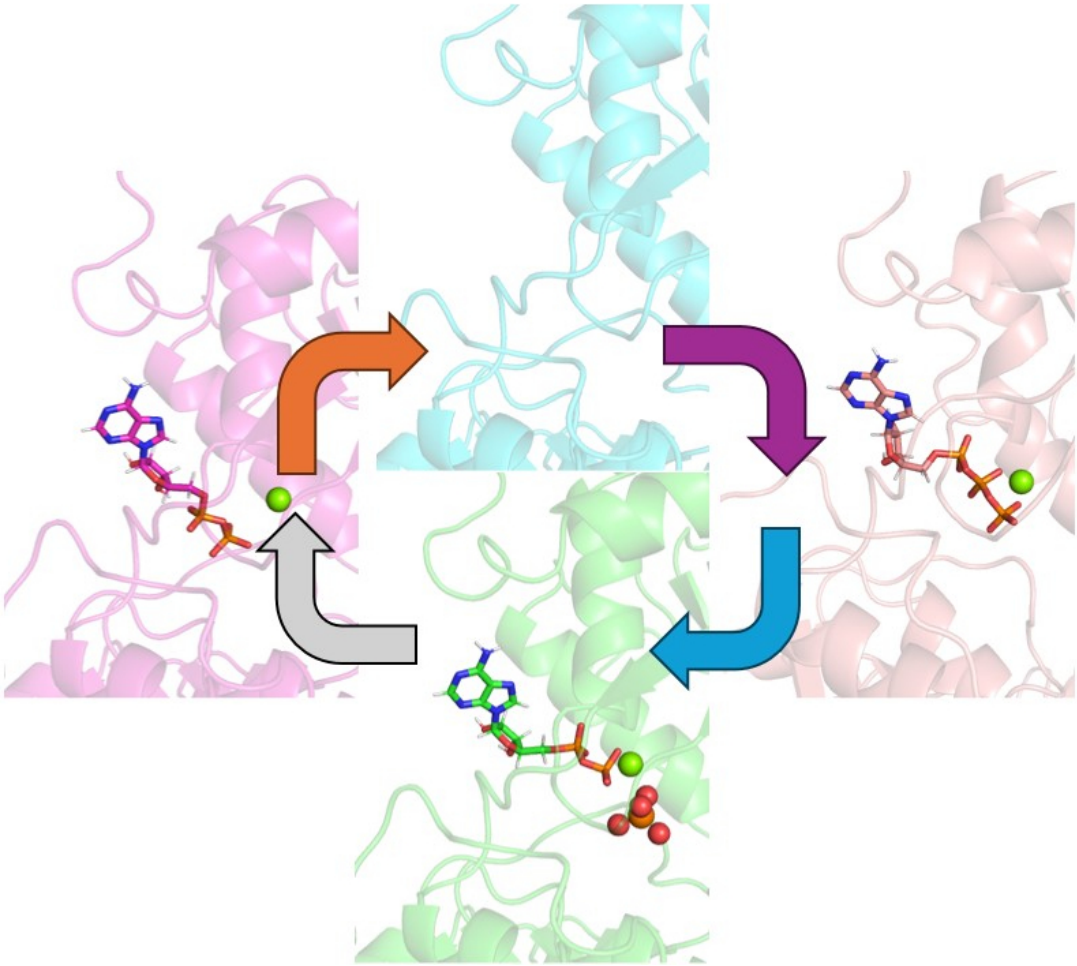


Figure 2.23: Models built for each stage of the NSP13 ATP cycle. Each model features a different set of bound substrates: Apo (top), ATP (right), ADP + PO (bottom), ADP (left).

Analysing the most frequent contacts for each j was used to inform the distance selection.

Once valid contacts had been identified, a simple biasing strategy was implemented using harmonic biases implemented in the NAMD colvars module[405,406]. To provide a valuable contribution to the discussion of backtracking and show the directionality of the helicase, the method was applied to both forwards and backwards translocation processes. The strategy employed was to bias the chosen protein contacts the phosphate next by supplementing the system with a harmonic potential energy well between them of the form:

$$V(v_1, \ldots, v_m) = \frac{1}{2}k\Big(\sum_{i=1}^{M}(v_i) - C\Big)^2 \qquad (2.4.3.2)$$

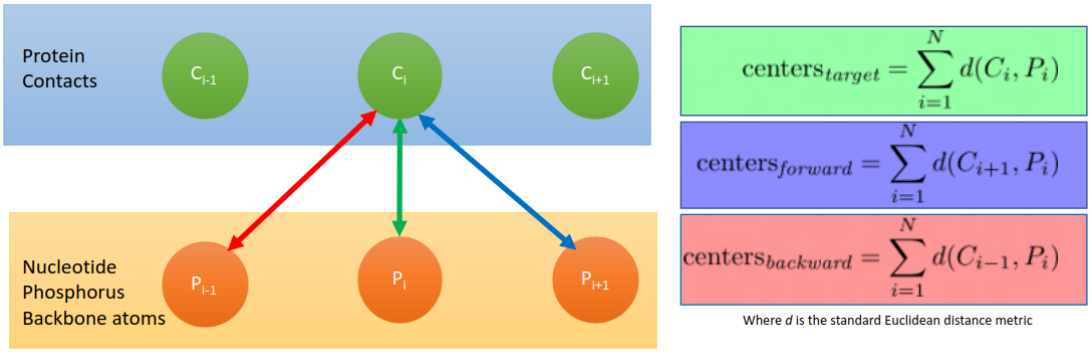


Figure 2.24: This diagram further explains the biasing system used on NSP13. The left side shows a simplified diagram of the atoms involved in the contacts, highlighting what is meant by forwards (blue) and backwards (red) translocation. The right hand side shows the equation for how the "centers" value is calculated. For example, in the forward biasing, this value is interpolated from $centers_{forward}$ *to* $centers_{target}$ *over a chosen number of steps, changing the centre of the added potential energy well over time.*

Where $v_1, .., v_m$ are the selected biasing distances, $V(\cdot)$ is the applied potential as a function of these distances, and k is the force constant. This term can be viewed as a simple addition to Eqn. (1.3.2.1) which forces the system towards a translocated state. This defines a 1-dimensional coordinate, defined as the sum of the m chosen distances, as the CV used. The value of C is the centre of the energy well defining the equilibrium value for the CV expressed within the sum. Adding the potential bias over a sum of distances rather than to each individual inter-atomic distance allows the system to have more freedom in the path it takes to reach the translocated state.

Single stable atom-phosphate contacts were identified for the 5 central RNA-phosphates, leaving an additional nucleotide at both the 3‘ and 5‘ ends. The summed distances of the contacts were averaged over the previous unbiased **S.1** simulation runs, providing the target value for C. To produce both forward and backward translocation processes, the chosen distances v_i were changed between two performed runs. In the forward case, v_i was the distance between i^{th} protein contact and the $(i+1)^{\text{th}}$ phosphate. For the backward, v_i was the distance between i^{th} protein contact and the $(i-1)^{\text{th}}$ phosphate. As the 3‘ end nucleic acid does not feature a phosphate, this meant the forwards bias would use 5 contacts while the backwards would only use 4. This is further is explained in Fig. (2.24).

Each simulation was equilibrated for 100ns, followed by a 500ns unbiased run. After the unbiased simulations had finished, two seperate translocation runs were performed for each model in each direction. The force constant of the bias was slowly increased over the next 100ns, up to a maximum of 15kcal/mol. The simulations were then run for a further 100ns without any biases to see if the translocated system was stable. Additionally to check if it was possible to identify the transition point, the translocated simulations conformations were sampled at 8 equidistant values on the CV. For each of these samples, and additional 10 unbiased simulations were run for 2ns to check if the conformation was stable.

All simulations were ran using the **S.1** set-up with the Langevin thermostat set to 300K and Langevin piston barostat set to 1atm. Both Langevin methods utilised a damping coefficient of $1ps^{-1}$, and the time-step was set to 2fs A rectangular unit cell was used, fitted to 10Å from the protein and solvated with TIP3P model water molecules. Ions were placed to neutralise charge in the system, varying with the charge of substrates bound. PBCs were used with the PME method for long-range electrostatics. Production run MD was preceded by 1000 steps of minimisation using the steepest descent algorithm followed by 50ns of NVT equilibration.

Early simulation tests showed the ZBD struggled to keep the tetrahedral coordination required, frequently allowing waters to access the binding site and unnaturally coordinating the zincs. While this problem was initially faced in the earlier simulations, and remedied using additional harmonic restraints on the zinc during equilibration, this was not an ideal solution. To fix this, the default CHARMM36 zinc FF parameters were supplemented to that of the ZAFF zinc force field as implemented in Amber14SB[407]. Additionally to maintain the realism of the simulation several new crystallographic waters observed in models released since the original work were added. These waters were from PDB IDs 6ZSL, 6JYT and 7ZSL[380,408,409].

2.4.4 Results & Discussion

Across all initial simulations analysed the first four PCA components accounted for greater than 80% of the observed variance. Results shown in Fig. (2.25) provide a direct comparison between the apo and holo monomer simulations. No large scale domain changes were observed separating the two types of simulation. The weighted RMSD modes for the first four PCA components show the residue displacements captured by each component for the apo and holo simulations. Increased flexibility in the RecA2 domain is the most clear and significant difference present between the apo and holo data. Residues around the RecA1 interface and ATP binding site, most notably in the outermost loop from THR450 to ALA510, are more flexible in the absence of ATP shown by the magnitude of the associated weighted RMSD mode. This observed flexibility is also consistent with the increased experimental B-factors of this region for the 6JYT and 6ZSL structures. The presence of ATP in the holo simulations stabilizes many of the key residues involved in binding along with the whole of the RecA2 domain, indicated by the decreased weighted RMSD in the holo simulations and thus smaller contribution to all PCA components.

A clear difference in the behaviour of the structures between holo and apo is the observed flexibility in the loop region from ILE334 to GLN354. This difference is shown both from the PCA decomposition and the DCC map, where it is seen to negatively correlate with the rest of RecA1. Positioned between key RNA and ATP residues, this motion could be indicative of a more significant interaction with the ATP binding site. The DCC map highlights the proteins tendency to oscillate as 3 distinct regions. This is more pronounced in the holo simulations as shown in Fig. (2.25)(e), showing the introduction of substrates stabilises several of the loop regions displaying a negative correlation in apo sims. The greater stability of the holo loops was observed across both FFs, as shown in the appendix Fig. (2.32). These maps also show the differences between the FFs, displaying the tendency of set-up **S.1** to have significantly more uncorrelated motions across loop-like regions.

The PyVol analysis identified 7 pockets in the holo protein. These were the ATP site, RNA pocket, 3 RecA2 pockets, a ZBD domain pocket, and a novel

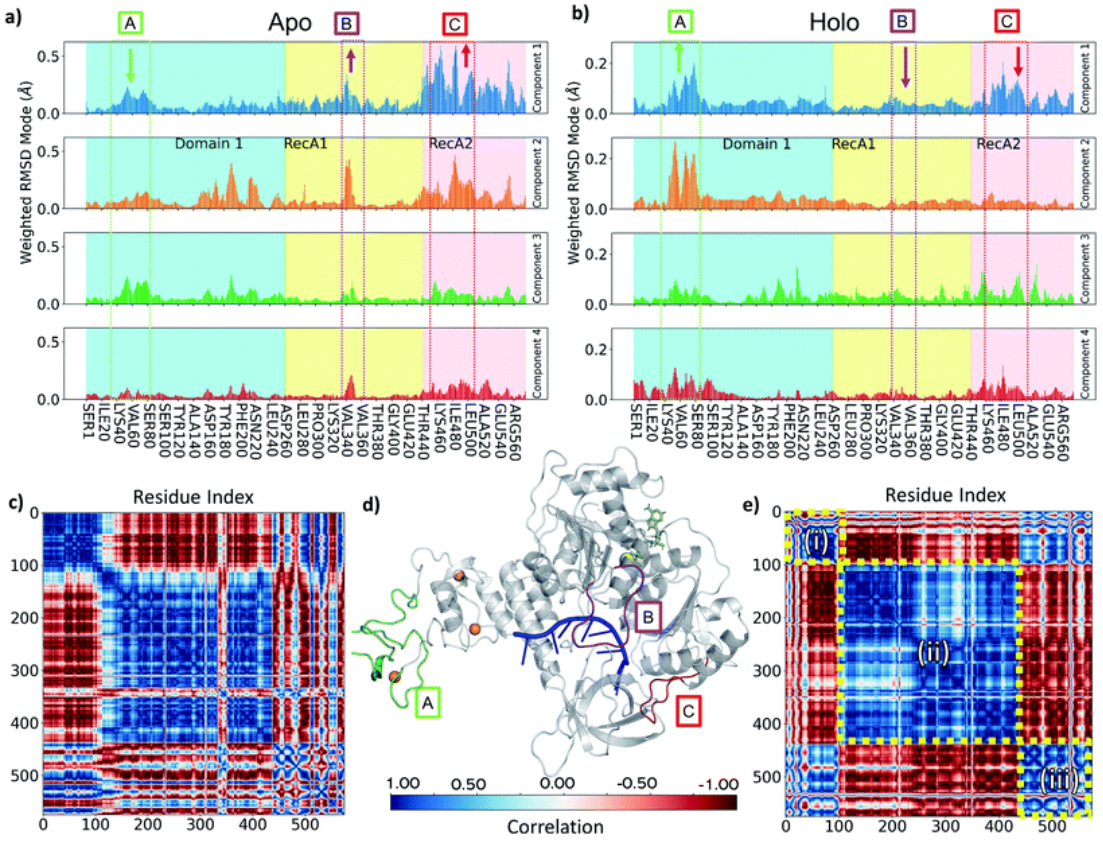


Figure 2.25: ***(a)*** *and* ***(b)****: Comparison of PCA Weighted-RMSD scores holo monomer* ***(a)*** *and apo monomer* ***(b)*** *simulations with Amber.* ***(c)****: DCC map of holo monomer simulation with Amber, over the motion of the first PCA component.* ***(d)****: Key areas highlighted in the helicase structure (d, grey cartoon) and on panels a and b (text box and arrows indicating relative change in magnitude): (A) Zinc Binding region (green), (B) Domain 1 loop from ILE334 to GLN354 (purple), (C) RecA1/2 interface loop from THR450 to ALA510 (red). Bound objects are shown and coloured: RNA in blue (centre), ATP in lime (upper right), zinc in orange (left spheres) and magnesium as a yellow sphere.* ***(e)*** *DCC map of apo monomer simulation with Amber for motions described by the first principal component. A colour bar providing the correlation scale is shown in the bottom centre.*

pocket in the stalk region present in all simulations. Not all pockets were found in all simulations, with the CHARMM monomer being the only system to display all. The identified pockets are shown on the CHARMM monomer in Fig. (2.26). This is possibly caused by the aforementioned observed increased flexibility across loop-like regions in this FF. Pocket-distance correlations are shown in the appendix for dimeric simulations Fig. (2.34) and monomers in Fig. (2.35). Additional work was performed by colleagues to calculate FEPs over the most correlated distances

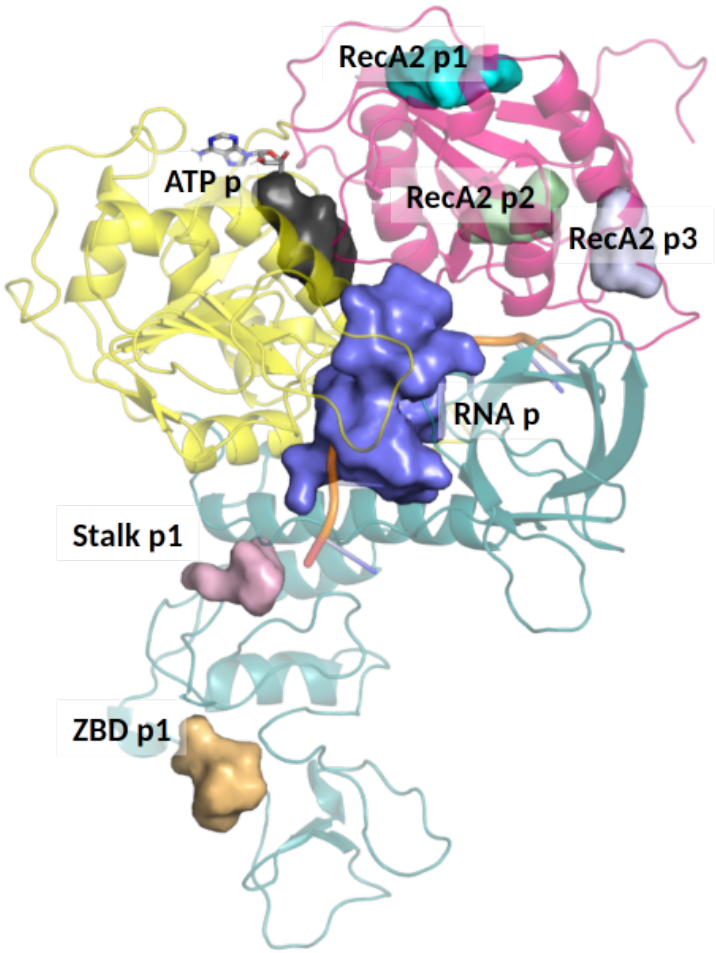


Figure 2.26: A visualisation of pockets identified using PyVol in the NSP13 simulations. All pockets are shown on the CHARMM monomer model, with substrates hidden.

found, and are featured in the final publication[401]. The stalk pocket would later serve as a ligand binding target in Case Study 4.

Contact analysis between the RNA-phosphates and protein for the holo simulations are shown in Fig. (2.27). The most frequent contacts from this were used as a basis to decide which distances to bias as shown in Fig. (2.28). For each nucleotide, the most frequent contacts were analysed by hand for suitability, as any loop like regions would not prove stable enough for biasing and so were replaced for nearby stable secondary structures. Studies of the RNA binding site had already identified two phosphate-protein H-bonds between THR359 in RecA1 and THR532 in RecA2, which both showed as highly frequent contacts

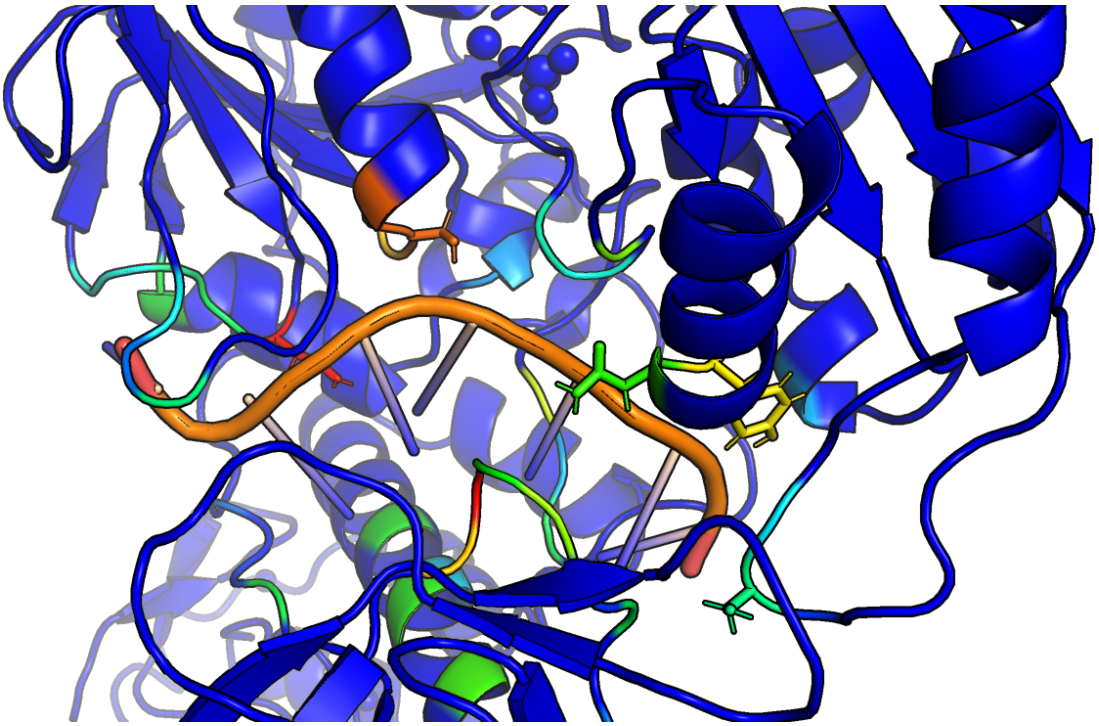


Figure 2.27: Holo model with ssRNA bound. showing the observed RNA-backbone phosphate contacts observed in holo CHARMM simulations. Contacts are coloured by the frequency of the contact calculated across all frames. This is from low to high: blue-cyan-green-yellow-orange-red.

in the subsequent analysis. Interestingly, some of the chosen contacts overlap with the RecD2-aligned functional helicase motifs. The H311 contact is in Motif 1A (residues 308-311) and contact T359 is close structurally to Motif 2 (residues 372-377), both of which additionally form part of the ATP binding site. This provided confirmation for the chosen strategy, as the biasing protocol could potentially mimic some of the associated conformational changes of this motifs that occurs during hydrolysis. It also validates the approach of mapping the RecD2 mechanism, as some of the key residues are identified to also play a critical role in stabilising the RNA in NSP13.

The biasing strategy was first performed on the ADP + PO, ADP, and Apo models in both the forwards and backwards directions, as these were the newly developed systems. All models displayed a numerical indication that the contacts had substantially shifted for both directions, as if translocation was occurring as shown in Fig. (2.29). Forward simulations displayed an incredibly quick rate of translocation, with even the smallest bias seemingly enough to significantly alter

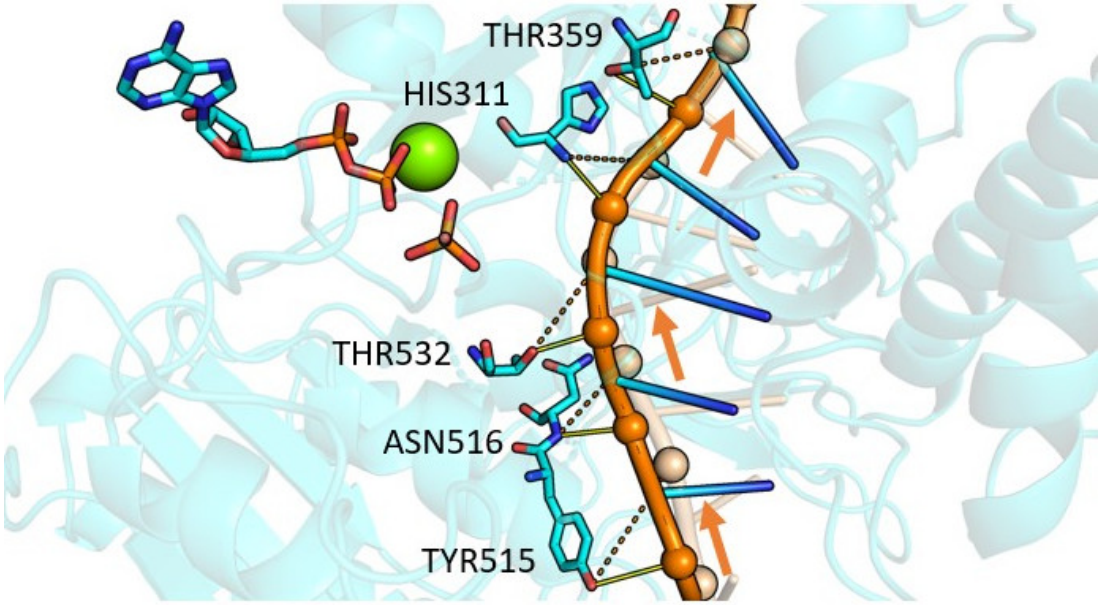


Figure 2.28: Figure showing initial contacts identified as biasing candidates to force RNA translocation. Residue chose are labelled, along with a solid line indicating the usual RNA-phosphate contact,and a dashed line to indicate the target phosphate it will be biased towards. Arrows denote the direction of translocation, and the transparent RNA strand shows its potential location after translocation. ADP, a magnesium atom, and H_2PO_4 molecule are shown in the binding pocket.

the observed distances. To make this process slower, an additional forward run was performed with a force constant interpolating to a maximum of 3kcal/mol. The results of this weaker forward simulation and the previous backward were sampled to test the stability and see if the transition state could be found. However the additional simulations ran from the equidistant conformation samples showed a tendency in both directionalities to return back to the original state, implying this transition was unstable and supported by just the biasing forces. The results for these are shown in Fig. (2.30) for the apo model simulations. No further unbiased simulation showed any indication of stability in the biased state, with all evaluated gradients of the CV returning to the original position.

While the numerical results initially looked promising, the evidence pointed towards an unrealistic simulated translocation process. Close visual inspection revealed several issues with both directionalities. The ends of the RNA showed clear unnatural conformations, with both ends folding back into the RNA binding cleft. In the backward simulation this occurred with the 5‘-end. Here it forced

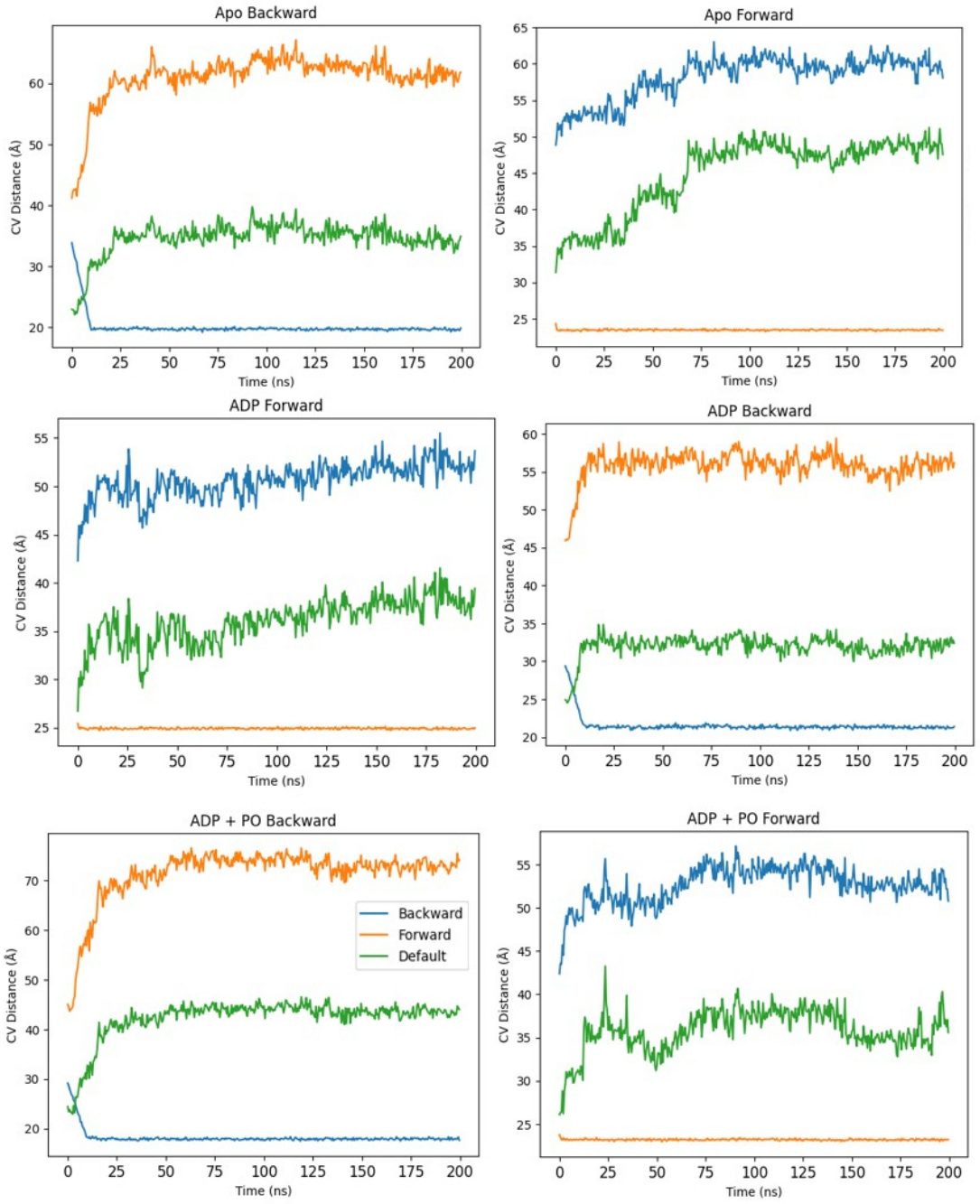


Figure 2.29: A plot showing the CV distances from all 3 simulation types ran with a force constant of 15kcal/mol forwards and backwards. In each instance the value of the forward, backward, and standard RNA- phosphate-protein contacts are shown.

the ssRNA into a clearly unnatural conformation, while stabilising the position through several $\pi-\pi$ stacking interactions with nearby residue rings and sometimes the U nucleotide itself. An example of the interaction is shown in the appendix in Fig. (2.36). While this formed a more stable conformation when held in place by

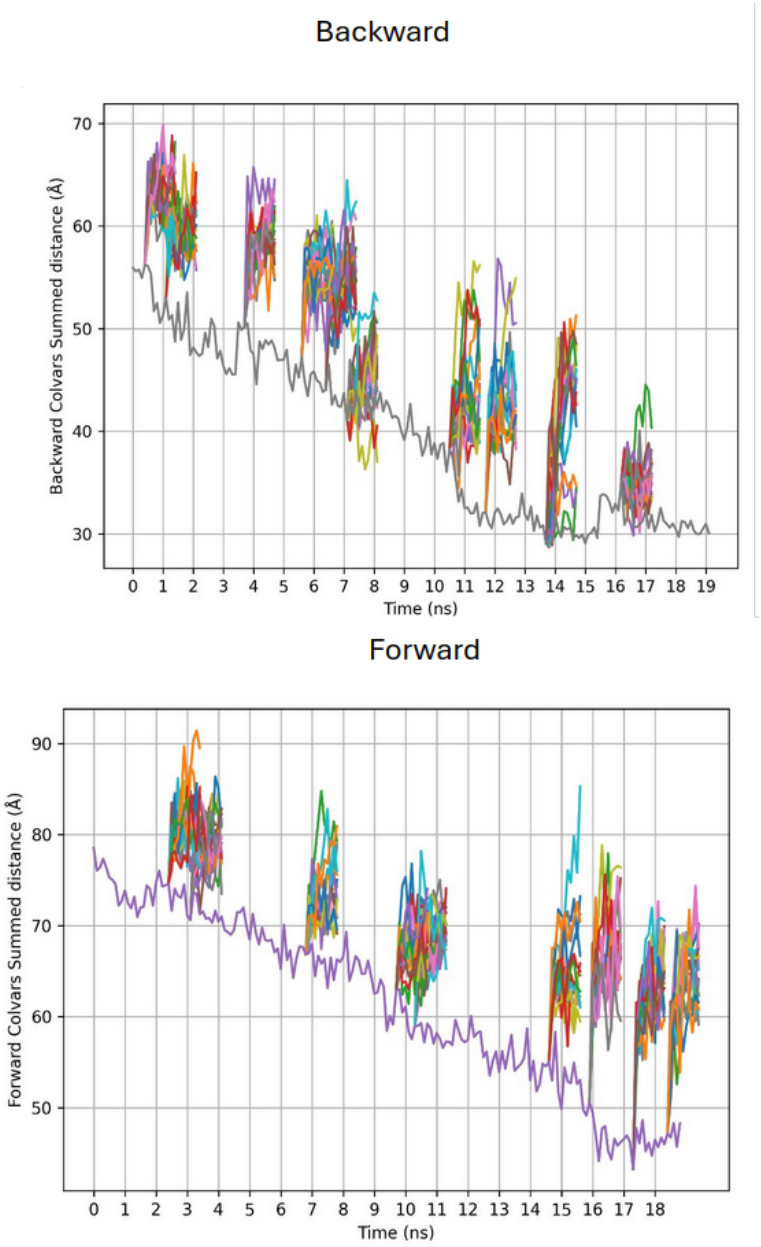


Figure 2.30: A plot of the combined contact distances over time during a biased Apo model NSP13 simulations. The upper plot shows the results from the backward simulations, and the lower from the forward. Additional lines come from the additional 2ns simulations ran from the 8 interpolated values of the CV, showing some stability in the backward simulations but none in the forward.

the biasing forces, the naturally unfavoured RNA backbone rotations rescinded to the expected, non-translocated positions once the bias was removed. As the ASN516:$C7$ was a measured distance in these backwards runs, this compression and rotation of the 3‘-end led to a much shorter value for the collective variable, and was responsible for much of the shortening of the CV seen.

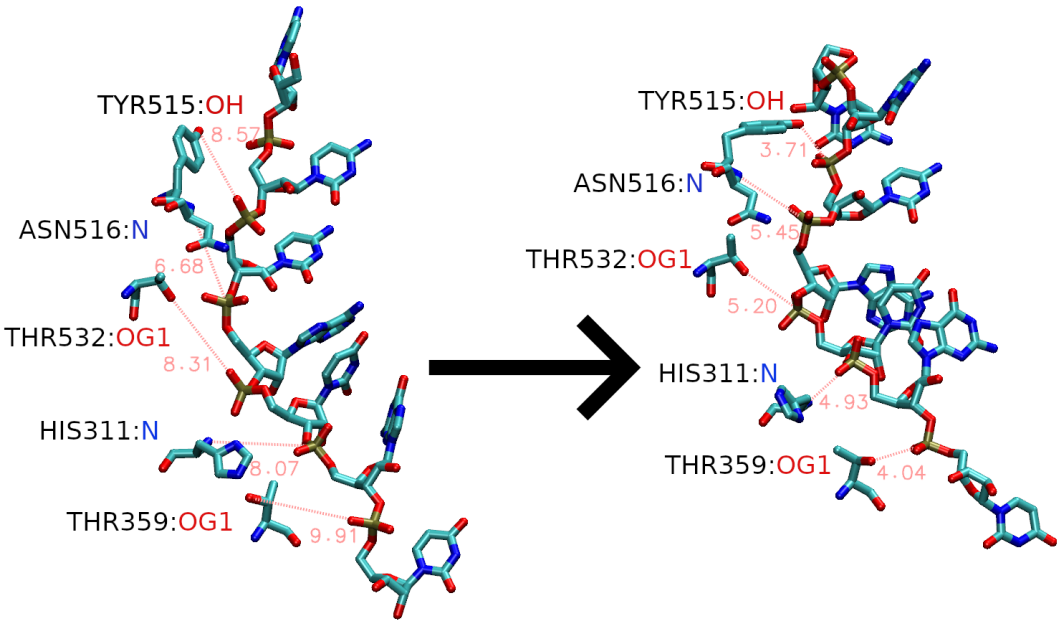


Figure 2.31: Structural comparison from the forwards biased apo simulations using key contacts identified from previous simulations. On the left is the initial conformation, and the right the final conformation. The rotation of TYR515 and looping of the final RNA phosphate are clearly seen.

In the forward simulations, a similar behaviour was observed by the 3‘-end of the ssRNA. Even with the smallest biasing force, the backbone phosphate seemed to fully rotate and direct $U12$ back towards the RNA-cleft. This was accompanied by a significant rotation of the TYR515 contact to lay outside its usual orientation and move nearer the bound RNA. As this backbone angle has a significant impact on both the TYR515-$C8$ and ASN516-$C9$ contacts, it took a significant portion of the supplemented biasing force and its rotation was significantly effected. Alongside these, the movement of the RNA in the forward direction was accompanied by a significant shift distance between both RecA domains and Domain 1B, as it increased to accommodate the folding of the RNA chain. This is shown in appendix Fig. (2.37), which shows the distance change between both domains and 1B during the introduction of the biasing forces. This breakage of the core protein structure is considered to be not mechanically relevant due to its instability, highlighting this is more likely an issue with the simulations rather than a functional conformational change. A visual inspection

for the forward case shown in Fig. (2.31), highlighting how the biasing results in the head of TYR515 rotating into the main cleft to cause the domains to separate.

2.4.5 Conclusion & Future Work

Unfortunately, despite numerous attempts to circumvent these issues the translocation process was never fully observed. Several strategies were tested- including increasing the RNA chain by 2 and 4 nucleotides to avoid the end interactions, reversing the forces to push the RNA rather than pull, and removing some of the problematic contacts. However none could avoid the simulation artefacts produced by the end nucleotides. No matter the strength of the bias, they consistently either interacted with the residues outside the pocket, or further increased the separation between RecA1/2 and Domain 1B. This resulted in an unphysical conformation, and so provided an unsatisfactory final result.

In order to effectively simulate the translocation process, and much more thorough and focused approach is required, as the process is significantly more complex than simply pulling the RNA chain through the central cleft. This motion does nothing to simulate the actual effects of ATP-binding which drives this process. If this study were to be carried forwards, I would instead focus more on the mechanical consequences of the ATP-dependent process on the identified motifs rather than viewing them as something to analysed after the simulation is performed. If anything the attained result is mechanistically consistent with the inchworm mechanism, as in the default state the ssRNA chain is held tightly such that it compresses rather than translocates when put under force. A better understanding of the movement of these motifs is necessary before adopting a biasing strategy, as I feel the one used here was too simple for such a complicated biological procedure. Using the structures generated here to instead perform long simulations to understand the effects of the change in substrate would be the next step, and I hope these results can provide some insight into how it could be possible to later observe helicase translocation.

Despite these issues with the latter project, the first work provided some key kinetic analysis results for NSP13 at a time where there were few studies available. In particular, the pocket analysis stands out as it was later utilised in

Case Study 4, highlighting the value of this work and similar studies to the drug discovery process. As side from this, the study was able to categorise some of the key differences between apo and holo simulations. While many of these results were as expected, such as substrate binding stabilising many of the loops, it is regardless useful to have these confirmed with long simulations.

This work will hopefully provide a useful basis for further drug discovery tasks and work on simulating translocation perhaps across different helicases. The prevalence of RNA-contacts with the already identified functional SF1B helicase motifs points towards this process being highly conserved across helicases. These critical proteins provide a useful target for many viral targets, and future work could elucidate this mechanism further.

2.4.6 Appendix

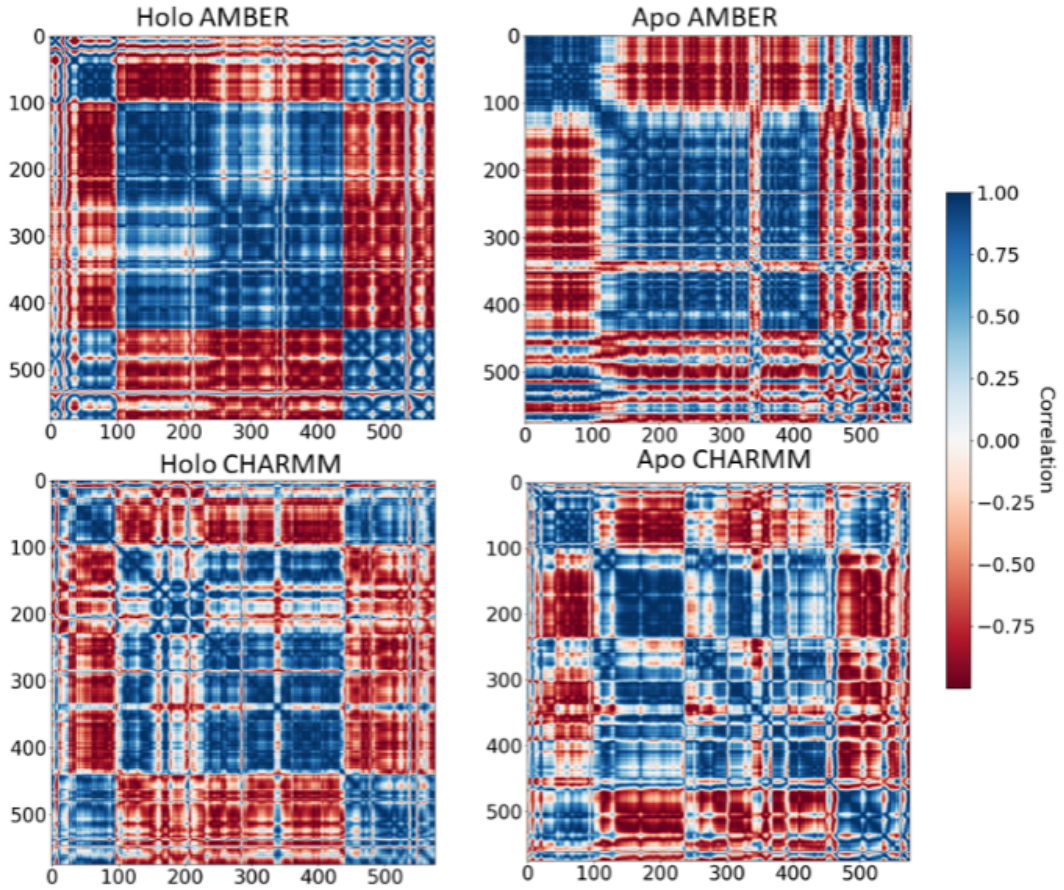


Figure 2.32: All DCC maps calculated for NSP13 monomer simulations, separated by FF used and substrate binding state. All show a similar pattern of 3 clear regions, although all loop-like regions show more variation in CHARMM models.

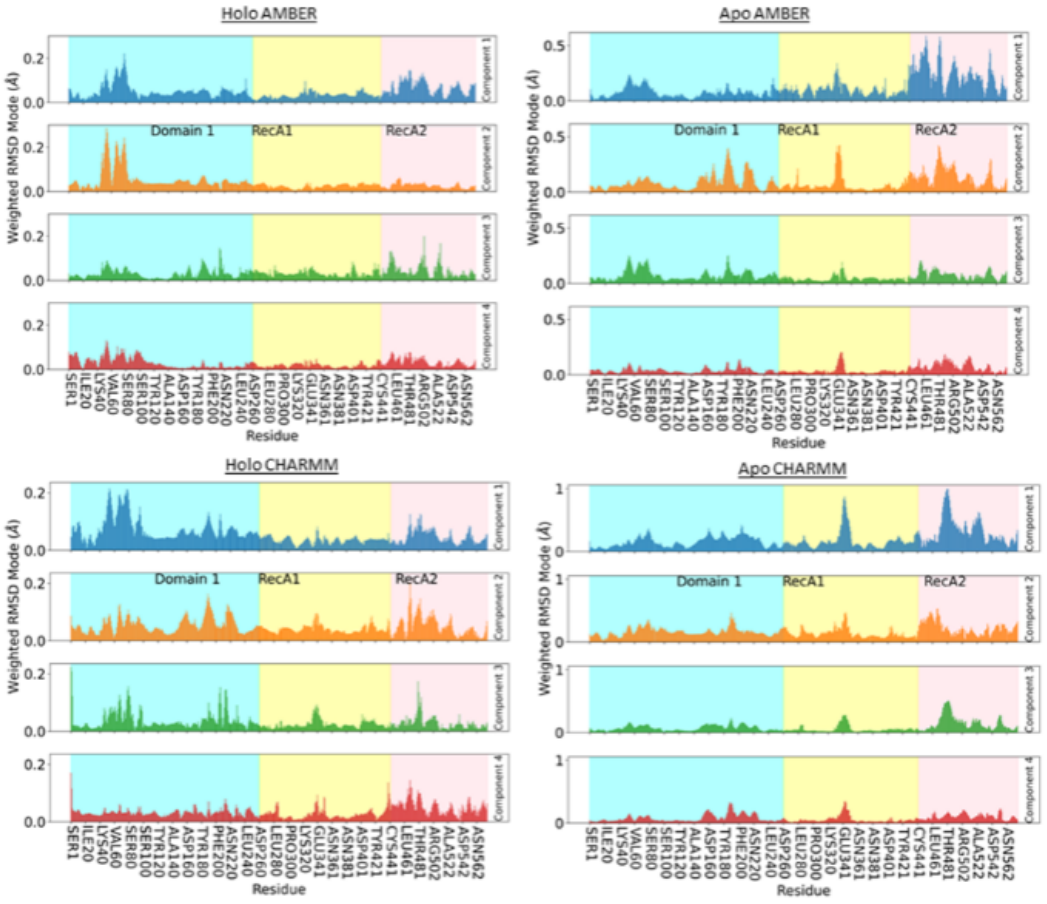
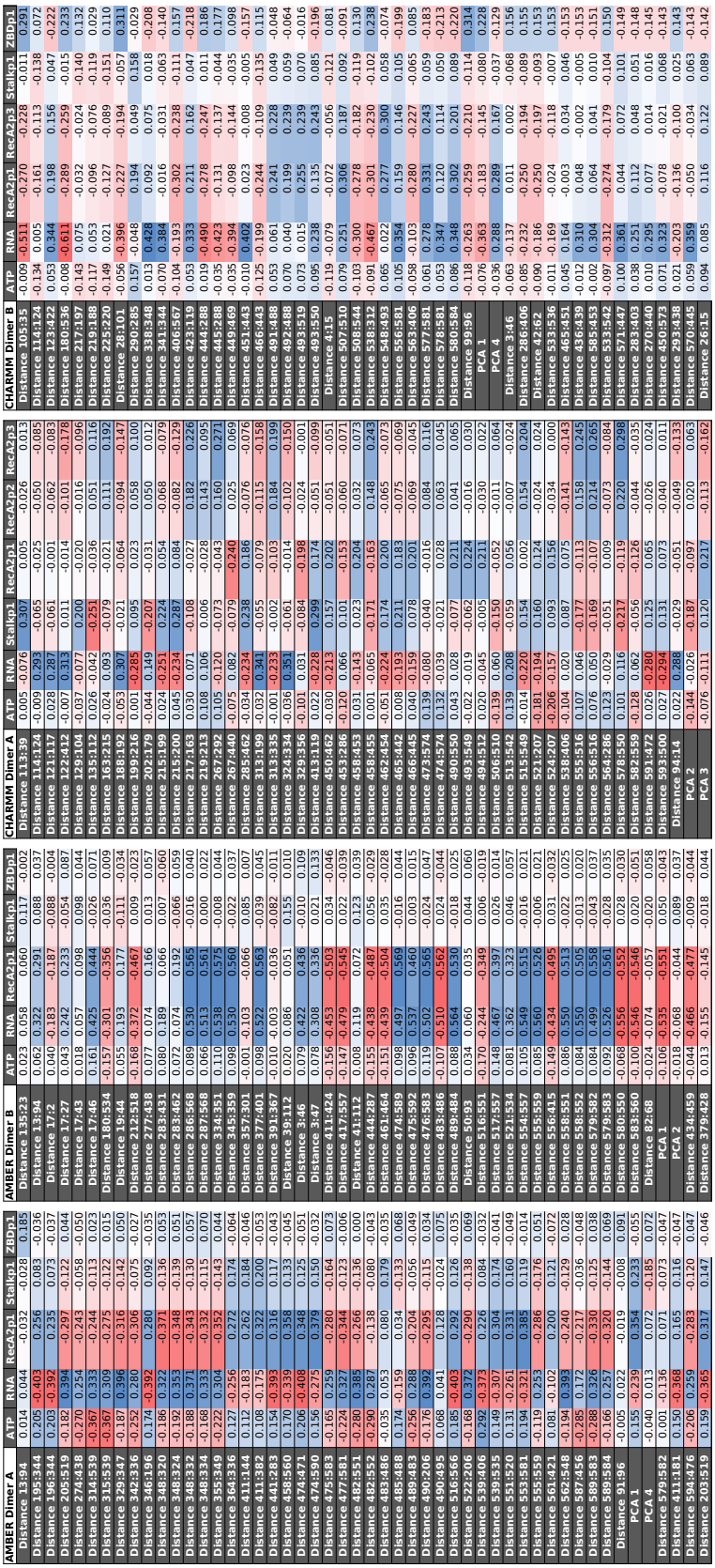


Figure 2.33: The PCA RMSD modes calculated for the first 4 PCs across all monomer NSP13 simulations. Motions are generally consistent between FF types, although holo CHARMM simulations show a significant variation in the ZBD.



AMBER Dimer A	ATP	RNA	Rec42p1	Stalk2p1	ZDP2p1
Distance 19:94	0.014	0.044	-0.032	-0.028	0.185
Distance 195:344	0.205	-0.403	0.256	0.083	-0.036
Distance 196:344	0.203	-0.392	0.235	0.073	-0.037
Distance 205:519	-0.182	0.394	-0.297	-0.122	0.044
Distance 274:438	-0.270	0.254	-0.243	-0.058	-0.050
Distance 314:539	-0.367	0.333	-0.244	-0.113	0.023
Distance 315:539	-0.367	0.309	-0.275	-0.122	0.015
Distance 329:347	-0.187	0.396	-0.316	-0.142	0.050
Distance 342:336	-0.252	0.280	-0.306	-0.075	-0.027
Distance 346:196	0.174	-0.392	0.280	0.092	-0.035
Distance 348:320	-0.186	0.322	-0.371	-0.136	0.053
Distance 348:324	-0.192	0.353	-0.348	-0.139	0.051
Distance 348:332	-0.188	0.371	-0.343	-0.130	0.057
Distance 348:334	-0.168	0.333	-0.332	-0.115	0.070
Distance 355:349	-0.222	0.304	-0.352	-0.143	0.044
Distance 364:336	0.127	-0.256	0.272	0.174	-0.064
Distance 411:144	0.122	-0.183	0.262	0.184	-0.046
Distance 411:382	0.108	-0.175	0.322	0.200	-0.053
Distance 441:283	0.154	-0.393	0.316	0.117	-0.043
Distance 458:560	0.170	-0.339	0.358	0.133	-0.045
Distance 478:471	0.206	-0.408	0.348	0.125	-0.051
Distance 474:590	0.156	-0.275	0.379	0.150	-0.032
Distance 475:583	-0.165	0.259	-0.280	-0.164	0.073
Distance 477:581	-0.224	0.327	-0.344	-0.123	-0.006
Distance 482:551	-0.280	0.385	-0.266	-0.136	0.000
Distance 482:552	-0.290	0.287	-0.138	-0.080	-0.043
Distance 483:486	-0.035	0.053	0.080	0.179	-0.035
Distance 485:488	0.174	-0.159	0.034	-0.133	0.068
Distance 489:483	-0.256	0.288	-0.204	-0.056	-0.049
Distance 490:206	-0.176	0.392	-0.295	-0.115	0.034
Distance 490:495	0.068	0.041	0.128	-0.024	0.075
Distance 516:566	0.185	-0.403	0.292	0.126	-0.035
Distance 522:206	-0.168	0.372	-0.290	-0.138	0.069
Distance 539:406	0.292	-0.373	0.226	0.084	-0.032
Distance 539:535	0.149	-0.307	0.304	0.174	-0.041
Distance 551:520	0.131	-0.261	0.331	0.160	-0.049
Distance 553:581	0.194	-0.321	0.385	0.119	-0.014
Distance 555:559	-0.119	0.253	-0.286	-0.176	0.051
Distance 561:421	0.081	-0.102	0.200	0.121	-0.072
Distance 562:548	-0.194	0.393	-0.240	-0.129	0.028
Distance 587:456	-0.285	0.172	-0.217	-0.036	-0.048
Distance 589:583	-0.288	0.326	-0.330	-0.125	0.038
Distance 589:584	-0.166	0.257	-0.320	-0.144	0.069
Distance 91:96	-0.005	0.022	-0.019	-0.008	0.091
PCA 1	0.155	-0.239	0.354	0.233	-0.055
PCA 4	-0.040	0.013	0.072	-0.185	0.072
Distance 579:582	0.001	-0.136	0.071	-0.073	-0.047
Distance 411:181	0.150	-0.368	0.165	0.116	-0.047
Distance 594:476	-0.206	0.259	-0.283	-0.120	0.047
Distance 203:519	0.159	-0.365	0.317	0.147	-0.046

AMBER Dimer B	ATP	RNA	Rec42p1	Stalk2p1	ZDP2p1
Distance 135:323	0.023	0.058	0.060	0.117	-0.002
Distance 13:94	0.062	0.322	0.291	0.088	0.037
Distance 17:2	0.040	-0.183	-0.187	-0.088	-0.004
Distance 17:27	0.043	0.242	0.233	-0.054	0.087
Distance 17:43	0.018	0.057	0.098	0.098	0.044
Distance 17:46	0.161	0.425	0.444	-0.026	0.071
Distance 180:534	-0.157	-0.301	-0.356	-0.036	0.009
Distance 19:44	0.055	0.193	0.177	-0.111	-0.034
Distance 212:518	-0.168	-0.372	-0.467	0.009	-0.023
Distance 277:438	0.077	0.074	0.166	0.013	0.057
Distance 283:431	0.080	0.189	0.066	0.007	-0.060
Distance 283:462	0.072	0.074	0.192	-0.066	0.059
Distance 286:568	0.089	0.530	0.565	-0.016	0.040
Distance 287:568	0.066	0.513	0.561	0.000	0.022
Distance 334:351	0.110	0.538	0.575	-0.008	0.044
Distance 345:359	0.098	0.530	0.569	-0.022	0.037
Distance 357:301	0.098	0.522	0.563	-0.039	0.045
Distance 377:401	-0.010	-0.003	-0.036	-0.082	-0.011
Distance 39:112	0.020	0.086	0.051	0.155	0.010
Distance 3:46	0.079	0.422	0.436	-0.010	0.109
Distance 3:47	0.078	0.308	0.336	0.021	0.133
Distance 411:424	-0.156	-0.453	-0.503	0.034	-0.046
Distance 417:557	-0.147	-0.479	-0.545	0.022	-0.039
Distance 41:112	0.008	0.119	0.072	0.123	0.039
Distance 444:287	-0.155	-0.438	-0.487	0.056	-0.029
Distance 461:464	-0.151	-0.439	-0.504	0.035	-0.028
Distance 474:589	0.098	0.497	0.569	-0.016	0.044
Distance 475:592	0.096	0.537	0.460	0.003	0.015
Distance 476:583	0.119	0.502	0.565	-0.024	0.047
Distance 483:486	-0.107	-0.510	-0.562	0.024	-0.044
Distance 489:484	0.088	0.564	0.530	-0.018	0.025
Distance 50:93	0.034	0.060	0.035	0.044	0.060
Distance 516:551	-0.170	-0.244	-0.349	0.006	-0.019
Distance 517:557	0.148	0.467	0.397	0.026	0.014
Distance 521:534	0.081	0.362	0.323	0.046	0.057
Distance 554:557	0.105	0.549	0.515	-0.016	0.021
Distance 555:559	0.085	0.560	0.526	-0.006	0.021
Distance 566:415	-0.149	-0.434	-0.495	0.031	-0.032
Distance 568:551	0.086	0.550	0.513	-0.022	0.025
Distance 568:552	0.084	0.550	0.505	-0.013	0.020
Distance 579:582	0.084	0.499	0.558	-0.043	0.037
Distance 579:583	0.092	0.526	0.561	-0.028	0.035
Distance 580:550	-0.068	-0.556	-0.552	0.028	-0.030
Distance 583:560	-0.100	-0.546	-0.546	0.020	-0.051
Distance 82:68	-0.024	-0.074	-0.057	-0.020	0.058
PCA 1	-0.106	-0.535	-0.551	0.050	-0.043
PCA 2	0.018	-0.068	-0.044	0.089	0.037
Distance 434:459	-0.044	-0.466	-0.477	-0.009	0.044
Distance 379:428	0.013	-0.155	-0.145	-0.018	0.044

CHARMM Dimer A	ATP	RNA	Rec42p1	Stalk2p1	Rec42p2	Rec42p3
Distance 113:59	0.005	-0.076	0.307	0.005	-0.026	0.013
Distance 114:124	-0.009	0.293	-0.065	-0.025	-0.050	-0.085
Distance 121:117	0.028	0.287	-0.061	-0.001	-0.062	-0.083
Distance 122:412	-0.007	0.313	0.011	-0.014	-0.101	-0.178
Distance 129:104	-0.037	-0.077	0.200	-0.020	-0.016	-0.096
Distance 135:112	0.026	-0.042	-0.251	-0.036	0.051	0.116
Distance 163:215	-0.024	0.093	-0.079	-0.021	0.111	0.192
Distance 188:192	-0.053	0.307	-0.021	-0.064	-0.094	-0.147
Distance 199:216	0.001	-0.285	0.095	0.023	0.058	0.100
Distance 202:179	-0.044	0.149	-0.207	-0.031	0.050	0.012
Distance 215:199	0.024	-0.251	0.224	0.054	-0.068	-0.079
Distance 215:200	0.045	-0.234	0.287	0.084	-0.082	-0.129
Distance 217:163	0.030	0.071	-0.108	-0.027	0.182	0.226
Distance 219:213	0.108	0.106	0.006	-0.028	0.143	0.095
Distance 267:292	0.105	-0.120	-0.073	-0.043	0.160	0.271
Distance 267:440	-0.075	0.082	-0.079	-0.240	0.025	0.069
Distance 285:462	-0.034	-0.234	0.238	0.186	-0.076	-0.076
Distance 313:199	-0.032	0.341	-0.055	-0.079	-0.115	-0.158
Distance 313:335	-0.001	-0.233	-0.002	-0.103	0.184	0.199
Distance 324:334	-0.036	0.351	-0.061	-0.014	-0.102	-0.150
Distance 329:356	-0.101	0.031	-0.084	-0.198	-0.024	-0.001
Distance 413:319	0.022	-0.228	0.299	0.174	-0.051	-0.099
Distance 450:462	-0.030	-0.213	0.157	0.202	-0.051	-0.051
Distance 453:286	-0.120	0.066	0.101	-0.153	-0.060	-0.071
Distance 458:453	0.031	-0.143	0.023	0.204	0.032	0.073
Distance 458:455	0.001	-0.065	-0.171	-0.163	0.148	0.243
Distance 462:454	-0.051	-0.224	0.174	0.200	-0.065	-0.073
Distance 465:442	0.008	-0.193	0.211	0.183	-0.075	-0.069
Distance 466:445	0.040	-0.159	0.078	0.201	-0.069	-0.045
Distance 473:574	0.139	-0.080	-0.040	-0.016	0.084	0.116
Distance 474:574	0.132	-0.039	-0.021	0.028	0.063	0.045
Distance 490:550	0.043	0.028	-0.077	0.211	0.041	0.065
Distance 493:549	-0.022	-0.019	-0.062	0.224	-0.016	0.030
Distance 494:512	0.020	-0.045	-0.005	0.211	-0.030	0.022
Distance 506:510	-0.139	0.060	-0.150	-0.052	-0.011	0.064
Distance 513:542	0.139	0.208	-0.059	0.056	-0.007	-0.024
Distance 515:549	-0.014	-0.220	0.154	0.002	0.154	0.204
Distance 521:207	-0.181	-0.194	0.160	0.124	-0.024	0.024
Distance 524:207	-0.206	-0.157	0.093	0.156	-0.034	0.000
Distance 538:406	-0.104	0.020	0.087	0.075	-0.141	-0.143
Distance 555:516	0.107	0.046	-0.177	-0.113	0.158	0.245
Distance 556:516	0.076	0.050	-0.169	-0.107	0.214	0.265
Distance 564:286	0.123	-0.029	-0.051	0.009	-0.073	-0.084
Distance 578:550	0.101	0.116	-0.217	-0.119	0.220	0.298
Distance 582:559	-0.128	0.062	-0.056	-0.126	-0.044	-0.035
Distance 591:472	0.026	-0.280	0.125	0.065	-0.026	0.024
Distance 593:500	0.027	-0.294	0.131	0.073	-0.040	0.011
Distance 94:14	0.022	0.288	-0.029	-0.051	-0.049	-0.133
PCA 2	-0.144	-0.026	-0.187	-0.097	0.020	0.063
PCA 3	-0.076	-0.111	0.120	0.217	-0.113	-0.162

CHARMM Dimer B	ATP	RNA	Rec42p1	Rec42p2	Rec42p3	Stalk2p1	ZDP2p1
Distance 105:35	-0.009	-0.311	-0.270	-0.228	-0.011	0.291	
Distance 114:124	-0.134	0.005	-0.161	-0.113	-0.138	0.072	
Distance 123:422	0.053	0.344	0.198	0.156	0.047	-0.222	
Distance 180:536	-0.008	-0.611	-0.289	-0.259	-0.015	0.233	
Distance 217:197	-0.143	0.075	-0.032	-0.024	-0.140	0.132	
Distance 219:188	-0.117	0.053	-0.096	-0.076	-0.119	0.029	
Distance 225:220	-0.149	0.021	-0.127	-0.089	-0.151	0.110	
Distance 28:101	-0.056	-0.396	-0.227	-0.194	-0.057	0.311	
Distance 290:285	0.157	-0.048	0.194	0.049	0.158	-0.029	
Distance 338:348	0.013	0.428	0.092	0.075	0.018	-0.208	
Distance 341:344	-0.070	0.384	-0.016	-0.031	-0.063	-0.140	
Distance 406:567	-0.104	-0.193	-0.302	-0.238	-0.111	0.157	
Distance 423:119	0.053	0.333	0.211	0.162	0.047	-0.218	
Distance 444:288	0.019	-0.490	-0.278	-0.247	0.011	0.186	
Distance 445:288	-0.035	-0.423	-0.131	-0.137	-0.044	0.177	
Distance 449:469	-0.035	-0.394	-0.098	-0.144	-0.035	0.098	
Distance 451:443	-0.010	-0.402	0.023	-0.108	-0.005	-0.157	
Distance 466:443	-0.125	-0.199	-0.244	-0.109	-0.135	0.115	
Distance 491:488	0.053	0.061	0.241	0.228	0.049	-0.048	
Distance 492:488	0.070	0.040	0.199	0.239	0.059	-0.064	
Distance 493:519	0.073	0.015	0.255	0.239	0.070	-0.016	
Distance 493:550	0.095	0.238	0.135	0.243	0.085	-0.196	
Distance 4:15	-0.119	-0.079	-0.072	-0.056	-0.121	0.081	
Distance 507:510	0.079	0.251	0.306	0.187	0.092	-0.091	
Distance 508:544	-0.103	-0.300	-0.278	-0.182	-0.119	0.130	
Distance 538:312	-0.091	-0.467	-0.301	-0.230	-0.102	0.238	
Distance 548:493	0.065	0.022	0.277	0.300	0.058	-0.074	
Distance 556:581	0.105	0.354	0.159	0.146	0.105	-0.199	
Distance 563:406	-0.058	-0.103	-0.280	-0.227	-0.065	0.085	
Distance 577:581	0.061	0.278	0.331	0.243	0.059	-0.183	
Distance 578:581	0.053	0.347	0.120	0.114	0.050	-0.213	
Distance 580:584	0.086	0.348	0.302	0.201	0.089	-0.220	
Distance 99:96	-0.118	-0.263	-0.259	-0.210	-0.114	0.314	
PCA 1	-0.076	-0.363	-0.183	-0.145	-0.080	0.228	
PCA 4	-0.036	0.288	0.289	0.167	-0.037	-0.129	
Distance 3:46	-0.063	-0.137	0.011	0.002	-0.068	0.156	
Distance 286:406	-0.085	-0.232	-0.250	-0.194	-0.089	0.155	
Distance 42:62	-0.090	-0.186	-0.250	-0.197	-0.093	0.153	
Distance 533:536	-0.011	-0.169	-0.024	-0.118	-0.007	0.153	
Distance 465:451	0.045	0.164	-0.003	0.034	0.046	-0.153	
Distance 436:439	-0.012	0.310	0.048	-0.002	-0.005	-0.153	
Distance 585:453	-0.002	0.304	0.064	0.041	0.010	-0.151	
Distance 533:542	-0.097	-0.312	-0.274	-0.179	-0.104	0.150	
Distance 571:447	0.100	0.361	0.044	0.072	0.101	-0.149	
Distance 283:403	0.038	0.251	0.112	0.048	0.051	-0.148	
Distance 270:440	0.010	0.295	0.077	0.014	0.016	-0.145	
Distance 450:573	0.071	0.323	-0.078	-0.021	0.068	-0.143	
Distance 293:438	0.021	-0.203	-0.136	-0.100	0.025	0.143	
Distance 570:445	0.059	0.359	-0.050	-0.034	0.063	-0.143	
Distance 26:15	0.094	0.085	0.116	0.122	0.089	-0.142	

Figure 2.34: Calculated Residue-Pocket PCCs for dimer simulations. Values are coloured from low to high using a red-white-blue colour scale.

AMBER Monomer	ATP	RNA	RecA2p1	RecA2p3	Stalkp1	ZBDp1
Distance 169:209	-0.066	0.453	0.003	0.093	-0.045	-0.013
Distance 180:214	0.223	-0.052	-0.143	-0.026	0.073	-0.024
Distance 203:534	0.231	0.046	-0.138	-0.165	-0.054	0.006
Distance 209:167	-0.073	0.477	-0.013	0.116	-0.037	-0.030
Distance 209:175	-0.067	0.439	-0.004	0.087	-0.046	-0.051
Distance 251:275	-0.145	-0.431	0.175	-0.095	-0.033	-0.034
Distance 275:254	-0.136	-0.424	0.188	-0.124	-0.053	-0.050
Distance 281:430	-0.076	-0.251	0.090	-0.162	-0.090	-0.062
Distance 283:463	0.054	-0.140	-0.172	0.006	0.089	-0.049
Distance 285:463	0.078	-0.102	-0.180	-0.012	0.085	-0.058
Distance 289:440	-0.016	0.438	0.037	0.134	-0.015	-0.032
Distance 290:319	0.235	0.304	-0.194	0.089	0.058	0.031
Distance 291:318	0.232	0.302	-0.143	0.083	0.065	0.031
Distance 294:319	0.233	0.075	-0.109	0.041	0.025	-0.013
Distance 301:254	-0.098	-0.383	0.212	-0.219	-0.065	-0.038
Distance 303:255	-0.021	0.073	-0.128	0.222	0.030	-0.013
Distance 314:539	-0.213	-0.295	0.090	-0.129	-0.019	-0.027
Distance 340:183	-0.028	0.451	0.027	0.186	0.029	0.064
Distance 342:216	0.225	-0.074	-0.212	-0.022	0.096	0.013
Distance 34:37	-0.101	-0.005	0.050	-0.014	-0.090	0.059
Distance 377:311	-0.302	-0.057	0.245	-0.063	-0.083	0.001
Distance 410:182	-0.118	0.539	0.035	0.236	-0.018	0.055
Distance 42:62	0.104	0.086	0.014	0.087	-0.027	0.224
Distance 449:589	-0.061	-0.199	0.344	-0.039	-0.019	0.011
Distance 452:456	0.093	-0.022	-0.025	-0.150	-0.047	-0.102
Distance 453:567	-0.023	0.018	0.116	-0.018	-0.012	0.103
Distance 457:454	0.079	0.075	-0.121	0.032	-0.002	-0.100
Distance 458:454	0.089	0.023	-0.117	-0.005	-0.004	-0.113
Distance 463:288	0.105	0.079	-0.118	0.060	0.086	0.005
Distance 466:443	0.204	-0.133	-0.249	-0.073	0.085	-0.039
Distance 470:447	-0.183	-0.271	0.296	-0.189	-0.071	0.002
Distance 473:574	-0.193	-0.108	0.438	0.063	-0.040	-0.012
Distance 475:576	0.013	0.488	-0.015	0.229	0.053	0.032
Distance 475:580	0.129	0.432	-0.160	0.117	0.049	0.052
Distance 476:501	0.017	0.088	-0.144	0.282	0.061	-0.043
Distance 476:575	-0.079	0.273	-0.009	0.273	0.050	-0.014
Distance 489:486	-0.149	0.206	0.082	0.134	0.010	0.119
Distance 490:552	0.167	0.288	-0.154	0.144	0.097	0.009
Distance 491:551	0.166	0.318	-0.321	0.087	0.039	0.033
Distance 491:552	0.184	0.203	-0.331	0.025	0.046	-0.032
Distance 493:479	-0.052	0.088	0.078	0.281	0.025	-0.001
Distance 497:577	0.200	0.165	-0.329	0.056	0.053	0.002
Distance 49:46	0.005	0.034	0.019	-0.019	-0.002	-0.108
Distance 513:498	0.129	0.131	-0.291	0.059	0.061	-0.016
Distance 522:205	0.193	0.143	-0.204	-0.005	0.086	-0.033
Distance 548:516	-0.059	0.367	0.107	0.314	0.067	-0.032
Distance 553:557	0.030	0.055	-0.016	-0.011	0.005	0.135
Distance 554:487	0.087	0.418	-0.094	0.215	0.049	-0.005
Distance 569:286	0.246	0.182	-0.046	0.051	0.040	0.036
Distance 580:583	0.163	0.200	-0.338	0.004	0.043	0.008

CHARMM Monomer	ATP	RNA	RecA2 p1	RecA2 p2	RecA2 p3	Stalk p1	ZBD p1
Distance 163:523	-0.042	-0.128	-0.033	-0.056	0.331	-0.220	0.111
Distance 179:339	0.027	0.056	-0.002	-0.033	0.443	-0.193	0.038
Distance 179:531	-0.056	-0.088	-0.035	0.014	0.235	-0.121	0.133
Distance 179:534	-0.057	-0.151	-0.098	-0.016	0.277	-0.189	0.134
Distance 181:230	-0.057	-0.212	-0.036	-0.195	0.286	-0.330	0.098
Distance 203:484	0.032	-0.084	-0.049	-0.110	0.288	-0.285	0.128
Distance 211:204	-0.018	-0.216	-0.081	-0.149	0.111	-0.155	0.063
Distance 227:182	-0.019	-0.130	-0.098	0.067	0.012	-0.006	0.146
Distance 234:390	-0.031	0.124	0.110	0.129	-0.027	0.022	-0.094
Distance 281:437	-0.013	-0.106	-0.071	-0.104	0.303	-0.277	0.036
Distance 284:456	0.008	-0.150	-0.054	0.003	0.198	-0.281	0.035
Distance 339:200	-0.011	0.008	-0.007	-0.037	0.358	-0.180	0.045
Distance 341:335	0.002	0.016	0.046	-0.049	0.323	-0.202	-0.040
Distance 347:351	0.095	-0.016	-0.039	-0.039	0.268	-0.167	0.065
Distance 375:536	0.009	0.028	0.070	-0.169	-0.083	-0.126	-0.037
Distance 387:398	0.066	0.130	0.026	0.135	-0.234	0.360	-0.067
Distance 402:430	0.052	-0.177	-0.122	-0.052	0.231	-0.157	0.101
Distance 404:422	-0.026	-0.161	-0.104	0.049	0.145	-0.032	0.067
Distance 41:38	-0.129	-0.031	-0.073	-0.028	-0.134	-0.046	-0.024
Distance 420:423	0.040	-0.095	-0.101	-0.071	0.309	-0.038	0.119
Distance 425:280	0.004	0.126	0.032	0.097	-0.217	0.231	-0.111
Distance 427:419	-0.002	0.132	0.118	0.033	-0.269	0.040	-0.089
Distance 433:428	0.087	0.197	0.014	0.075	-0.108	0.366	-0.018
Distance 433:429	-0.029	-0.009	-0.091	0.036	-0.115	0.232	-0.073
Distance 436:397	-0.066	-0.236	-0.092	-0.076	0.254	-0.262	0.019
Distance 437:287	0.033	0.000	0.016	-0.027	0.329	-0.223	0.017
Distance 443:464	0.027	0.128	0.072	-0.049	0.366	-0.172	0.013
Distance 444:464	0.057	0.189	0.069	-0.032	0.339	-0.080	-0.016
Distance 444:567	-0.067	-0.048	-0.044	0.094	-0.340	0.158	-0.006
Distance 445:463	0.023	0.138	0.104	-0.041	0.318	-0.130	-0.037
Distance 451:455	-0.124	0.033	0.008	0.003	0.068	-0.023	0.043
Distance 460:439	-0.031	-0.101	0.011	0.142	-0.165	0.018	0.029
Distance 461:437	0.040	0.166	0.105	-0.016	0.269	-0.118	-0.024
Distance 461:456	-0.030	-0.240	-0.088	0.012	0.033	-0.206	0.056
Distance 465:443	-0.092	-0.231	-0.029	-0.004	-0.229	-0.044	0.038
Distance 465:568	-0.101	-0.212	-0.048	-0.051	-0.189	-0.085	0.031
Distance 468:445	-0.122	-0.275	-0.086	0.078	-0.008	-0.053	0.028
Distance 470:447	-0.069	-0.246	-0.093	-0.043	0.068	-0.089	0.039
Distance 474:498	-0.030	-0.200	-0.053	-0.131	0.267	-0.148	0.073
Distance 483:206	0.036	0.089	0.095	-0.133	-0.083	-0.089	-0.019
Distance 499:525	-0.130	-0.119	-0.053	-0.048	0.028	0.027	0.040
Distance 49:90	0.113	0.056	-0.048	-0.006	0.107	-0.030	0.130
Distance 49:92	0.112	-0.016	-0.062	-0.075	0.220	-0.112	0.144
Distance 500:525	-0.037	-0.091	-0.088	-0.069	0.134	-0.060	0.144
Distance 508:527	-0.104	-0.124	-0.058	0.021	-0.199	0.093	-0.045
Distance 513:531	-0.028	-0.273	-0.109	0.007	0.069	-0.161	0.103
Distance 514:480	0.023	0.010	-0.015	0.122	-0.092	0.039	-0.067
Distance 514:485	-0.001	-0.196	-0.160	-0.070	-0.044	-0.031	0.039
Distance 520:496	0.043	0.002	-0.015	0.031	0.149	-0.015	0.121
Distance 524:527	0.014	-0.014	-0.017	0.004	-0.338	0.145	-0.074

Figure 2.35: Calculated Residue-Pocket PCCs for monomer simulations. Values are coloured from low to high using a red-white-blue colour scale.

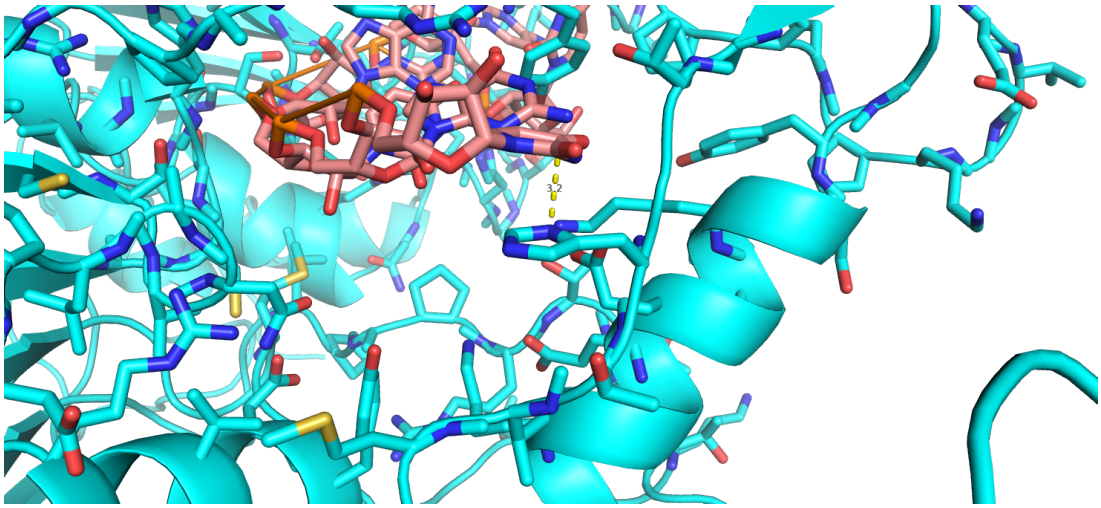


Figure 2.36: One example of the $\pi - \pi$ stacking stabilising interaction observed between U12 and HIS230 commonly observed during backward translocation simulations. This interaction heavily altered the movement of the ssRNA from what was aimed for, but was still not strong enough to stabilise the state once the biasing forces were removed.

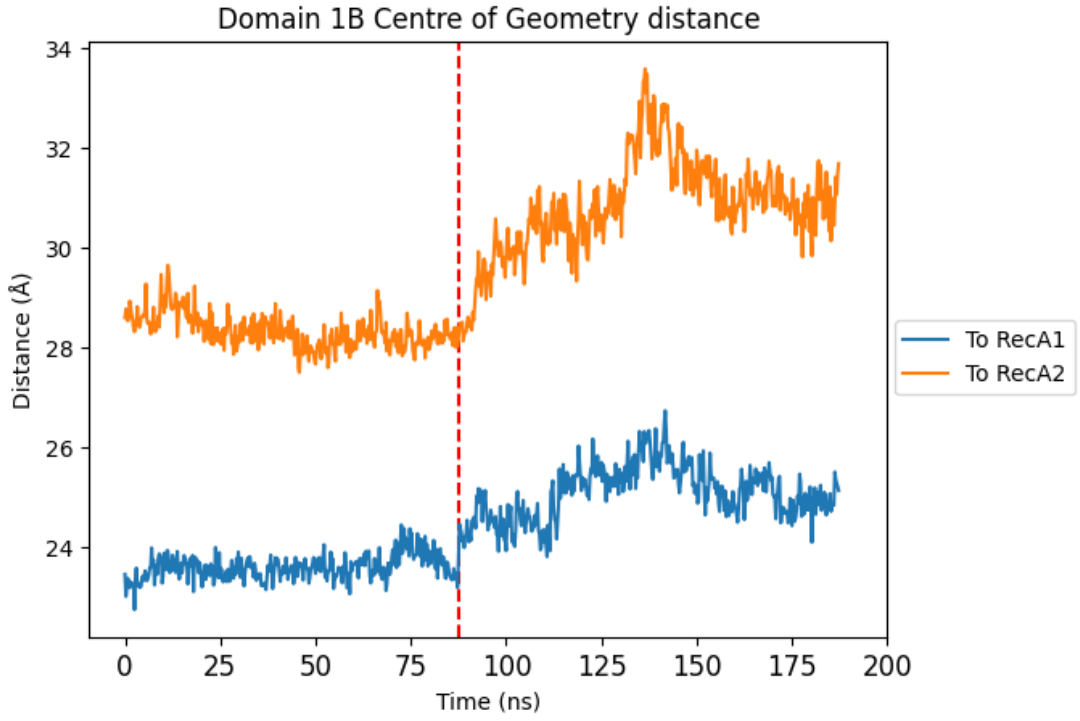


Figure 2.37: Distances to Domain1B from both RecA domains during the biased forward simulations. Dashed red line indicates the introduction of any biasing forces.

2.5 Case Study 3: Kemeny Constant optimization

2.5.1 Overview

This method development project looked to develop upon the group's prior work creating clustering methods to generate Markov chains optimised to preserve the Kemeny constant. These chains have a wide applicability across biology and other sciences, emerging as a useful strategy to avoid many of the complications of using coordinates and determining effective RCs. Previous work had constructed a closed form expression for the constant in terms of the equilibrium distribution of a clustered chain. The aim was to exploit the smooth nature of these probabilities to perform a gradient based optimisation of a clustering, building upon similar work in the area applying GNNs to clustering problems. This application was not as simple as first thought, but provided a relatively novel application of GNNs by turning their training instability into somewhat of a positive. It shows how large-scale training of GNNs can be used as both a sampling and optimization scheme for difficult globally non-convex criteria on graphs, and serves as a testament to their expressive ability. My responsibility for the work consisted of devising and implementing the initial GNN methodology using the linear decoder and running all experiments, along with writing the paper and providing the analysis and figures. The implementations of the transformer decoder networks and the penalty function were handled by my co-authors. It resulted in the publication of an academic paper:

- Kemeny Constant-Based Optimization of Network Clustering Using Graph Neural Networks, Sam Alexander Martino, João Morado, Chenghao Li, Zhenghao Lu, and Edina Rosta, The Journal of Physical Chemistry B 2024 128 (34), 8103-8115, DOI: 10.1021/acs.jpcb.3c08213

2.5.2 Introduction

Markov chains (MCs), and their extensions, have become a key tool across multiple domains to model dynamical systems[410,411]. In their simplest form, MCs incorporate sequential data as a Markovian process into a stochastic model, where the probability of observing the next event depends solely on the preceding event.

By eliminating the long timescale dependencies that often occur in complex systems, MCs offer a robust and formal method to interpret high-dimensional data sets in the emergent era of big data. The simplicity, transferability, and capability of MCs reduce the difficulties associated with modelling and interpreting ordered dynamics, and have led to a wide range of use cases. These applications encompass modelling character sequences in computer science and genomics, predicting web-user activity, and modelling time-dependent processes in physics and chemistry[412–416].

An application that has widely adopted these techniques is biomolecular simulation, where the Markov State Models (MSMs) formalism has emerged as a strategy for interpreting the results of MD simulations as transitions between aggregated conformational states[417–419]. The surge in popularity of MSMs mirrors the increased ability to quickly generate large quantities of MD data, providing a useful tool to condense massive data sets into a small and intuitive probabilistic format. MSMs offer intuitive descriptions of complex dynamic systems as transitions between metastable minima and saddle points serving as transition states, helping to elucidate the underlying functional kinetics of interest(*420–422*). Over the past few decades, several domain-specific advances led to the development of techniques to automatically generate MSM representations of molecular systems. These techniques have been employed to further sample the multitude of different configurations of biomolecular systems[423,424]. MSMs circumvent the need for complex RCs and CVs to be deciphered from dynamical protein datasets. This coordinate-free interpretation of MD simulations provide an intuitive way to cluster simulations and extract potentially biologically relevant states, critical for further applications in drug discovery.

Despite being smaller than the huge data sets from which they originate, many MSMs remain high-dimensional, often unmanageable for downstream tasks such as drug design and biomolecular engineering[425]. As size often becomes the limiting factor to establishing an intuitive understanding of systems, many different approaches have been proposed to reduce the size of MCs in a kinetically relevant fashion[426–429]. Traditional graph partitioning (GP) algorithms, such as spectral clustering, k-means, or multilevel methods, have been widely employed to address this problem[430–432]. However, these algorithms often face limitations in capturing

the complex and intricate dependencies on global structure in MCs and offer no guarantee of preserving dynamical relevant information. Alternative Markov-chain Monte Carlo approaches have been applied for dimensionality reduction on MCs, which can be used with arbitrary clustering criteria[433–435]. Nevertheless, these approaches encounter difficulties in efficiently sampling the most relevant regions of partitioning space, resulting in impractical computation times and frequent overlooking of the most optimal solutions.

This work looks to leverage ML models based on GNNs as a strategy to optimize dynamical network clusterings. It builds from previous work in the group for reducing the dimensionality of MCs in a kinetically faithful way, to build a methodology capitalising on recent developments in using GNNs to solve the more general GP problem[436–439]. Although several existing approaches use GNNs to minimize partitioning cuts or modularity, these applications do not extend to kinetic graphs or optimization based on dynamical criteria[440,441]. Additionally, these approaches mostly focus on providing approximate partitionings for large graphs rather than exploring their potential for identifying optimal partitionings in smaller networks. This project looked to extend these methods by representing MCs as kinetic networks and show how performing gradient descent on a large number of randomly initialized small GNNs can yield optimal network partitionings using complex optimization criteria.

The change of the Kemeny constant (KC) is used as a metric to quantitatively assess the degree to which a clustering preserves the underlying dynamics of the MSM. KC is an inherently dynamical quantity derived from the Mean First Passage Times (MFPTs) associated with any kinetic network, including those used within biomolecular simulations[442]. Results from the proposed implementations are compared a commonly used kinetic network clustering method for MSMs, the robust Perron-cluster analysis (PCCA+), which does not use the KC as the optimization criterion[443]. The proposed MC clustering method can produce a faithful low-dimensional replication of a system's underlying dynamics, while also providing a quantifiable measurement related to the amount of kinetic information lost. Along with implementing a general strategy for the large-scale training of GNNs for GP, the method's performance was tested on several different kinds of synthetic graphs and an example derived from MD simulations. A more complex

GNN architecture was also tested, and comparison shows how large-scale training of simple GNNs is the most effective for the task.

2.5.3 Theory

Markov Chains

While so far avoided, MCs unify many of the discussed approaches in both alignment, dimensionality reduction, and sampling[444–447]. This theory not only underpins many advanced approaches to further modelling in these areas, but provides a link to the wider area of stochastic modelling as whole. The basic concept of MCs, known as Markovianity, interprets dynamics as a probabilistic process of transitions between states where the probability of transitioning is based only on the current state. Expressing dynamics as memoryless has been shown to be a powerful strategy, as it allows researchers to disregard many of the usual complexities of dealing with long time-dependence.
The dynamical clustering criterion based on KC used in this paper is derived from the kinetic properties of an MC. Previous theoretical work in the group explored the relationship between the MFPTs and KC, providing thorough derivations of all the quantities presented below. Here the key results are summarised and interested readers are directed to the previous paper for a detailed reference[448].

In the most basic discrete-time, homogeneous MC comprising of n states, the probability vector of a random walk occupying any of the n-states at time t, $\mathbf{p}(t)$, can be expressed using a stochastic rate matrix, $\mathbf{K}$, and a set of initial occupation probabilities, $\mathbf{p}(0)$, such that[449]:

$$\mathbf{p}(t) = e^{\mathbf{K}t}\mathbf{p}(0) \tag{2.5.3.1}$$

where $t = l\tau$, since in the discrete case transitions between states occur at integer multiples of a given time interval τ. The quantity $\mathbf{Q}(\tau) = e^{\mathbf{K}\tau}$ is known as the Markov matrix and is commonly used as the fundamental representation of a Markov chain. When using this representation, time propagation corresponds

simply to repeatedly multiplying $\mathbf{Q}$ by the current state of the system.

$$\mathbf{p}(t) = \mathbf{Q}^l \mathbf{p}(0) \tag{2.5.3.2}$$

To ensure the conservation of probability, the columns and rows of the matrix $\mathbf{K}$ must sum to zero:

$$\begin{aligned} \sum_j K_{ji} &= 0, \ \forall i \\ \sum_i K_{ji} &= 0, \ \forall j \end{aligned} \tag{2.5.3.3}$$

The matrix $\mathbf{K}$, or similarly $\mathbf{Q}$, can be calculated from experimental or simulated transition data, such as MD trajectories, or a series of string-like sequences as typically used in genomics.
$\mathbf{K}$ fully defines the dynamic properties of the MC and can be used to derive several useful quantities relating to its underlying dynamics. Assuming detailed balance, the spectral decomposition of $\mathbf{K}$ into left, ψ_i^l, and right, ϕ_i^l, eigenvectors and corresponding eigenvalues, λ_l, is given by:

$$\mathbf{K} = \sum_{l=1}^{N} \lambda_l \psi^{(l)} \phi^{(l)} \tag{2.5.3.4}$$

where the index l orders the eigenvalues in descending order. These eigenvalues are upper-bounded by 0 and are guaranteed to be real due to the detailed balanced assumption. The eigenvalue corresponding to the largest right eigenvector, $\psi^{(1)}$, is known as the equilibrium probability and is commonly denoted by $\mathbf{p}^{\text{eq}}$, corresponding to the distribution the system converges to as $t \to \infty$.

The other eigenvectors contain useful kinetic information corresponding to the relaxation and mixing times of the MC. These eigenvectors can be used to derive an expression for the average time it takes for a random walk starting from state i to first arrive at state j, *i.e.*, the MFPTs of an MC, denoted t_{ji}:

$$t_{ji} = \frac{1}{p_j^{\text{eq}}} \left[\sum_{l>1} \frac{1}{|\lambda_l|} \psi_j^{(l)} (\phi_j^{(l)} - \phi_i^{(l)}) \right] \tag{2.5.3.5}$$

which can be alternatively expressed in matrix form as:

$$t_{ji} = \frac{1}{p_j^{\text{eq}}}\left[(\mathbf{p}^{\text{eq}}\mathbf{1}_N^T - \mathbf{K})_{jj}^{-1} - (\mathbf{p}^{\text{eq}}\mathbf{1}_N^T - \mathbf{K})_{ji}^{-1}\right] \tag{2.5.3.6}$$

KC is the surprising result that the sum over all states of the product of the MFPTs with the equilibrium probabilities, p_j^{eq}, is a constant value for all i, *i.e.*:

$$\zeta = \sum_j p_j^{\text{eq}} t_{ji} = \sum_j \sum_{l>1} \frac{1}{|\lambda_l|} \psi_j^{(l)} (\phi_j^{(l)} - \phi_i^{(l)}) = \sum_{l>1} \frac{1}{|\lambda_l|} \tag{2.5.3.7}$$

where the fact that $\sum_j \phi_j^{(l)}\ \psi_j^{(l)} = 1, \forall l$ and $\sum_j \psi_j^{(l)} = \delta_{l1}$ was used[448].
Intuitively, the KC is an important quantity to measure the overall timescales present in a dynamical system as it is equal to the sum of all relaxation times, $1/|\lambda_l|$.

Graph Partitioning

In what follows, the terms clustering and partitioning are used interchangeably to refer to the GP problem.
The general partitioning problem consists of dividing the nodes of G into M discrete non-overlapping subsets, $L \in \{1, \ldots, M\}$, such that specific criteria are met. These objectives can be general and applicable to nearly all graphs, and include, *e.g.*, optimizing connectivity, minimizing inter-cluster edges, or maximizing intra-cluster similarities. Furthermore, these objectives can also be domain-specific, based on the underlying model the network represents, such as how the KC represents the dynamical information of the system. The $(N \times M)$ partitioning assignment matrix, $\mathbf{S}$, assigns the nodes of G into M distinct states:

$$S_{iL} = \begin{cases} 1, & \text{if } v_i \in L \\ 0, & \text{if } v_i \notin L \end{cases} \tag{2.5.3.8}$$

Since the discrete nature of the assignment matrix poses difficulties for gradient-based methods, it is common to relax this constraint and work with continuous probabilities instead. In practice, this means introducing a soft assignment matrix instead, where S_{iL} now denotes the probability of node v_i belonging to cluster L,

subject to the conservation of probability:

$$\sum_{L}^{M} S_{iL} = 1 \; \forall \; v_i \in V \tag{2.5.3.9}$$

where L runs over all M possible clusters.

Some of the most commonly used clustering criteria used for the GP problem are the modularity and the minimum cut. The modularity, Q, measures how the connectivity of a GP differs from what would be expected in a purely random graph with the same node degree distribution. Consequently, this metric provides a quantitative assessment of how statistically surprising a proposed partitioning is, with higher values implying the presence of underlying communities within the data[450]. The modularity is calculated using the equation:

$$Q = \frac{1}{2m} \sum_{ij}^{N} \sum_{L}^{M} \left[A_{ij} - \frac{d_i d_j}{2m} \right] S_{iL} S_{jL} \tag{2.5.3.10}$$

where m is the total number of edges in the graph, d_i and d_j are the degrees of nodes v_i and v_j, respectively, A_{ij} is given by Eqn. (1.2.2.12), and S_{iL} and S_{jL} are given by Eqn. (2.5.3.8). Networks characterized by high modularity exhibit dense intra-cluster node connections, but sparse inter-cluster node connections. While there have been criticisms of the underlying assumptions for modularity, it is a widely used and useful partitioning criterion in many areas[451–453].

The minimum cut, C, of a given partitioning is the number of inter-cluster edges, which can be calculated using:

$$C = \sum_{i}^{N} \sum_{j \neq i}^{N} \sum_{K}^{M} \sum_{L \neq K}^{M} S_{iK} S_{jL} A_{ij} \tag{2.5.3.11}$$

where all indices and matrix values are given as previously defined. The simplicity of this criterion has led to its popularity across multiple domains, spurred on by its proven equivalence to the max-flow criterion[454]. It stands as one of the most studied partitioning criteria , alongside its further generalisations commonly used to obtain more balanced clusters with minimal cuts[455–459]. Intuitively, the

minimum cut partitioning of a network corresponds to the partitioning with the least amount of inter-cluster edges.

The Davies-Bouldin index (DBI) quantifies the average similarity between clusters[460]. Calculated based on the distance between clusters relative to the size of the clusters themselves:

$$\mathrm{DBI} = \frac{1}{M}\sum_{L}^{M}\max_{K\neq L}\left[\frac{\sigma_K + \sigma_L}{\Delta_{KL}}\right] \tag{2.5.3.12}$$

where L runs over all possible M clusters, Δ_{KL} denotes the distance between cluster centroids, $\Delta_{KL} = \|\mathbf{B}_K - \mathbf{B}_L\|$. σ_L represents the cluster diameter, calculated as the average distance between the feature vectors and the centroid of the respective cluster:

$$\sigma_L = \frac{1}{N}\sum_{i}^{N}\|\mathbf{F}_i - \mathbf{B}_L\| \tag{2.5.3.13}$$

where $\mathbf{F}_i$ represents the feature vector of node i. The centroid $\mathbf{B}_L$ is computed by averaging the features vectors of all N nodes i within cluster L. A lower DBI value (bounded from below by 0) indicates better partitioning, as it suggests higher inter-cluster dissimilarity in comparison to intra-cluster similarity. Therefore the DBI provides a metric for assessing clustering performance by its ability to separate clusters depending on their node features.

Considering a MC clustering, the equilibrium probability of cluster L, P_L^{eq}, can be defined as simply the sum of the populations of the individual nodes:

$$P_L^{\mathrm{eq}} = \sum_{i\in L} p_i S_{iL} \tag{2.5.3.14}$$

The KC can then be redefined in terms of a partitioned MCs KC:

$$\zeta_{\mathbf{S}} = \sum_{L} P_L^{\mathrm{eq}}\hat{t}_{LK} \tag{2.5.3.15}$$

where $\hat{t}_{LK}$ is the coarse-grained mean first passage time from state K to L, defined as $\hat{t}_{LK}$[448]:

$$\hat{t}_{LK} = \frac{1}{P_K^{\mathrm{eq}}P_L^{\mathrm{eq}}}\sum_{i\in K, j\in L} p_j^{\mathrm{eq}}p_i^{\mathrm{eq}}t_{ji} - \frac{1}{(P_L^{\mathrm{eq}})^2}\sum_{i,j\in L} p_j^{\mathrm{eq}}p_i^{\mathrm{eq}}t_{ji}, \tag{2.5.3.16}$$

which leads to a variational form compared with the unclustered KC(*436*):

$$\zeta = \zeta_{\mathbf{S}} + \sum_{L}^{M} \frac{1}{P_L^{\mathrm{eq}}} \sum_{i,j \in L} p_j^{\mathrm{eq}} p_i^{\mathrm{eq}} t_{ji} \tag{2.5.3.17}$$

Rearranging this equation yields the partitioning metric ΔK:

$$\zeta - \zeta_{\mathbf{S}} = \Delta K = \sum_{L}^{M} \frac{1}{P_L^{\mathrm{eq}}} \sum_{i,j \in L} p_j^{\mathrm{eq}} p_i^{\mathrm{eq}} t_{ji} = \sum_{L}^{M} \frac{1}{P_L^{\mathrm{eq}}} \sum_{i,j} p_j^{\mathrm{eq}} p_i^{\mathrm{eq}} t_{ji} S_{iL} S_{jL} \tag{2.5.3.18}$$

From this equation, it is clear that for the trivial single-state partitioning $M = 1$, $\Delta K = \zeta$ holds, and when $M = N$, the value of ΔK is 0. As KC represents the sum of relaxation times in a system, and a clustering speeds up relaxation times by removing intra-cluster relaxation times, maximizing the KC of a proposed partitioning preserves the original timescales as closely as possible. From Eqn. (2.5.3.18), it can be seen that maximizing the KC is identical to minimizing ΔK, and so this provides a useful metric of how kinetically disconnected the clusters are, in a form suitable for gradient-based optimization methods like neural networks.

2.5.4 Methodology

The proposed method assumes that a given GNN architecture, $\Phi(\theta)$, is sufficiently expressive such that the mappings it explores, from the space of d_{feat} dimensional node features, $\mathbf{F} \in \mathcal{R}^{N \times d_{\mathrm{feat}}}$, to the space of possible M-state partitionings, $\mathbf{S} \in \mathcal{R}^{N \times M}$, includes the optimal solution, $\mathbf{S}^{\mathrm{opt}}$, to the KC GP problem. Alongside this, it is assumed there is a region of parameter space which provides a smooth path to this solution, and thus can be optimized via gradient descent to yield it. Similar to Monte Carlo approaches to similar problems, $\mathbf{S}^{\mathrm{opt}}$ is searched for by initializing Φ with several different sets of parameters θ and optimizing them in parallel using modern ML tooling. In this framework, GNNs are an effective Ansatz to begin partitioning graphs using complex criteria. They also provide a generic and intuitive optimization framework with gradient descent. As Φ is a general model, operating in $\mathcal{R}$ without any problem-specific restrictions on its values or gradients, it can yield the trivial solutions to Eqn. (2.5.3.18), such as the aforementioned $M = N$ case. To overcome this, inspiration from similar

work on producing balanced clusters provided evidence for augmenting the loss with a penalty function to ΔK, which drives the network away from trivial solutions[440,441]. The final form of the loss function used, $\mathcal{L}$, was subsequently given by:

$$\mathcal{L} = \underbrace{\sum_{L}^{M} \frac{1}{P_L^{\mathrm{eq}}} \sum_{i,j}^{N} p_j^{\mathrm{eq}} p_i^{\mathrm{eq}} t_{ji} S_{iL} S_{jL}}_{\Delta K} + \underbrace{\sum_{L}^{M} \frac{\eta}{\sqrt{2\pi\sigma}} \exp\left\{ \left[-0.5 \left(\frac{\sum_i^N S_{iL}}{\sigma} \right)^2 \right] \right\}}_{\text{Penalty Function}} \tag{2.5.4.1}$$

Optimal KC partitionings can have unbalanced subsets, with clusters that contain only a few nodes often being the most interesting transition states[420]. Therefore, the form of the penalty function is designed to penalise any invalid clustering (where any cluster has zero nodes), while also allowing the network to converge significantly unbalanced partitions with at least one node. The final penalty function is based on a Gaussian centred at 0, parametrized in this work using $\sigma = 1.0$ and $\eta = 15$. This means the penalty increases as any cluster approaches 0 nodes and becomes negligible for clusters with 1 or more nodes.

As MCs do not fit into the generic graph structure outlined in Section (1.2.2), simple methods were derived for interpreting node connectivity using an adjacency matrix, $\mathbf{A}$, and node feature matrix, $\mathbf{F}$. The connectivity of the graph was decided using a cut-off method, taking all node pairs (v_i, v_j) with $(Q_{ij} + Q_{ji})/2 > c$, where c is a constant. A constant of $c = 0.015$ was found to be sufficient for all graphs in performed experiments, although in other applications this could be changed depending on the density of the resulting network.

In order learn effectively, GNNs require a distinct feature vector F_i associated with each node N, which is used as the initial input to the model. There are several different ways to produce N feature vectors of dimension d_{feat}. Here, general features were produced based on the properties of the MSM, and did not include system specific information. A systematic exploration of various node features was undertaken, and below is a list of the different featurization methods used for KC-based clustering.

- **Markov/Rate Matrix:** As an MC can be fully defined by either a Markov matrix $\mathbf{Q}$ or rate matrix $\mathbf{K}$, both of shape $(N \times N)$, the simplest featurization

is to use the columns of either defining matrix, depending on which is used to define the MC. It produces a set of features for each node $\mathbf{F_i}$ where each element of the feature matrix is simply a column of the Markov or rate matrix: $\mathbf{F_i} = (Q_{1i}, ..., Q_{Ni})$ or $\mathbf{F_i} = (K_{1i}, ..., K_{Ni})$. This corresponds to the outgoing rates/transition probabilities for each node.

As done in other work on clustering using GNNs, it was also tested whether performing PCA on the matrix $\mathbf{K}$ or $\mathbf{Q}$ prior to splitting into individual F_i would improve performance[440]. To do so, PCA was performed using the Sci-Kit Learn Python library[461,462]. Interpreting each column $\mathbf{F}_i$ as a sample and the rows as features, all components of the PCA were taken. This can be seen as a rotation of the defining matrix onto the perpendicular directions along which the node transition probabilities or rates vary the most, with the aim of making nodes with similar transitions appear more similar.

- **Adjacency Matrix:** The adjacency matrix of an MC, defined using a simple cut-off of the defining matrix $(Q_{ij} + Q_{ji})/2 > c = 0.015$. This yields a binary $(N \times N)$ square matrix, which can be split into columns representing each node with $d_{\text{feat}} = N$.

  A PCA of this was also tested, utilizing the same strategy as before.

- **MFPT Matrix:** The MFPT matrix with elements t_{ji} of the original graph was calculated using Eqn. (2.5.3.6). This generates an $(N \times N)$ matrix of the MFPTs between nodes in the network. The matrix was split into feature vectors F_i of size $d_{\text{feat}} = N$ using the columns of the MFPT matrix. A PCA of this was also tested.

- **Eigenvectors:** As the eigenvalues and eigenvectors of the Markov or rate matrix describe important kinetic properties, the eigenvectors of these were also tested as features. For this, the N eigenvectors were concatenated into a $(N \times N)$ matrix, and its columns were assigned as individual node features.

- **Trainable Embedding:** Trainable embeddings have found much success in LLMs, where they provide an efficient way to represent the sentiments of different tokens, and have been employed previously as random features

for GNNs[463]. Trainable embeddings are equivalent to an associative array data structure, where a predefined set of keys map each point to a vector of arbitrary size. The vector is initialized with random numbers which are then trained along with the rest of the network during back-propagation steps. The torch implementation was used, which uses integer keys from $\{1 \dots N\}$ representing each node[464]. Several embedding sizes were tested before settling on $d_{\text{feat}} = 128$ which provided the best performance across all systems.

- **Single layer linear network:** A single linear layer that takes as input for each node the $3N$-dimensional vector that is formed by concatenating the corresponding columns of the (i) Markov/rate, (ii) adjacency, and (iii) MFPT matrices. The output dimension was $d_{\text{feat}} = 128$ in all systems to match the trainable embedding, and the weights of this layer were updated as part of the training process.

All neural network architectures tested used a GNN-based encoder, mapping the d_{feat} dimensional features into an embedding dimension d_{embed} via a hidden representation of size d_{hidden}. The initially devised network used 4 GraphSAGE layers as an encoder, using mean pooling was used for the aggregation function as it provided better stability during training. This was then followed by a series of 4 linear layers of dimension d_{hidden} as a decoder. Building from this simplest model, more complex architectures were tested to see if they could improve performance. GATv2 layers were implemented in the encoder. A Transformer Decoder (TD) was also tested, utilising two 4-head multi-head attention layers and an additive positional encoding system parametrised by a linear network operating on each nodes features. This resulted in four models being tested:

1. GraphSAGE layers and the linear decoder (GraphSAGE + LD)
2. Linear decoder with GATv2 layers (GATv2 + LD)
3. GraphSAGE with transformer decoder (GraphSAGE + TD)
4. GATv2 with transformer decoder (GraphSAGE + TD)

The TD implementation used two transformer layers with a dropout layer between them, along with skip connections at each step. This transformer-like, attention-based decoder was chosen to provide a comparison between a simpler model and one that is larger, more expressive, and better suited to handle highly contextual sequences, such as the node-cluster assignment matrix. All NNs used are described in detail in Fig. (2.38).

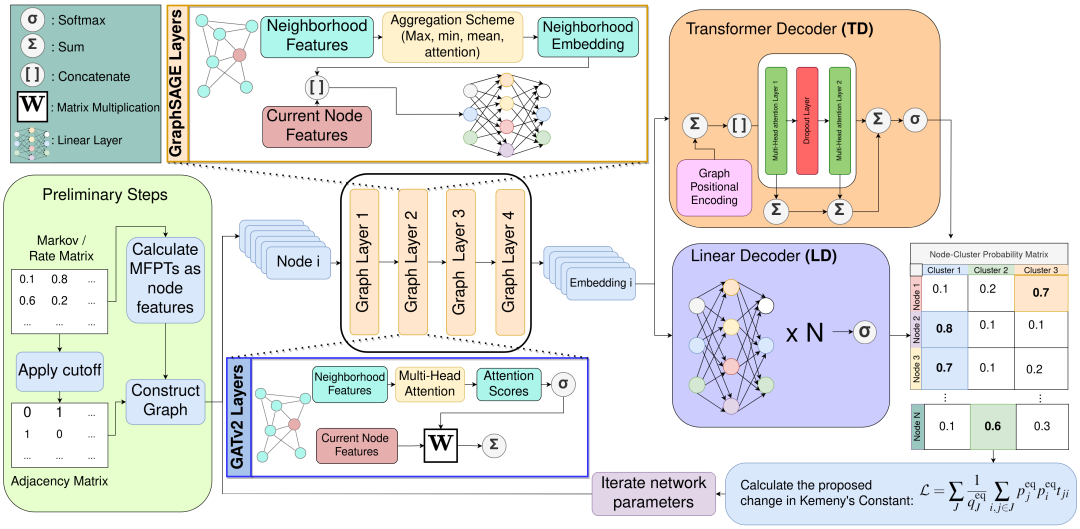


Figure 2.38: Details of all four GNN architectures used, along with a description of the preprocessing required to begin training, together with an overview of the training loop process for a single gradient optimization step. Implementations of the GraphSAGE and GATv2 GNN layers used in the encoder are provided, along with a schematic description of the transformer decoder (TD) and Linear Decoder (LD) architectures. Activation functions for the various layers are not included for brevity.

To enable fast and efficient training of the networks, a general parallel methodology for training GNNs for partitioning was established, outlined in Fig. (2.39). This requires three additional functions to be defined, as well as a valid loss function and the number of processes P:

- **Generator Function** $\mathcal{I}$: This function returns valid parameters θ for a given neural network Φ. This provides the initial starting point in parameter space for optimization, and complex generators can be used to provide a quicker convergence. In this work, however, simple initialization using both Kaiming and Glorot algorithms was tested. The Glorot method was used in final runs due to its slight advantage observed in early tests[465,466].

- **Culling Function** $\mathcal{C}$: This function determines whether or not a worker thread has finished training a network, and is called after a predetermined number of optimization steps has finished. Observing individual runs, it is common to see neural networks which initialize in local minima of parameter space and are unable to be trained. An effective culling function allows for these types of runs to be terminated early, allowing for more time to be spent in interesting regions of parameter space. To show the validity of the method, all experiments in this paper terminate after a predetermined number of training steps.

- **Termination Function** $\mathcal{T}$: This function determines when to stop training, and when combined with effective choices for $\mathcal{I}$ and $\mathcal{C}$, can allow for the implementation of complex methodologies for determining the convergence of an experiment.

This methodology aims to be as general as possible to make implementing future applications for different criteria straightforward. It is summarised in Fig. (2.39)

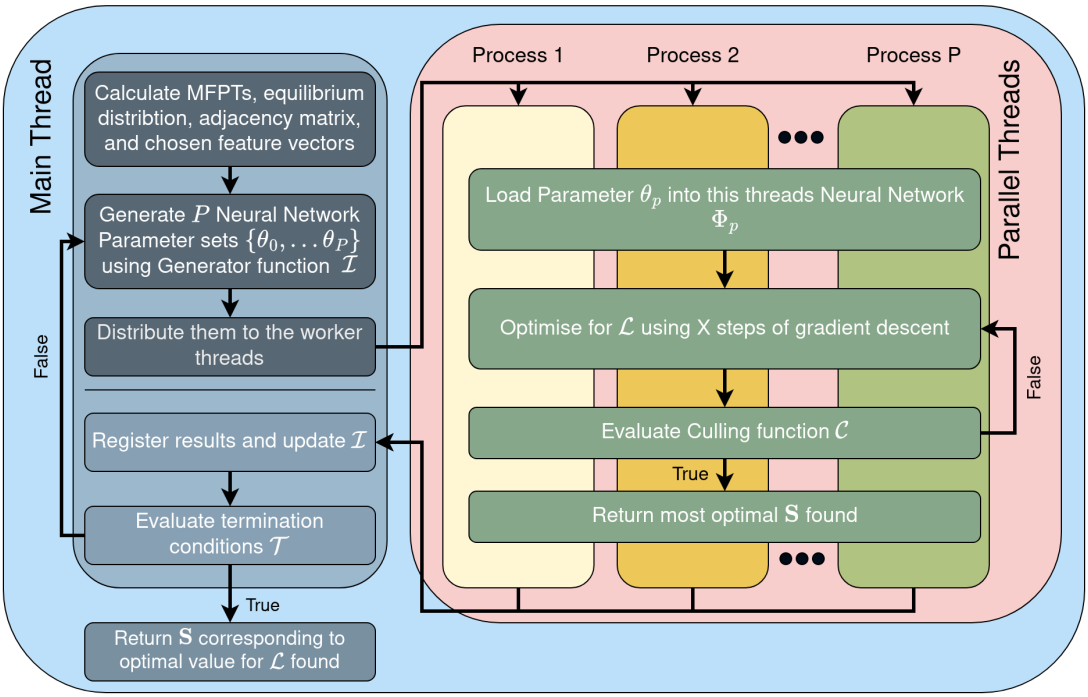


Figure 2.39: Schematic description of the algorithm used to train multiple networks in parallel, and the use of the general functions $\mathcal{I}$, $\mathcal{C}$, and $\mathcal{T}$.

To test the validity and effectiveness of using GNNs to cluster Markov chains, the method was first tested on synthetic kinetic graphs. Firstly, a simple Stochastic Block Model (SBM) was employed to generate random networks with 4 neighbourhoods[467]. The initial set of networks used sampled inter-cluster connection probabilities from a unit Gaussian distribution scaled to be in the range $[0, 0.25]$ and set intra-cluster probabilities to a constant 0.5. This led to networks with dense clusters, where the optimal partitioning always successfully separated them. Also tested were SBM networks where the optimal clustering was less clear, sampling all connection probabilities from the same Gaussian as before. For both sets of examples, SBM networks with disconnected nodes were discarded, and the random-walk normalized Laplacian matrix, $\mathbf{L}$, was used to generate rate matrices via:

$$\mathbf{L} = \mathbf{I} - \mathbf{A}\mathbf{D}^{-1} \tag{2.5.4.2}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{D}$ is a matrix with the degrees of each node along the leading diagonal. Results on these synthetic graphs were compared with clusterings and ΔK values obtained from the PCCA+ clustering. Each SBM experiment trained 100 independently initialized GraphSAGE + LD models for 250 steps of gradient descent on 500 randomly generated graphs, ranging from $N = 150$ to $N = 300$ nodes in increments of 50. This experiment was performed to test the validity of large-scale GNN training.

To establish a baseline performance for all models, the NNs were tested on a graph corresponding to a rate matrix derived from an analytical 1D potential simulation[436]. The system consisted of 100 nodes distributed along the potential, which comprised four distinct wells: three with 20 nodes, and one with 40 nodes. The optimal KC partitioning successfully separated all 4 wells into individual partitions. This example of a linear chain network architecture was chosen to test the network in situations where connectivity is limited. All four models were tested 1000 times on this example for 500 gradient descent steps each. A feature comparison was also performed on this system, where 1000 GraphSAGE + LD models were trained for 500 steps using each set of features described above.

Ultimately, the method was tested on a network derived from MD data. The MSM benchmark system from PyEMMA was used corresponding to 500 ns long

implicit water MD simulations of a small 5-amino-acid polypeptide, here referred to as the pentapeptide system[468]. The polypeptide's single letter sequence was WLALL, yielding a tiny 94-atom system perfect for inexpensively testing such methods. The simulation initially underwent clustering into 250 states using K-means on tICA projected data, followed by the calculation of a Markov matrix, which served as the input to this method[469,470]. Each experiment consisted of training 1000 models for 500 gradient descent steps to cluster the 250-state MSM into 5 states.

Given the relevance of this example to biomolecular simulations, an additional hyper-parameter search was conducted to identify the optimal network architecture. For this purpose, 1000 GraphSAGE + LD models were trained for 500 steps while varying the number of layers in both the decoder and the encoder, along with the dimensions d_{hidden} and d_{embed}. Furthermore, a feature comparison following the same methodology as previously described was also performed on this example. The results of this example were compared to both the PCCA+ and Parallel Tempering Variational Clustering (PTVC), the group's previously published Markov-chain Monte-Carlo approach[436]. The DBI score with the Markov matrix as the node features vectors was used as a metric to assess the quality of the partitionings produced.

The neural networks and relevant code were implemented in Python 3.10.6 using PyTorch 2.0 (LD models) or Tensorflow (TD models)[98,99]. Network diagrams were generated using the Netgraph library[471]. To perform gradient descent, the Adam optimizer was employed with a learning rate of 1×10^{-4}. All networks utilized $d_{\text{embed}} = 32$ and $d_{\text{hidden}} = 64$, with 4 GNN layers in the encoder and 2 layers in the decoder, unless otherwise specified. Each experiment involved the simultaneous execution of 90 processes in parallel. To compute the DBI scores, the scikit-learn implementation was utilized[461].

2.5.5 Results & Discussion

The GraphSAGE + LD approach successfully found the optimal clustering for all SBM networks in the first set, where the clusters were well-defined, in 100 training runs or less. The rate matrices were used as the features in all SBM

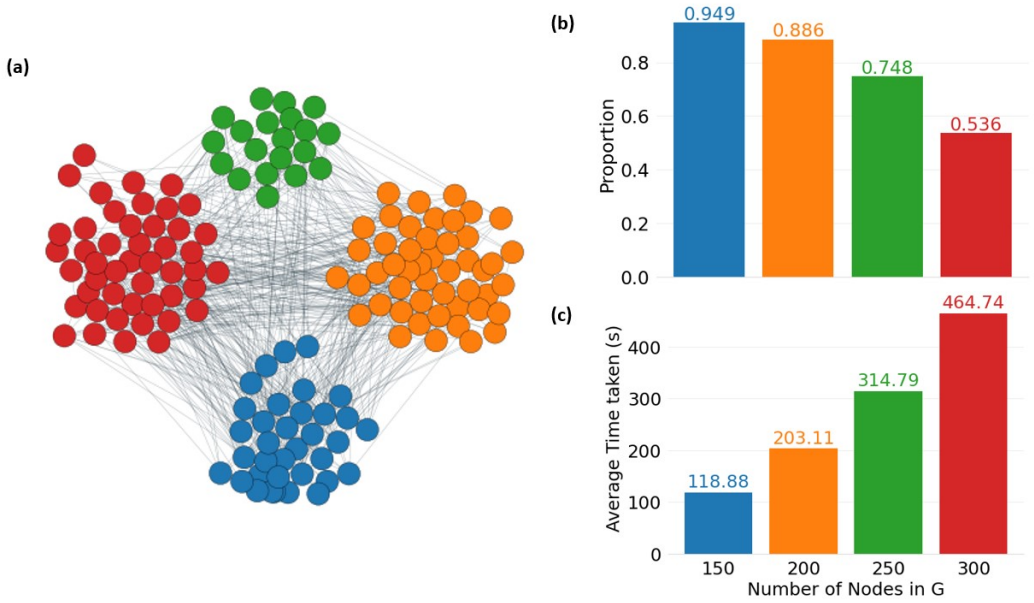


Figure 2.40: The performance of the SBM network clustering for randomly sampled cluster connection probabilities. ***(a)*** *shows an example of a partitioning produced by the GNN optimization on a SBM network.* ***(b)*** *and* ***(c)*** *provide quantitative assessments of the method's performance over 100 runs for the 500 different networks.* ***(b)*** *shows the proportion of runs where the method yields a partitioning with a lower ΔK value than PCCA+, while* ***(c)*** *highlights the average time needed in seconds for one training run when limited to run on one 2.20GHz CPU core.*

examples and the results for simple networks are shown in Fig. (2.40)(a). In this experiment the optimal clustering always matched the PCCA+ solution. Fig. (2.40)(b) and Fig. (2.40)(c) detail the performance on the second set of SBM graphs featuring a more diverse range of networks, where the PCCA+ and ΔK solutions often differed. Here the GNN implementation identified partitionings which better optimized ΔK than the PCCA+ as shown in the appendix in Fig. (2.46). The improvement diminished as the number of nodes in the network decreased, as often the clustering became clearer and able to be found by PCCA+. The stochastic nature of the GNN-based method however lends itself more to these systems where several viable partitionings have a low ΔK value. In these examples the optimal clustering is close to many other similar solutions, therefore it is helpful to evaluate many initializations to evaluate most of the very low ΔK partitionings. These initial results confirm the viability of KC as an optimization criterion and the usage of GNNs, showcasing how they are able to provide a novel clustering distinct from existing methods on non-trivial MCs.

The 1D potential Fig. (2.41)(a) was used to test the approach on sparse graphs where connectivity is limited to a linear chain, and the majority of effective ΔK partitionings lie in a small region of partitioning space. In this example the GraphSAGE + LD model was again capable of finding the global minimum clustering, although with less regularity than in the SBM models. In this regime gradient descent steps become less effective as the majority of initialised partitionings are far from the region of interest, and NNs can therefore be unable to escape their local minima. Training more neural networks can alleviate the problem by providing a wider range of starting points for optimization, and with the 1000 models used the GNN was shown to reliably find the minimum. In future applications, an effective choice of the initialization function $\mathcal{I}$ could speed up convergence, whereby $\mathcal{I}$ can learn to avoid initializing in uninteresting regions of parameter space.

A comprehensive analysis of the different node features were performed on this example to see if they improved the frequency of successful optimization outcomes. The feature comparison results are shown in Fig. (2.41)(b). They show that it is possible to use improved node features, such as the adjacency matrix or a trainable embedding in this case, to achieve a much more successful optimization for the 1D model potential-derived network.
When using the rate matrix as node features in this example, the GNN fails to find the correct clustering even once. This may be due to the difficulty in discriminating nodes at high energies (with low populations) based solely on the rate matrix, as these nodes have similar transitions. The adjacency matrix provides direct information on a node's location in the global structure of the network, leading to a noticeable advantage. The GNN implementation allows for simple changes like this to enable defining the node features on a problem-specific basis, therefore giving the method additional viability across a wide range of MCs which can vary greatly in structure. Performing such a feature comparison in an automated strategy is a viable approach to find which features work the best on a case by case basis.

All four GNN architectures were tested on the 1D potential, training 1000 models of each variety using the adjacency matrix as node features. Results are shown in Fig. (2.42)(b) and described in Table 2.3, showing that simpler models

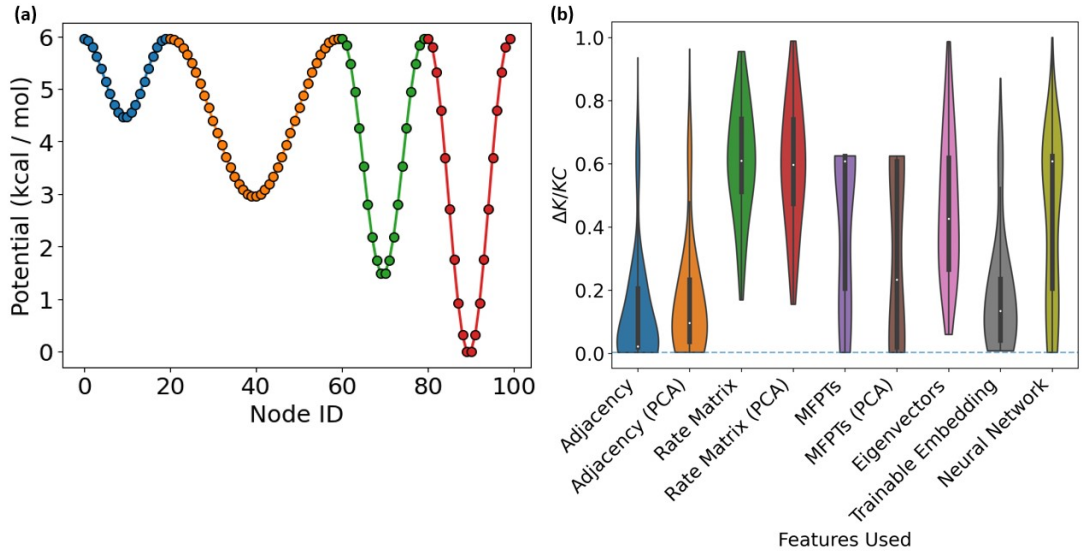


*Figure 2.41: (**a**) Plot showing the 1D potential used for an example of a linear chain network. Nodes are coloured according to the most optimal 4-state clustering for the system. (**b**) Violin plots for feature comparisons on the 1D potential network. Each experiment was run 1000 times with each set of features. The corresponding distribution of final $\Delta K/KC$ values is shown, along with a dotted blue line signifying the $\Delta K/KC$ value of the optimal solution for each problem.*

perform better at finding optimal partitionings over more complex models in this framework. Despite the better average performance the GATv2 and TD models provided, no models but the GraphSAGE + LD were able to find the optimal clustering over 1000 runs. In fact all other models were an order of magnitude away from finding any close solution. The large magnitude of some solutions is indicative of a partitioning problem where there is only a small region of interest in partitioning space, and there is not a clear global pathway to this area. The wide variance of the observed results showcases the poor ability of all models to optimize initial partitionings on this network when started from many regions of parameter space. However, in the smaller models this is mitigated by the smaller solution space explored and shorter training time required. Larger models like the GATv2 and TD models to struggle to find this region, possibly due to the more complex and unclear gradients calculated when the node embeddings are more co-dependent.

There is a sharp cut-off in the performance of all larger models, with the best value attained being consistently around 806 and the corresponding clustering

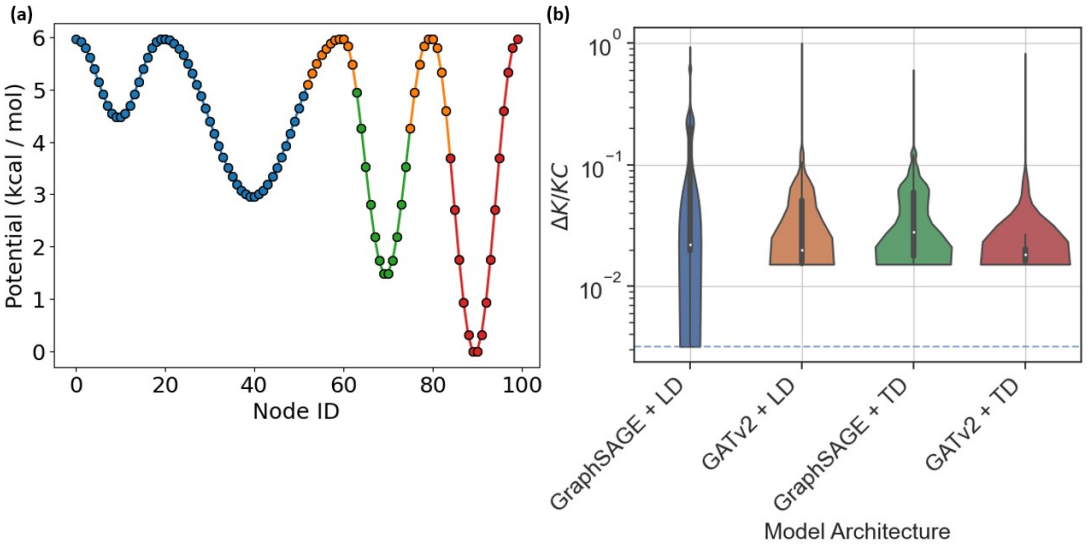


Figure 2.42: ***(a)*** *The clustering corresponding to the lowest $\Delta K/KC$ values found by all GNN models other than GraphSAGE + LD.* ***(b)*** *The distribution of $\Delta K/KC$ values from training 1000 copies of each neural network architecture on the 1D potential derived network. A logarithmic scale is used to highlight the large differences between the different models. The blue dashed line indicates the value associated with the optimal clustering for this system.*

shown in Fig. (2.42)(a). While it is difficult to pinpoint the exact cause of this lower bound, it can be theorised to be due to the larger networks' inability to discriminate between nodes. This problem is particularly challenging for GP objective functions that are not derived from kinetics-based measures, and such measures produce qualitatively incorrect optimal clustering, such as modularity[436]. Accordingly, the non-optimal solution of the GNNs shown in Fig. (2.42)(a) corresponds to a lower DBI value than that of Fig. (2.41)(a), which suggests that the DBI measure is also inadequate for this system. Values of the DBI measure are presented in the appendix in Fig. (2.47), and as this measure is based upon the similarity between node features, it supports the notion that larger networks have great difficulty discriminating nodes in this case. The optimal ΔK clustering is identical to the solution found by PCCA+ in this example.

Examples derived from numerical simulations of biomolecular systems often result in noisy derived MCs with dense connections, such as in the pentapeptide case. Here, the optimal PCCA+ clustering does not correspond to the optimal ΔK solution, with four nodes assigned to different clusters. These nodes are highlighted

Model Name	Mean	Minimum	Global Minimum Accuracy
GraphSAGE + LD	5884.74 (± 8372.62)	**169.69**	**5.6%**
GATv2 + LD	1988.50 (± 2696.11)	806.48	0.0%
GraphSAGE + TD	2201.59 (± 1796.45)	806.48	0.0%
GATv2 + TD	**1508.19 (± 2728.35)**	806.48	0.0%
PCCA+	169.69 (± 0.0)	169.69	N/A

Table 2.3: GNN architecture comparison for the 1D potential derived network. The four network architectures are compared in terms of the mean (± standard deviation) and minimum value of the KC found, alongside the number of times the optimal solution was found, calculated across 1000 experiments.

in Fig. (2.43)(a) and contribute 7.4% of the overall equilibrium probabilities as shown in appendix Fig. (2.48), a non-neglible difference to any clustering. As shown in Fig. (2.44)(b) and Table 2.4, again the simplest GraphSAGE + LD method outperforms all other GNN architectures similarly to the 1D example above. Additionally the PTVC method was not found to identify the optimal ΔK clustering when using a random initialization. The optimal solutions from different methods are compared in Fig. (2.44)(a) by evaluating the DBI score. Interestingly, the optimal KC clustering also corresponds to the lowest DBI score observed, indicating clustering better able to distribute the node features.

The GraphSAGE + LD method finds the optimal clustering less frequently than in the synthetic examples of SBM networks. This could be due to the increased number of low ΔK partitionings available, which provide many local minima which are more difficult to escape. However this is not a problem unique to the GNN method, as the PTVC method also has difficulty finding the relevant areas of partitioning space and is unable to find the optimal clustering in this example using random initialization. The dense and interconnected nature of the nodes means it is difficult for PCCA+-based methods to identify the most optimal clustering as well, as more optimal clusters may not differ much in terms of meta-stability. Larger GNN models suffer from a vanishing gradient due to the heavily conditional nature of optimizing all node partition probabilities at the same time, resulting in an inability to find the optimal clustering even once. The best clusterings found by the GraphSAGE and TD models are shown in the appendix in Fig. (2.49).

While further analysis of learning rate adjustments could provide an effective solution to train larger and more complex models on this problem, it is clear that training simple networks is both quicker and more reliable in finding the optimal clustering. Feature comparisons on the GraphSAGE + LD example show the success of the method in finding the optimal partitioning over 1000 individual runs, with all features tested finding it except for the eigenvectors. A breakdown of the distributions for each feature set is shown in Fig. (2.43)(b). MFPTs show the best performance, finding the optimal solution the greatest number of times. Most likely due to the computational simplicity in finding effective mappings from the MFPTs to optimized partitionings, as the value of Eqn. (2.5.4.1) directly depends upon them. The poor performance of the eigenvectors is most likely down to numerical stability, as some of them are very small floating point numbers.

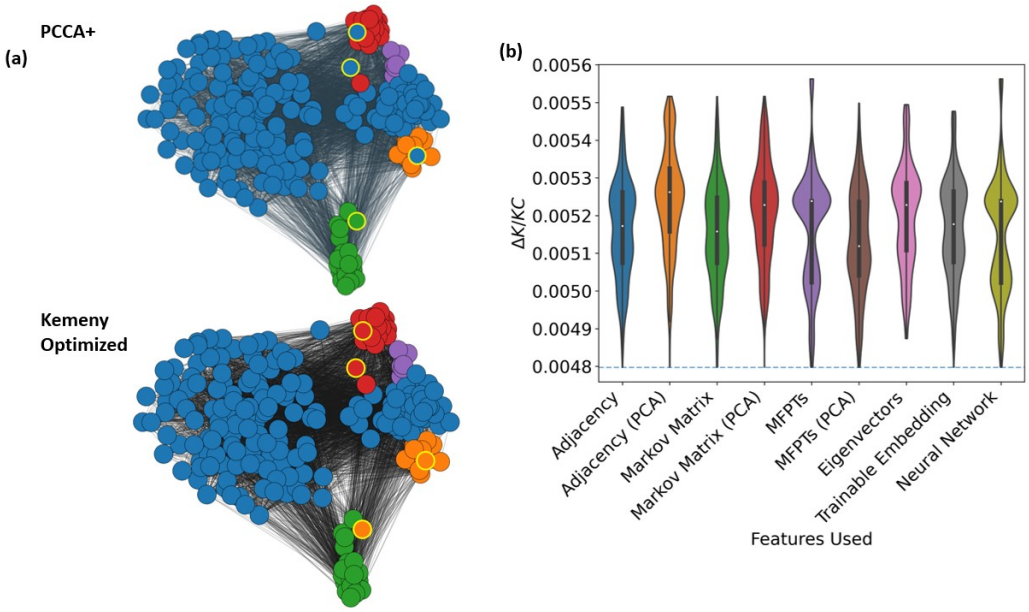


Figure 2.43: ***(a)*** *Comparison of the PCCA+ solution with the optimal ΔK clustering for the pentapeptide system, which has DBI scores of 0.823 and 0.758 respectively. Networks are displayed using a force-directed layout weighted with the MFPTs. Nodes which differ between the two clusters are highlighted with a yellow border.*
(b) *Violin plot distributions for 1000 experiments with each set of nine features. The corresponding distribution of final $\Delta K/KC$ values is shown as a violin plot, along with a blue dashed line for the optimal $\Delta K/KC$ value.*

Fig. (2.45) shows the results of a hyper-parameter search using the best-performing simple GNN model to identify the optimal parameters for the network on the pentapeptide example. It highlights the problem that the network's ef-

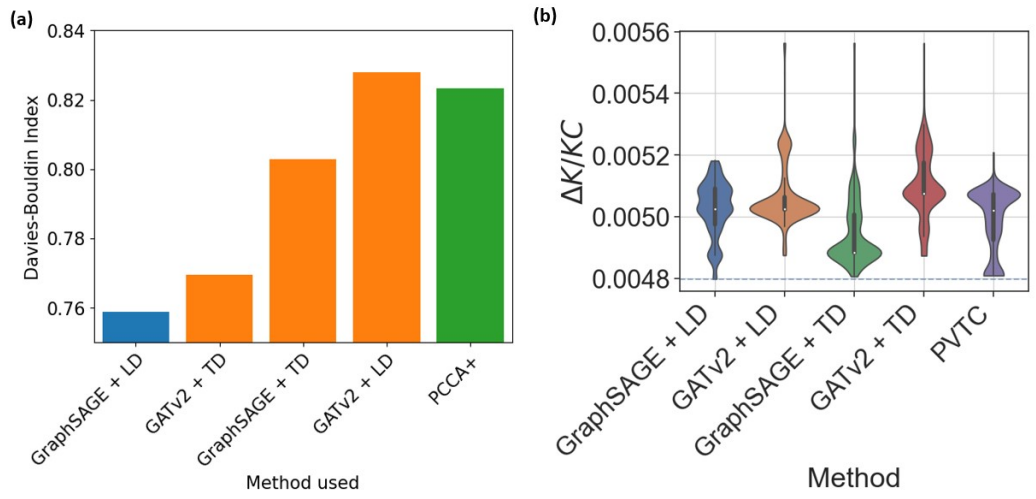


Figure 2.44: ***(a)*** *A bar-plot showing the DBI values associated with the optimal ΔK partitionings found by each method used.* ***(b)*** *A violin distribution plot of $\Delta K/K$ values acquired from training 1000 copies of each neural network architecture on the pentapeptide system. The blue dashed line indicates the value associated with the optimal clustering.*

fectiveness is highly dependent upon the chosen hyper-parameters of the neural network and the size of the network. The best parameter space for a given problem must be sufficiently large to be able to express the wide multitude of possible **S** matrices, while also being small enough to allow for effective optimization in a feasible time. This is shown by the poorer performance of most networks featuring four linear layers in the decoder, where most of the time the method either fails to find the optimal solution or does so rarely it is classified as an outlier. Future work can further verify whether a varying or learnable adjustment to the learning rate could provide a reasonable improvement, as this would allow networks to more efficiently probe the relevant regions of parameter space. Alternatively, further work could evaluate if it is possible to quantify the relationship between the size of the graph being clustered and the most optimal network size.

2.5.6 Conclusion & Future Work

This work conceived and implemented a novel GNN-based clustering optimization method, which used the computing infrastructure and tooling that has risen around ML to provide a working basis for solving future difficult optimization problems on

Model Name	Mean	Minimum	Global Minimum Accuracy
GraphSAGE + LD	271.03 (± 4.80)	**258.67**	**2.8%**
GATv2 + LD	273.28 (± 5.71)	262.81	0.0%
GraphSAGE + TD	**266.66 (± 5.94)**	259.098	0.0%
GATv2 + TD	274.92 (± 5.81)	262.74	0.0%
PCCA+	258.86 (± 0.0)	258.86	N/A

Table 2.4: Network architecture comparison for the pentapeptide system. The four network architectures are compared in terms of the mean (± standard deviation) and minimum value of the KC found, alongside the number of times the optimal solution was found, calculated across 1000 experiments.

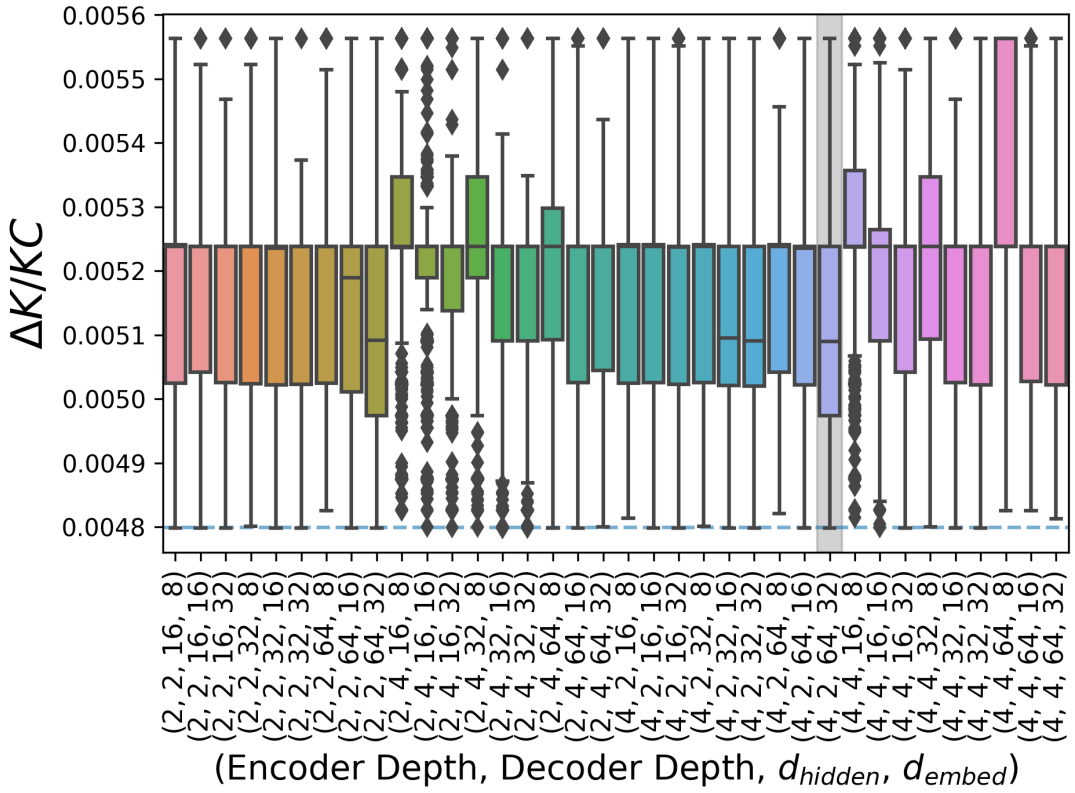


Figure 2.45: Parameter grid search on the pentapeptide example. 1000 networks were initialized for each set of parameters, and the resulting $\Delta K/KC$ distribution is shown. The grey background indicates the parameters identified as optimal.

MCs. The benefits are numerous, as GNNs provide an ample Ansatz for beginning optimization for partitioning phenomena in situations where otherwise finding an efficient representation for the task proves difficult. It also highlights the KC as an important dynamical quantity associated to MCs, and shows how it can be used to obtain metastable kinetically optimal clusters distinct from those provided by existing approaches. Such techniques have been unexplored by much of the

literature, and it is hoped the publication of this work can spark interest in these techniques in the wider community.

The thorough analysis of the proposed method looked at both the benefits and shortcomings of this technique. Comparing the performance of the four different architectures across three different kinetic network types showcased the benefit of smaller networks for this type of work. Node feature comparison highlighted the important role effective features play in GP problems, and how it requires a problem specific choice. The results show that as larger networks become more difficult to optimize, they often cannot find optimal solutions as well as simpler alternatives. GraphSAGE layers combined with a linear layer decoder achieved both greater consistency and accuracy than any tested attention-based alternatives.

Testing the method on SBM graphs generated to have a clear optimal partitioning made contact with available clustering methods. A linear chain corresponding to an analytic 1D free energy profile posed a difficult task for GNNs due to the low connectivity, providing an example of MCs which GNNs can struggle with. It was shown this can be overcome by adapting the node features the GNN uses and the number of runs to alleviate this problem. However, the optimal clustering is only found in about 5% of the attempts, and therefore improving the reliability of the method still requires future work. The outline of a general methodology, introducing the strategy defining generator $\mathcal{I}$, culling $\mathcal{C}$, and termination $\mathcal{T}$ functions, provides an ample framework for future work in this direction. It is hoped that this general methodology can provide a workable basis for future research, and be applied to optimize MC for practical application in the near future.

The MD-simulation derived MSM for a pentapeptide is the most complex example considered here, closest to a practical application of the method. It has a slightly more optimal ΔK clustering that seems to better capture metastable states than the PCCA+-derived clusters, as also seen by comparing the corresponding DBI scores. This highlights the potential benefit of alternate clustering methods. While a GNN approach was shown to be viable, it is yet to be tested on more significant and larger MCs originating from longer simulations of proteins. Clustering these MCs is a current challenge for any technique, and it remains to

be seen how this method will scale with the size of the network and therefore a smaller relevant part of parameter space.

This project explored a number of node features that were used by the GNNs to enable efficient optimization. The MFPT-based features were some of the most successful ones, as these directly relate to the definition of ΔK. However, for examples derived from biomolecular simulations, features that are specifically relevant to physical and structural observables (angles, distances, energy terms, etc.) could outperform the general features used here. This is a promising avenue for future directions, moving towards the automated building of KC optimised clustered MCs. In addition, an interesting application could be towards reducing the dimensionality of Markov models used for protein family classification.

In summary, the validity of using GNNs as an effective method to solve the GP problem applied to MCs has been shown. Alongside this, a general framework outlining the strategy for optimizing partitionings using GNNs in parallel on graph-like structures has been proposed. Such a framework, along with basic GNN architectures, can be applied to solve the many different and difficult non-convex optimization tasks which are commonplace in computational fields using graph structures, taking full advantage of the speed, availability, and efficiency of modern ML tooling and associated hardware. The general training paradigm outlined provides a working basis for generalizing this strategy to different problems, and the results highlight the usefulness of simpler GNN architectures when solving such optimization problems. It remains to be seen how this can be improved by implementing more complex parameter space sampling strategies to aid the identification of more optimal solutions, taking inspiration from similar techniques used for enhanced sampling in MD.

2.5.7 Appendix

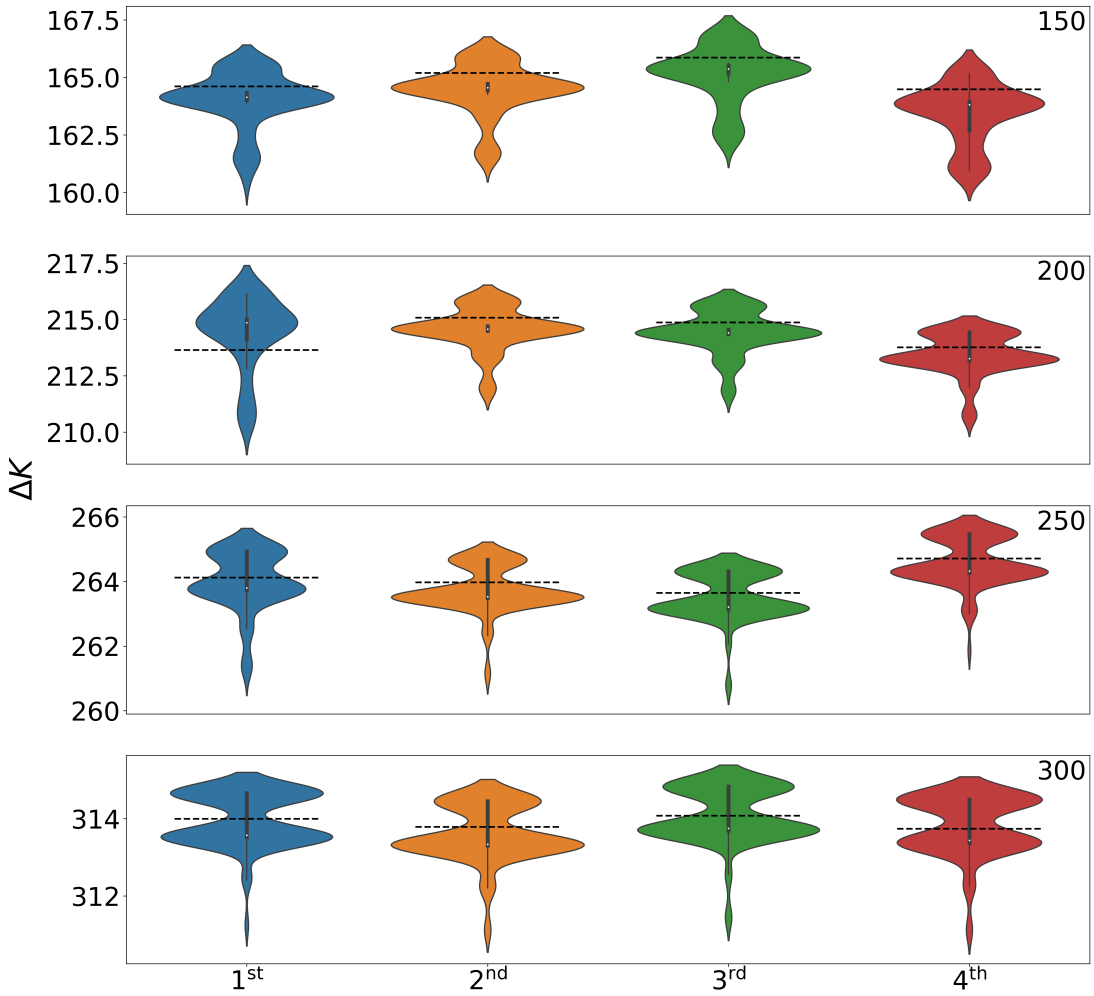


Figure 2.46: A comparison of the ΔK values of the partitionings found by PCCA+ (the dotted black line) and the GraphSAGE + LD GNN method on four SBM networks, where the GNN provides a clustering with the most optimal value. The numbers on the right of each plot specify the number of nodes for the corresponding SBM network. These represent the largest difference found between the KC optimum and the ΔK value corresponding to the PCCA+ clustering.

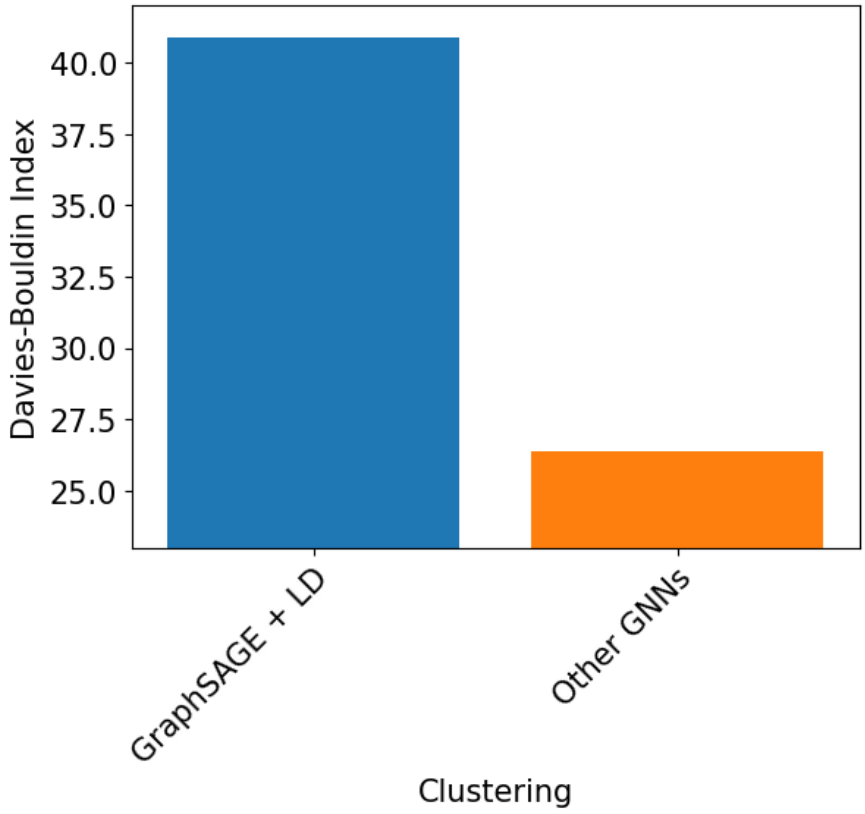


Figure 2.47: DBI values corresponding to clusterings found for the 1D example.

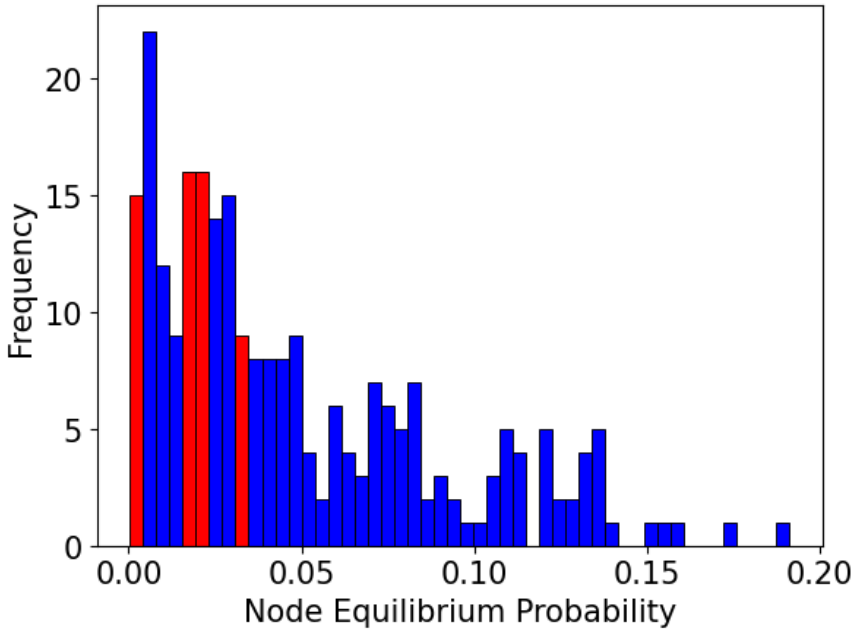


Figure 2.48: A histogram of the equilibrium probability distributions of each node in the pentapeptide example. Red bars indicate the presence of a node which is placed into a different cluster by PCCA+ and the GraphSAGE + LD method. These differing nodes represent 7.4% of the overall node populations.

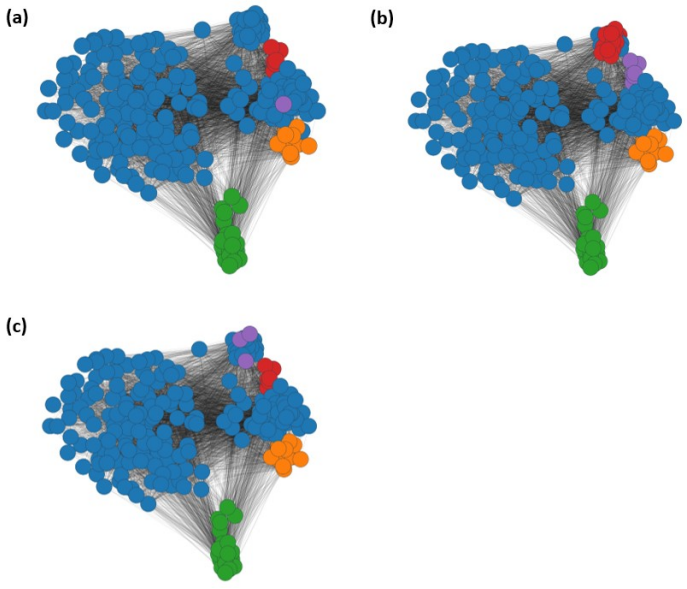


*Figure 2.49: A comparison of the pentapeptide network partitionings with the lowest ΔK value found by each method used. Figures **(a)**, **(b)**, **(c)** show those found by the GATv2 + TD, GraphSAGE + TD, and GATv2 + LD respectively*

Chapter 3

Large Scale Screening

3.1 Introduction

Screening is the process of assessing the suitability of a small ligand for repurposing as a therapeutic compound. In modern drug discovery, this is one of the most difficult tasks, requiring potentially billions of molecules to be assessed. This section will focus on working with small molecules, taking the reader from beginning with a structure containing a suitable identified pocket, to coalescing it with a small molecule similar to a known binder in order to generate a complex. In order to choose the best software for the task at hand, a well-rounded review of what approaches are available is required, which the subsequent methodology section looks to provide. After which it will briefly look at some enhanced simulation options available to estimate the binding affinity of a ligand-protein complex.

While the previous section provides a basis for the initial work identifying targets, there has been no discussion yet of how they may be modulated. Now the focus turns to small drug-like molecules and how to practically deal with and assess their suitability as a BAC. This process always culminates in experimental procedures to test for the desired effect, and this is the greatest and most unavoidable expense required to confirm activity. It is also by far the bottleneck of the overall process, as experimental testing is a long and arduous process which can regularly require several stages before any results can be statistically confirmed. Due to this computationalists play the role of ensuring only compounds with the greatest potential are tested- and this is the main goal of any effective

Drug Discovery Pipeline (DDP). It is a staggered computational process where a significant portion of molecules are discarded at each step.

This section focuses on the computational aspect of developing and running such a DDP in situations where there are no prior Structure–Activity Relationships (SARs), but there are some experimentally confirmed binders available. In other circumstances there may already be empirical information on BACs, which can be refined using regression or classification methods into a system-specific Quantitive SAR model. Simply put, these quantitively relate the structure of a small molecule to its physical effect on a system. This can lead to the development of advanced pharmacophore and fragment-based models. These can be used for the process of Lead-Optimization (LO), where the potency of compounds is increased through chemical alterations. Identification of the groups critical to a drug's function is a viable strategy for drug development, and has found wide successes[472]. While this is a common approach such models will not be discussed further, as when there is little information available on a systems bioactivity the first step instead requires a significantly more general and scaleable approach to lay the basis for LO.

The variety in desired function means general measure for what defines a "good drug" are ill-defined, requiring a significant amount of system specificity. Despite this there are general empirical rules of thumb, such as Lipsinki's rule of 5, the Ghose filter, and Veber's Rule- amongst the many proposed[473,474]. However these are not conclusive as exceptions are easy to find, and generally focus on the pharmacological properties of a compound such as its absorption, distribution, metabolism, and excretion rather than specific activity. No set of rules can mitigate the need for a good pharmaceutical chemist, who is able to identify important interactions and patterns across drug-like compounds via inspection, and there is still no computational replacement for this required expert input. This work rather than discussing the interactions and pharmacokinetics defining a good drug, instead focuses is on how computational practitioners can enable the chemists they work with to use their talents and time as efficiently as possible. This is through quickly identifying potential candidate compounds, disregarding poor leads, and automating as much of the procedure leading up to visual inspection as possible.

To end, 2 Case Studies look to implement many of the discussed techniques. Case Study 4 looks at how the discussed theory and methodology was applied during the CACHE2 drug discovery challenge to investigate small binders on SARS-CoV-2 NSP13. This work entailed building a large generic pipeline to assess the suitability of ligands for this system, for participation in the CACHE2 drug discovery challenge. Case Study 5 takes this pipeline and develops it further, using a mutated G12D RAS protein as a test case. RAS is an oncogenic protein with mutations common across multiple cancers. This project looks to extend the built pipeline to new system specific simulations and to include an additional GNN screening process. It looks to show a clear example of the increased performance of this new process in comparison to using docking scores as a screening methodology.

3.2 Methodology

Computational organic chemistry and therefore drug design requires a vast number of file formats, and this is undoubtedly because the task any comprehensive attempt faces is incredibly difficult. Small molecules resist any natural representation on a computer. Researchers in pure chemical fields may spend their careers trying to understand the electronic structure configurations of a single compound, while a DDP will parse potentially billions in a single day. Therefore it is naive to assume there exists a one-size-fits-all format to appease all researchers and instead it is common to have a variety of formats for use on seemingly the same task. The process of developing a DDP can be seen as moving molecules between these different formats, each providing a different level of detail necessary for the next step. As there is a computational cost associated with pushing any molecule along this process, it is important that a large proportion of unpromising molecules are discarded at each point. A pipeline outputs a quantitive way to asses whether it is worthwhile to move a molecule further along this process. While it undoubtedly has a non-negligible probability to discard promising compounds, the hope is that this iteratively converges on a collection of promising leads.

The main issue is the size of the space of available compounds which is both unfeasible to inspect and incredibly difficult to search. Efficient usage of this data requires significant computational expertise in order to process, and much of the modern work is enabled by heavy parallelization on both CPUs and GPUs. Large libraries of such compounds often come packaged up by chemical manufacturers in the form of Virtual Screening (VS) libraries, such as ZINC or EnamineREAL, which feature well over billions (10^8) of the compounds they are able to produce. These are developed using combinatorial chemistry, where various small fragments are combined algorithmically to create large lists consisting of all purchasable compounds priced depending on the difficulty of synthesis.

Exploring and navigating this chemical space is a real challenge, and assessing compounds requires effectively allocating limited computational resources. The process of screening such a library is known as High Throughput Screening (HTS), and effective techniques for this are on the cutting-edge of current research. Its difficulty is magnified by the curse of dimensionality, as any point in chemical space

is inherently very high-dimensional with the ability to add, remove, or swap any of the atoms for a large number of alternatives. The pure magnitude is also far beyond current computational capacity in terms of memory and bandwidth, as the number of compounds under 500Da satisfying Lipsinki's rule of 5 is predicted to be around 10^{60}[475]. These issues pose many difficulties for computational specialists in the area, requiring a significant amount of interfacing between different software and formats. The order of this section follows this process, beginning with simple text and vector based representations, before moving onto the charged 3D structures required for coalescing ligands with proteins.

3.2.1 SMILES strings & Molecular Fingerprints

The first formats encountered are the smallest, and aim to give researchers the basic outline of a small molecule. These formats are also built from the least information, requiring little more than a specification of atoms and bonds and hence requiring the least computation to produce and memory to store. These formats aim to solve several issues of scale posed by these libraries at some expense of expressiveness, uniqueness, and detail. There are many different approaches available, with each focusing on either a specific quality or computational efficiency.

SMILES strings are the most common format for VS libraries and therefore serve as the common entry point to a generic DDP. Molecular fingerprinting is the next most basic representations considered, as they aim to create numerical representations. The initial stage of the DDP considered is to assess the molecular similarity between a proposed and known binder. This requires the parsing of a SMILES string, supplied by a chemical company, and conversion of this to a numerical format suitable for HTS tasks.

SMILES

The Simplified Molecular Input Line Entry System, or SMILES string, is a human and machine readable ASCII-string based format that allows for specifying compounds in a highly adaptable way[476]. Individually these are shareable, small, identifiable, and interoperable, at some expense of uniqueness and consistency. It finds significant usage as the de-facto format for large databases, where `.smi` files

and their various extensions contain large numbers of molecules along with arbitrary metadata. These formats contain simple delimiter separated data columns. In this usage it is easy to parse programmatically or inspect visually, along with having a phenomenal ability to compress to sizes requiring as little as 0.42 bytes per atom. This results in a database of one-billion (10^9) molecules with an average number of 20 atoms per molecule, being just 8.4GB.

These strings are substituted for more specific and detailed representations in later stages, but still provide an effective format for sharing and initial searches. The original SMILES format specification is proprietary, yet has still become widely used and available using various parsers each with their own specifications. For this reason, a general overview of the format is offered below for completeness, but it is to be noted that there are many alterations, alternate ways of specifying, and extensions which are missed yet commonplace in VS libraries. In any areas of debate, this work falls back to the Chemaxon `.CXSMILES` format specification for consistency with the EnamineREAL VS library used in the further Case Studies[477].

A SMILES string is a language representing a depth-first traversal of a molecular graph excluding hydrogens, where each character signifies an atom and a subset of punctuation is used to specify chemical properties or phenomena. The choice of start node is arbitrary, leading to an innate redundancy in the format. The most common atoms in organic molecules: B, C, N, O, P, S, F, Cl, Br, or I, can be represented using these abbreviations so long as they have no formal charge. Any unfilled valence implied from this will be assumed to be filled with a hydrogen atom. When this is not the case, or generally for rarer elements, atoms can be specified by enclosing their two letter abbreviations in square brackets `[·]`. Atoms with charges which can be defined this way by suffixing a number of `+` or `-` symbols denoting the charge, and additionally the number of bound hydrogens can be made explicit by using `Hx` as the suffix where `x` is the number of bonded hydrogens. Although rarely used for the majority of drug-like molecules, isotopes of different elements can be specified when using square-brackets by prefixing the element symbol with the atomic mass. Single bonds are implicit for adjacent characters or denoted with a hyphen `-`, and double or triple bonds can be specified

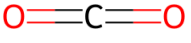
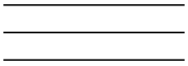
SMILES string	Chemical Name	Chemical representation
`C`	Methane	
`CC` / `C-C`	Ethane	
`O=C=O`	Carbon Dioxide	
`C#C`	Ethyne	
`[NH4+]`	Ammonium	

Table 3.1: SMILES representations of simple molecules.

by `=` and `#` respectively. There are no spaces or gaps allowed in a SMILES string. Examples of basic SMILES are shown in Table 3.1:

For more complex molecules, additional rules are required to deal with branching, rings, and stereo-chemical properties. Branches are handled using parentheses `(·)`, where the internal string is also a valid SMILES. These can be nested, and are always read to be associated to the preceding atom. For bonds other than single, the bond type specifier must be included as the first character of the branch. Simple cyclic structures, such as cyclohexane, can be described by breaking a single bond and following it by a numerical identifier. A second occurrence of the same number later in the string shuts the ring. Several rings may be specified in a single SMILES and any number is a valid identifier so long as it appears twice. Aromatic compounds pose a special case, as cycles can be represented by several different contributing structures. To display these, it is common for many parsers to use the lower-case forms of the usual atomic symbols along with a ring identifier. Alternatively it can be simply written as if it was aliphatic as there is no strict requirement to specify a ring as aromatic, leaving the work of checking to later algorithms. A less frequently seen but equally valid approach is to specify each bond using colons `C:C`. Examples of branched and ringed smiles strings are shown in Table 3.2.

Much like aromaticity, there is no strict rule requiring the definition of stereochemical properties of compounds and not including any still generates a valid SMILES string. Users who wish to show the configuration around double bonds may do so by using pairs of slashes `/` or `\` where two are required on each atom

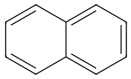
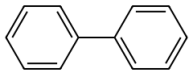
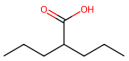
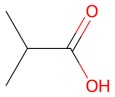
SMILES string	Chemical Name	Chemical representation
`C1CCCCC1`	Cyclohexane	
`C1=CCCCC1`	Napthalene	
`c1ccccc1c2ccccc2`	Biphenyl	
`CCCC(C(=O)O)CCC`	4-Heptanoic Acid	
`CC(C)C(=O)O`	Isobutyric acid	

Table 3.2: SMILES representations of ringed and branched molecules.

either side of the specified bond. Two of the same slash denote a trans isomer and alternating slashes denote cis. Chirality is additionally another optional property. In cases where it is important to be specific, the central chiral centre atom may be specified with either a `@` or `@@` symbol. Some examples of complex molecules, taken from drugs mentioned in this work, are shown in Table 3.3

The SMARTS extension to the language is akin to regular expressions used in text parsing[478]. SMARTS strings extend the SMILES specification to include support for wildcard matching on atom and bond types alongside general logical operators. These include wildcard characters for specifying any atom using an asterisk `*` or any bond with a tilde `~`. Full lists of all primitives are available in the SMARTS specification. This allows for a robust method to search for substructures, and allows chemists familiar with the SMILES format to quickly find molecules fulfilling basic chemical or structural criteria.

SMARTS strings also allow researchers to parse for problematic compounds. These include those which show non-specific binding which would show promising results on binding-based experiments and calculations, but which would be unsuitable for any therapeutic purposes. This is because their capability to bind to

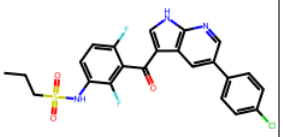
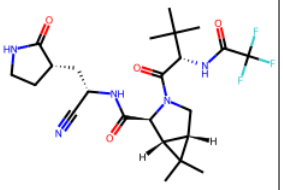
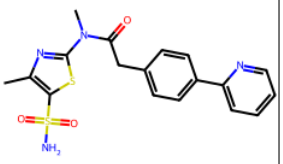
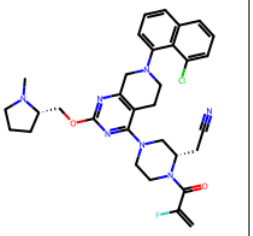
SMILES string	Chemical Name	Chemical representation
CCCS(=O)(=O)Nc1ccc(F)c(c1F)C(=O)c2c[nH]c3c2cc(cn3)c4ccc(Cl)cc4	Vemurafenib	
CC1([C@@H]2[C@H]1[C@H](N(C2)C(=O)[C@H](C(C)(C)C)NC(=O)C(F)(F)F)C(=O)N[C@@H](C[C@@H]3CCNC3=O)C#N)C	Nirmatrelvir	
Cc1c(sc(n1)N(C)C(=O)Cc2ccc(cc2)c3ccccn3)S(=O)(=O)N	Pritelivir	
CN1CCC[C@H]1COC2=NC3=C(CCN(C3)C4=CC=CC5=C4C(=CC=C5)Cl)C(=N2)N6CCN([C@H](C6)CC#N)C(=O)C(=C)F	Adagrasib	

Table 3.3: SMILES representations of some complex drug molecules mentioned in this thesis, featuring chiral centres.

any protein would have damaging and unpredictable consequences inside a cell. Such compounds are known as promiscuous, and it is inevitable that they are found during HTS simply due to the number of molecules screened[479,480].

Despite the clear advantages of SMILES strings as a small and shareable language for distributing molecules, they have a significant number of shortcomings especially when considering their usage for more complex tasks than storage. Due to the arbitrary choice of a starting node there is a high degree of redundancy and various SMILES string generators or proprietors use different algorithms resulting in different possible strings. In addition as they are implicit about

hydrogen atoms and prone to inconsistencies with chiral structures, software using SMILES strings can be prone to producing the wrong compounds or missing key properties. There are many examples above of optional parameters, and due to each piece of software outlining a different specification this is prone to parsing errors. Therefore most the issues surrounding using SMILES are due to inconsistencies in the algorithms between software and redundancy in the specification itself. This leads to downstream problems as it is common to find structures with structural issues.

To fix these issues the community needs to accept one implementation which is difficult due to the large amount of proprietary code used in the field. Additionally a major issue for using SMILES strings for further purposes is that there is no natural distance metric which can be used to compare different SMILES strings. Natural string metrics such as the Hamming distance or Levenshtein metric have no bearing in compound space, and do not comply with basic chemical principles defining molecular similarity. Therefore additional computation is required from these strings to be able to utilize the information they contain for searching and measuring similarity. Despite this, various ML approaches have seen SMILES strings as one of the first chemical languages where researchers have applied tokenization and other NLP principles to. This could be a promising interface between LLM-based models and chemical space[481].

Molecular Fingerprints

While now it is clear how large numbers of molecules can be represented on a computer as strings, it still remains to use them for a functional purpose. At these early stages where comparison between molecules is the main focus, SMILES strings and similar formats lack the natural form to make this process efficient. The relative overhead needed to read, parse, and compare SMILES strings already puts the pairwise comparison of millions of molecules outside the realm of feasibility. Molecular fingerprinting aims to make this significantly easier by assigning a numerical bit-vector to a molecule in order to reduce the complexity of the similarity operation[482,483].

The bit-vector data structure reduces much of the similarity operation to a dot product, which is easily handled in large volumes by SIMD hardware such as the GPU. A successful molecular fingerprint is as small as possible to enable a large number to be processed at once, and allows for a natural, computationally simple, and effective similarity/distance metric. Although there is still a degree of additional processing required to calculate each vector, this only needs to be performed once and allows for repeated subsequent searching. Naturally, the scope of such techniques is incredibly broad as it is dependent upon the information initially available and the purpose of the search, which can vary greatly. Some fingerprinting techniques use 3D-coordinates and properties of the target structure- such as pharmacophore approaches- and while they find usage these will not be discussed. Instead, this will focus on generic approaches valid for all targets and molecules which are capable of scaling effectively and generated from just the SMILES string. Most of the well-known fingerprint generation algorithms then fall into the substructure-key, topological, or circular based algorithm categories.

The individual bits in a substructure-key bit-vector represent the presence of a specific substructure or atom in a molecule, such as a specific functional group, bond, or ring. The naming convention relates each bit in the vector to the key in a hash-map, with 1 indicating the molecule contains the property known as a key or a query. Keys can include any query one can ask of a molecule which yields a binary answer, giving a large scope to the uses of these fingerprints. These can be generated at a high speed and low computational overhead, for example by a regular expression parsing of a SMARTS string across a SMILES database. The dot-product similarity measure then indicates the number of shared substructures between the two chosen molecules, and is useful for a specific indication of chemical similarity. The length is determined by the number of queries. For searches where the presence of certain motifs is required or prohibited, it is trivial to generate a custom fingerprint from basic SMARTS strings. Otherwise for general searching purposes there are a number of widely used substructure-key fingerprints that look for established criteria suitable for DDP purposes.

These include the MACCS family, which comes in either 960 or 166 length bit-vectors[i]. Originally invented for basic substructure screening, these were picked up as some of the first types of fingerprints and later optimized for similarity searching[485]. While there are many variations to these they differ only in length and the features used, and it is important to note the impact a diverse and potentially complex set of keys will have on the speed of the fingerprint generation process. The numerical values may not be useful as a direct similarity metric as one would expect. A small molecule with just the valid keys will score the same as incredibly large molecules featuring significant distance between the different keys. Also when working from observed binders the SAR is much more complex than simply the presence of a sole functional group. Therefore their effective usage is mostly when applied to systems which have a known SAR.

Topological fingerprints were designed provide a general comparison by instead focusing on the unique paths contained within molecules. In doing so, they culminated in avoiding having to use any well-defined criteria or queries altogether. Initial implementations conceived in the 1980's used the concept of atom pairs, collating every unique non-hydrogen pair of atoms in the compound along with their basic properties and minimum separation in terms of bonds between them[486]. These pair counts could then be compared computationally against other molecules using a variation of the Manhattan metric calculated from how many pairs are shared. An alternative early approach instead focused on the unique "torsions" of a molecule, or sets of 4 atoms similar to how atomic dihedral angles are calculated[487]. Approaches such as these lay the groundwork for representing molecules based upon their constituent neighbourhoods by calculating linear bonded chains of atoms. As the methodologies developed, the choices of how many atoms constitute a primitive fragment have become the defining difference between different methods. Many modern schemes, such as the RDKit and Daylight implementations, leave the choice to the end user who can dictate the minimum or maximum sizes of the subgraphs considered, which can be adjusted to specialize the fingerprint to smaller or larger molecules. These fingerprints also do not consider repetitions

[i]There is contention however to do with the implementation and specification of this format[484].

of the fragments, leading to a simple measure that may not be suitable for long chain molecules[488]. They find wide usage for substructure screening.

Circular fingerprints can be considered a subset or extension to topological fingerprints, sharing the same focus on studying the constituent fragments of molecules. They build upon the path concept by instead using radial substructures as molecular building blocks rather than linear chains. They are often referred to by the name Morgan fingerprints, stemming from the author of the original paper which introduced a combinatorial radial numbering system for representing molecules in the 1960's avoiding the molecular isomorphism problem[489]. This idea was later repurposed for similarity uses under the name Extended-Connectivity Fingerprints (ECFPs), finding success in molecular similarity for drug design[490]. The main concept underpinning them all is that by specifying all radial atomic neighbourhoods within a molecule, it is possible to measure similarity by again counting the number shared. Despite the nomenclature confusion, the main difference is that a classical Morgan fingerprint exhausts all possible neighbourhoods whereas ECFP fingerprints are specified to an integer denoting the maximum radius considered. For example the common ECFP4 works up until every region of a diameter atoms up to 4 bonds away are considered. Again this limit can be set depending upon the size of the molecules required or the size of any functional groups or motifs of interest, and is purpose-specific.

While both topological and circular fingerprints are clearly better suited to finding similar compounds without specific requirements, there is the issue of variable sizes which has been circumvented thus far. As queries for a fixed-length substructure-key fingerprints are predetermined, it is guaranteed that all will be of the same size. However for topological and circular fingerprints different molecules will inevitably consist of a differing number of fragments no matter the choice of neighbourhood considered.

The common method for taking a fragment count and arriving at a bit-vector is to use an arbitrary hashing function $\mathcal{G}(\cdot)$. The first step in generating any topological or circular fingerprint is to assign each atom in a molecule a integer label. This is done by applying the hashing function $\mathcal{G}$ to any atomic property, for example the Daylight atomic invariants used in both Daylight and ECFP generation. Once each atom is labelled, a feature array is constructed for each

atom. Different fingerprints determine the form of this array, for example it consists of each neighbourhood atom along with an identifier for the bond type in ECFP generation. In topological fingerprints, this process involves gathering labels for each unique path along the molecule. In most implementations, this feature array is first checked to be unique and if so it is then hashed using $\mathcal{G}$ to construct a new identifier assigned to the atom. If the feature vector has already been included via another atom's neighbourhood, it is discarded. Repeating this process n times means each iteration the label features information from another bond-distance away, drawing analogies with information spreading procedure in GNNs.

Once all labels are established, the bit-vector may be constructed in a process known as fingerprint folding. Initially a bit-vector of arbitrary length M is initialized with all 0's. The modulo of each hashed label with M is then taken, and the element of the bit-vector corresponding to this remainder is set to 1. This is why bit-vector representations cannot account for the number of times a neighbourhood occurs. If this is desired many relevant libraries allow for feature array representations to be used instead for non-high throughput operations.

While this scheme enables the usage of arbitrary length bit-vectors as effective fingerprint representations, it also compromises the certainty of the measure when compared to substructure-key methods. The chosen length of the bit-vector determines the probability of a bit-collision occurring, that is two different fragments corresponding to the same bit. This means that while a bit difference in the same index decisively means there is no shared fragment, a common bit is no guarantee it was generated by the same structure. Increasing the size of the bit-vector decreases the collision probability, and so it is up to researchers to quantify how much sensitivity to uncertainty is acceptable. The size must also be suitable for the hardware the search is run on. For searches based on GPUs, it must be taken into account that all fingerprints must reside in onboard RAM, therefore a size must be suitable for a large number to reside in device memory. Commonly sizes of 1024-bits are sufficient in DDP uses, as this allows speed which is often more important and the collision probability is still effectively low at this size. In an idealization with little memory overhead, such a size allows for just over $200,000,000$ compounds to be represented on a card with $24GB$ VRAM.

Fingerprint generation is a significant performance bottleneck of any DDP, requiring a significant amount of time and resource allocation to initialize a database of suitable size before any molecules can be screened. The benefit is that this process only has to be done once and may be transferred to other projects afterwards. The time taken is also minimal when compared to algorithms requiring 3D-structures or shape based comparisons, and the space required for storage is miniscule. There are no algorithms which generate fingerprints directly on the GPU. This may be down to the staggered nature of the process for different molecules, leading to a less significant speed-up and potentially complicated control flow. This serves as a possible target for further GPU-based optimisation in DDPs.

Similarity Metrics

Once a set of fingerprints have been generated for a large number of molecules, their similarity can be assessed using a variety of metrics[491]. In order to differentiate between small molecules of different sizes, care must be taken on how to construct such a measure. Despite the nomenclature often these do not qualify as full mathematical metrics, and the term is instead used as an identifier for a way of measuring some sense of similarity. The choice of form to use has also been shown to have a dramatic effect on the performance of a search[492]. Pairwise comparison metrics between fingerprint vectors $\mathbf{A}$ and $\mathbf{B}$ of length M generally utilizes 3 key quantities calculated from the two:

$$a = \mathbf{A} \setminus \mathbf{B} = \sum \mathbf{A} - \mathbf{A} \cdot \mathbf{B} \tag{3.2.1.1}$$

$$b = \mathbf{B} \setminus \mathbf{A} = \sum \mathbf{B} - \mathbf{A} \cdot \mathbf{B} \tag{3.2.1.2}$$

$$c = \mathbf{A} \cap \mathbf{B} = \mathbf{A} \cdot \mathbf{B} \tag{3.2.1.3}$$

$$\tag{3.2.1.4}$$

Where $\setminus$ represents the set difference and $\cap$ represents the intersection between two sets. All these are calculated via the dot-product, which is efficient and simple to implement on GPU hardware using bit-wise operators.

Similarity Metric	Formula	Bounds
Tanimoto Coefficient	$\frac{c}{a+b-c}$	$[0,1]$
Dice-Sørensen Coefficient	$\frac{2c}{2c+b+c}$	$[0,1]$
Cosine Coefficient	$\frac{c}{\sqrt{(a+c)(b+c)}}$	$[0,1]$
Euclidean Distance	$\sqrt{\sum_{i=1}^{M}(B_i - A_i)^2}$	$[0,M]$
Manhattan Distance	$a+b$	$[0,M]$

Table 3.4: The equations of some commonly used similarity metrics.

The Tanimoto coefficient, better known as the Jaccard index outside of cheminformatics, is the most common metric used for similarity searching. Originally devised for measuring the similarity of plants, it is a simple measure applicable to any two bit-vectors which was later shown to be equivalent to the Jaccard index. It is the ratio between the intersection and union:

$$T(\mathbf{A},\mathbf{B}) = \frac{|A \cap B|}{|A \cup B|} = \frac{c}{a+b-c} \tag{3.2.1.5}$$

Dividing by the intersection means this is bounded between $0 \geq T(\mathbf{A},\mathbf{B}) \geq 1$, with a value of 1 denoting $\mathbf{A} = \mathbf{B}$ and 0 they are complements of each other. Despite this, as stated above due to the folding process this does not mean the molecules themselves are identical. Its simple form accounts for its popularity, making it incredibly efficient and scalable.

Table 3.4 shows an outline of various other metrics that have found usage for similarity searching in terms of the quantities outlined in Eqn. (3.2.1.1). It makes a distinction between coefficients which are usually bounded between 0 and 1 and quantify similarity, and full metrics which are limited by the size of the fingerprint and measure the dissimilarity. Both the Euclidean and Manhattan distances qualify as full metrics by satisfying the triangle inequality, imbuing the space with a geometry suitable for comparison between more than just a pair of molecules. This can lead to them being used for more complex uses, such as clustering or topological analysis. They can both also be implemented as normalized versions, but may be better suited to not be in use cases where molecule size is important as they can quantify the difference in molecules size. The coefficients are instead

limited to pairwise comparison, but their simple expressions and intuitiveness make them significantly more widespread in DDPs. The Dice-Sørensen and Cosine coefficient differ in that they compare the bits in common to either the geometric or arithmetic mean respectively. Both are more computationally expensive than the Tanimoto coefficient due to their denominators and as there is no clear theoretical advantage to using them for screening, they find less widespread usage.

3.2.2 Structure Generation

Once compounds similar to a known binding molecule or fragment have been identified, the next step is to move past string and bit-vector based representations and begin to build 3D-coordinates for small molecules. Naturally such a leap requires a significantly greater amount of information and computation time. Mapping from SMILES strings to 3D-structures is a complex process with various different approaches all with associated pitfalls. It is decisively not injective, and programs can often produce several viable structures for a single molecule. This is due to conformational diversity due to chiral centres or rotatable bonds, protonation states, and the aforementioned inconsistencies in the SMILES specification.

These reasons often lead to a significant number of generated structures from a single SMILES string, increasing the number of compounds sent for downstream testing. As with the strings themselves, this is a fallible process and dealing with errors from 3D structure generation software is frequent especially in larger molecules[493,494]. It is common to see unrealistically angled bonds, deformed rings and groups, and bad protonation states, all of which can fail to be picked up until the final stages of expert inspection. Effective ligand modelling is seen as a critical open problem in drug discovery, and many studies have focused on establishing comparisons between experimental BAC structures and automatically generated conformers[495,496]. Therefore it is imperative for all practitioners in the field to understand the underlying approaches in order to identify and avoid the common errors.

The simplest way to generate a realistic and viable conformation for a small ligand is to ask a talented chemist to create one by hand. For a VS library consisting of billions of structures, this is not viable or worth the time to do so.

Generation of the complexed ligand-protein structure is commonly handled by the docking software, and there is evidence pointing towards minimum energy conformations being far from biologically active ones[497]. Therefore at this point a structure is needed to simply provide input to downstream algorithm, and an automated structure generator needs only to be relatively realistic and free from significant errors preventing later optimisation.

Even this is a large challenge, and it must be taken into account there will be unavoidable errors when using any automated strategy[498]. Several comprehensive reviews cover both analysis and rankings of the most prevalent 3D-conformer generation tools, most commonly by comparing their predictions to crystallised structures of bound known BACs[499]. Here an overview of strategies underlying the common approaches to 3D structure generation will be discussed, and readers will be pointed to the relevant literature covering each approach. General strategies to automatic structure generation can be classified into rule-based, data-based, or numerical, with most modern approaches being a conglomerate process combining steps from each[500].

Rule-based methods attempt to implement chemical knowledge as a set of explicit pre-defined rules, defining explicit characteristics for specific bonds and functional groups. This covers some of the earliest approaches of automatic structure creation, where a systematic approach was taken to map out the common conformations of motifs and define comprehensive rules for bond lengths and torsional angles. Generating a structure then became a two-fold process of identifying the component subgroups and recalling either the relevant pre-calculated conformations or rules governing its position. Subgroups are identified by dividing a molecule based on rotatable bonds into smaller fragments.

Methods such as CORINA and CONCORD exemplify this early approach, and both look to produce a single low-energy conformer for a given molecule using complete set of rules[501,502]. It derives positions from stitching pre-optimised parts together, but still had issues with the relative orientation of these parts. Dealing with these issues often means introducing different approaches, such as in CORINA where a supplementary optimisation using a simplified "pseudo-MM" forcefield is used. Such approaches find difficulty effectively cover all of chemical space, especially as across chemistry there are a large number of edge cases and modern

VS libraries push this to the limit with fragment-based synthesis. Alongside this, these early approaches were later shown to be naive in assuming a single conformer can effectively represent all possible bioactive conformations for a BAC. Over time, rule-based approaches largely became subsumed by more heuristic approaches, while still regularly finding use as a starting point for optimisation or as a screening mechanism to remove poor conformers.

Data-based methods operate on a similar principle of decomposing a molecule into constituent parts, however differ in the aspect of deriving the resulting positions from experimental or ab initio QM fragment structures. Several open databases of small fragment conformations exist, including ChemDB, The Cambridge Structural database, and the Crystallography Open Database to name just a few[503–505]. Once the constituent parts are identified, the fragment database may be queried using a SMARTS search and positions mapped onto the molecules atoms.

Several implementations of this strategy are available, and differ mostly in the databases they query or the additional methods used to determine relative fragment orientation. Methods such as the popular OBabel library revert to a rule-based strategy to determine orientation and non-matching positions once experimental fragments are stitched together. While methods such as BCL::Conf move to Monte-Carlo sampling the torsions depending on steric clashes and a custom scoring function[506,507]. Alternatively methods such as OMEGA optimise positions using a modified force field with the electrostatic and Van der Waals terms removed[508]. Data-based libraries provide an easily justifiable approach to conformer generation more free from potential for edge cases than rule-based methods. They additionally benefit from requiring only an evaluation of a corresponding SMARTS string once ad database is constructed, often making them more computationally efficient[509]. However they are only as good as the data they are built from, and again the nature of VS libraries can often push them towards regions where they have very few viable samples to draw from.

Numerical approaches find usage across nearly all techniques. Distance geometry provided an early starting point for such techniques and still finds usage today[510]. Positions are derived by finding solutions to strict restrictions on upper and lower limits of bond lengths, requiring only atom and bond types to be prede-

fined. This strategy has rigorous mathematical background, and was widespread in the early days due to its relative computational simplicity. As computational speed and memory increased many numerical approaches similar to those considered in Section (1.3) became more prevalent. Relying on optimising atomic positions using an underlying FF, these encompass approaches from classical MM to detailed QM approaches. The general strategy is to find low-energy conformations of molecules through optimising the atomic positions relative to some energy function.

QM approaches are the most detailed, expensive, and accurate methods possible[511]. Using methods at this level of detail, such as DFT calculations, has been shown to be able to calculate some of the most physically accurate and diverse conformers[512,513]. However the computational cost of even the most basic of these puts them outside the realm of feasibility for screening purposes due to their inability to scale. Despite this several quick and approximate methods exist, and find the majority of their usage in the final optimisation step for other methods[514,515].

The MM approach finds itself as the dominant strategy due to its computational efficiency. Its focus is on using classical and empirically parametrised FFs to find stable configurations, by defining parameters for each atom type encountered. Several FFs exist for such purposes[516]. These include aptly named MM family consisting of MM2, MM3, and MM4, derived from large scale ab initio calculation of small molecules[517–519]. The MMFF94 FF is similarly parametrised via QM calculations, however it aims to be applicable across biological systems up to the protein scale, giving it a specific focus for drug design simulations of bound ligand states[520]. The OPLS family also find significant usage for these tasks[521]. These use a functional form very similar to the AMBER FF discussed previously, supplemented with various CHARMM parameters for alkanes as it gave better reproduction of ab initio results specific to small molecules.

Using comprehensive FFs such as these allows for a full range of simulation techniques to be used, including water molecules and the protein pocket. However such computational costs are generally saved for ligands which have passed the docking stage, and at this step the most common usage is for brief atomic position optimisations. GPUs have allowed these methods to scale exceptionally,

for example by achieving a factor of 55× speed-up for the MMFF94 making it more viable for large-scale screening tasks[522].

An additional required step is the prediction of the protonation states of small molecules, and this can take place before or after coordinate generation depending on the method used. Hydrogen atoms have so far been ignored, however their presence can have dramatic effects on the dynamical and electrostatic properties of a small molecule and its propensity to bind. Biological pH is commonly taken to range from around 7.35 to 7.45, although this can be different depending on the localisation of the system considered. At this pH multiple probable states with different hydrogens bound as possible, known as protomers of a compound[523]. In the cell these protomers can interconvert between each other, and therefore any thorough strategy must test all probable configurations. Predicting the presence of a hydrogen atom requires estimating the dissociation constant K_a, or its commonly used logarithm pK_a, by comparing the equilibrium prevalence between a proton H^+ dissociated from the molecule A^- and its associated form HA.

$$pK_a = -\log(K_a) = -\log\left(\frac{[A^-][H^+]}{[HA]}\right) \tag{3.2.2.1}$$

where square brackets denote the equilibrium molar concentrations.
Comparison between this value for all possible protomers allows for calculation of the most probable states observed at a specific pH, and there are several computational methods to do so.

As before, estimating this is most accurate using QM methods which fail to scale to the sizes required for screening[524,525]. Therefore many methods have emerged to provide a estimations at various levels of trade-off between accuracy and speed. Classical approaches are often based upon the Hammett-Taft method or a similar empirical approach, requiring an existing database of both ionisable groups and their equilibrium constants[526]. Pattern-matching methods such as SMARTS can then be used to locate ionizable atoms within a molecule. The patterns unperturbed pK_a value is then linearly modulated by experimentally derived fixed parameters representing how much each bonded group in the local neighbourhood is predicted to affect this value[527]. There are several existing implementations of such methods, and while theses techniques are well established and in use for

decades, it suffers from a simplistic approach incapable of representing screening effects from multiple groups. This step is vital to an effective VS search, as wrong determination of a ligands charge can have drastic effects on its binding affinity.

Over time, the community has adapted to using methods which encompass facets of all discussed approaches in order to maximise conformer diversity and accuracy. The Schrödinger LigPrep/ConfGen software suite provides one such example, finding widespread adoption throughout the VS community for a complete approach to generating 3D-structures[528]. Its pipeline begins with the molecule being put through Epik for protomer generation[529]. ConfGen is responsible for conformer generation from the protomers, using a data-driven fragment based approach as its first step. This stage looks to identify rings in the molecule against a database of over 1000 template ring conformations with predefined energy values. These ring templates are found via a comprehensive minimisation procedure using the various different FF models for completeness, before being hand-checked. Afterwards all rotatable bonds are enumerated and the OPLS FF is evaluated over a grid of equally spaced angles spanning a complete rotation of each rotatable bond. This is then put through a rule-based filter to prevent steric clashes, before the remaining conformations are Monte-Carlo sampled and scored using an heuristic scoring function. After a selection of low-energy conformers are generated, a final step of further OPLS based optimisation is performed. All this can be performed in a matter of a second per ligand on a single core, highlighting the importance of all 3 historic strategies when it comes to creating an efficient and effective structure generation pipeline.

The structure generation problem has received substantial attention from the ML community, as it provides a novel application of generative models. These include applications of normalizing flows, RL approaches, language-based, diffusion, and equivariant models in order to generate conformers[530–534]. Such approaches offer an incredibly efficient way to sample from large molecular latent spaces when compared to classical approaches, enhanced by the intrinsic parallelism of sampling from such models. Protonation state prediction serves as an example where ML models have become one of the leading methodologies, with the popular Epik pK_a prediction software swapping to using a GNN-based approach in its latest revision[529]. This provides a computational speed up with the GPGPU-

enabled GNN version of Epik taking $0.072s$ per molecule in comparison to the previous versions speed of $0.106s$, while additionally attaining greater accuracy than it[528,535,536]. Similarly ML based approaches to FFs look to make QM level accuracy possible with a classical cost by extrapolating from ab initio datasets[537]. Overall nearly all tasks in the structure generation pipeline show a clear potential for benefits from using generative and predictive NN models, but as of writing such techniques have not been around long enough to receive widespread adoption. With the amount of focus from the biotechnology and wider ML community this is set to change, and methods able to capitalise on the speed and tooling capabilities of modern ML are on the precipice of becoming commonplace.

3.2.3 Docking Software

Once 3D-structures have been generated for a molecule, the target protein can be reintroduced as focus turns to predicting how the two structures coalesce in a process called molecular docking. While a similarity search may screen around 90% of library as having no significant relationship to a known binder, at these scales this will still result in millions of compounds which require further testing. This is often further increased once a structure generation pipeline produces all possible conformers and protomers. In order to gauge their suitability, the first assessment of compatibility between their 3D-structure and the proteins conformation must now be calculated in a much more computationally demanding process than anything considered so far.

In the early days of rational drug design, computational chemists used interactive molecular graphics programs to move molecules and proteins into positions deemed suitable by hand[538]. Looking for suitable hydrogen bonds, π-π stacking, and other non-covalent stabilizing interactions, they sought to identify by observation whether a molecule had a good chance of binding[539]. Modern scales are far too large for such a bespoke process, and docking software has emerged to try to automate these procedures[540]. This is an incredibly diverse procedure with a significant number of approaches, as over 100 competing methods currently find usage[541]. Subsequently a comprehensive overview covering all approaches is beyond the remit of this work, and a general overview of the various strategies

will instead be provided along with specific examples for the software used in this work.

Fortunately, most modern docking software share a similar strategy, consisting of two directly linked stages[542]. The first is the generation of a complex by binding a small molecule to a protein. This requires an efficient sampling scheme in order to explore a significant amount of the relevant conformational space in a feasible time, and can utilize a variety of schemes such as Monte-Carlo approaches and evolutionary algorithms. Secondly is the determination of a scoring function, which provides a ranking of different conformations of the same ligand alongside different ligands. Most software approaches to this problem provide a unified approach, providing both a sampling method which modulates the structures in order to optimise the scoring system. The value score here is relatively arbitrary as it is implementation specific, and much of the difficulty of choosing or rating different docking software comes from the inability to compare different scoring systems.

In essence, a good score should be correlated with the binding energy ΔG of a ligand-protein complex. This may be defined as simply the difference between the free energy of the bound and unbound states of a given complex:

$$\Delta G = G(\text{bound}) - G(\text{unbound}) \tag{3.2.3.1}$$

The lower this value, the greater the stability of the structure and hence the more likely it is to occur.
Binding free energies are incredibly difficult to calculate, as despite its commonly small value it arises from potentially very large contributions of both entropic and enthalpic effects. For example the formation of strong bonds or ionic interactions between the protein and ligand are exothermic, decreasing the enthalpy and contributing to a favourable ΔG. However these same interactions may lead to a loss of flexibility in the system, leading to a loss of entropy and a less favourable value. This entropy-enthalpy compensation complicates direct calculations of binding free energies, and has led to a diverse range of strategies to accommodate for this at all computational time-scales. Direct methods to estimate this more

accurately are explored in Section (3.2.4), however these do not provide methods to generate complexes and are far too resource intensive to extend to these scales.

Instead docking software must again strike a delicate balance between speed and accuracy, and the different approaches exist at different points along this continuum. They most commonly verified by determining how well they can reproduce databases of experimental structures through measuring the RMSD between the in silico and experimental results, and many such reviews exist comparing methodologies[543–546]. Although this of course raises questions for the support of such comparison criteria when compared to results across a diverse VS library, along with the great impact of conformer, tautomer, and sampling strategies[547]. To somewhat mitigate these issues, recent studies have proposed using several different docking software and studying the consensus between them, achieving an increased accuracy and reliability[548,549].

While there are critical technical and implementation differences amongst docking software, the most common way to categorise modern approaches is by the level of flexibility used. Early approaches strictly stuck to the lock and key mechanism of substrate binding, using fixed ligand and protein pocket positions in a procedure known as rigid body docking. Initial versions of the DOCK software exemplified this approach, using purely geometric criteria with hard-sphere repulsions modelling steric clashes and a hydrogen bonding term simple enough to optimise using a least-squares routine[550,551]. This was one of the first examples of software capable to scale for screening purposes, but even the authors acknowledged its clear limitations. They suffered from many of the aforementioned issues with conformer generation, requiring and accurate and biologically active ligand position to be predetermined along with a viable pocket conformation[552,553]. Solving either of these problems is difficult, and hence their usage is limited currently to either looking to replace crystal structures with already bound substrates using pre-optimised ligands or as a preliminary screening step[554]. However for problems where these are already known, they are by far the most computationally efficient due to only requiring the calculation of an optimal 6-dimensional Euclidean transform[555]. This can be analytically solved using the formulas outlined in Section (2.2.1).

It is important to note that in reality, ligand-receptor binding is anything but a static process and the induced fit model is much more realistic[556]. This provides two additional levels of possible flexibility. Flexible ligand approaches use a similar approach to conformation generation schemes by separating a molecule into fragments based upon rotatable bonds[557]. Further approaches additionally introduce flexible receptor atoms. In these methods docking becomes a significantly harder optimization problem, with additional dimensionality caused by each conformational variable of both the receptor binding site and ligand. Existing techniques have attempted to reduce the computational load for example through minimising the allowed receptor degrees of freedom or using simplified FFs for protein atoms[558]. Despite these attempts, most popular screening tasks deem a flexible ligand approach as the most suitable compromise between speed and accuracy, and this serves as the main focus on the rest of this section[559]. These can be supplemented to simulate flexible receptors by performing multiple runs to an ensemble of structures, which can often be more efficient than a single flexible receptor run[560,561].

Further categorisation of the flexible ligand approaches used by different software solutions is then based on analysing how their scoring system is derived and how sampling is performed. A comprehensive overview of sampling techniques is beyond this document, as there are countless ways to optimise for a given scoring function. Many established methodologies such as Monte-Carlo sampling and its many extensions, genetic algorithms, and gradient based methods, all find usage in various tools[562–565]. Instead this section will focus on an overview of scoring functions, as these provide more specific examples to modelling protein-ligand docking and are mostly agnostic to the sampler used. There are three widespread approaches to deriving a scoring function from first principles, using either a force, knowledge, or empirical based approach.

Force based approaches rely on a functional form of the scoring function similar to a MD FF, with many approaches looking to simply augment the existing FFs with small molecule parameters. For example CGenFF aims to provide the CHARMM FFs to cover small drug-like molecules[566]. Others offer a more simplified procedure, such as the more modern DOCK family which directly implements only the non-bonded electrostatic and LJ terms from the AMBER

FF[567]. These methods look to directly evaluate the non-bonded interactions between the protein and ligand, sometimes alongside the internal conformational stress of the ligand. The focus is to identify key stabilizing hydrogen bonds and electrostatic interactions. Methods such as RosettaGenFF provide a full FF approach derived for small molecules, presenting a significant overlap with the general MM FFs discussed in Section (3.2.2) or MD FFs[568]. While this provides an intuitive approach to attaining a score for each complex from the comprehensive potential energy calculation, deriving an effective parameterisation is difficult and accurate prediction of complexes' dynamics is a major research topic. For screening purposes anything approaching a full MD treatment is rarely feasible, and so FF methodologies are found either incredibly simplified, saved for downstream tasks, or regularly as constituent parts of larger solutions.

It is important to note that FF approaches require atomic charges in order to evaluate the electrostatic terms, and in small molecules this requires some knowledge of the electron distribution determining the partial charge of each atom. As always, QM methodologies provide the best accuracy but are unable to scale, and as a result several approaches aim to do this in a computationally efficient manner. The two most popular approaches are the Gasteiger and AM1-BCC methods[569,570]. Gasteiger charges offer perhaps the simplest approach, modelling the charge Q_i of atom i as convergent sum:

$$Q_i = \sum_{\alpha} \left[\left(\sum_{j} \frac{x_j - x_i}{D_i} + \sum_{k} \frac{x_k - x_i}{D_k} \right) \left(\frac{1}{2} \right)^{\alpha - 1} \right] \tag{3.2.3.2}$$

where the indices j and k run over neighbouring atoms that have a higher or lower charge that i respectively.

The terms $D_.$ are reference polarity constants specific to each atom type, and $x_.$ are parabolas parametrised to provide a smooth function interpolating from the charge of i in its anionic, neutral, and cationic states. Due to the damping factor, this sum converges usually in around 6 iterations, making it an incredibly fast approximate method that has shown to corroborate with some experimental data. It was subsequently extended to support more complex binding patterns, although the core form remained the same, and this is currently the most widely used form[571].

AM1-BCC charges instead look toward a fitted approach, aiming to reproduce the results of ab initio QM level calculations carried out across a training set of molecules by correcting the output of a much faster semi-empirical method. It estimates the charge on each atom via an Austin Model 1 (AM1) charge calculation supplemented with a linear Bond Corrective Charge (BCC)[572]:

$$Q_i = q_i^{AM1} + q_i^{BCC} = q_i^{AM1} + \sum_{\alpha}^{\text{bonded}} T_{i\alpha} B_\alpha \tag{3.2.3.3}$$

This secondary term is evaluated over all bonded atoms. The BCC contribution is parametrised by $T_{..}$, is known as the bond connectivity template matrix, and $B_.$, a parameter dependent what type of bond α is. Both of these are fitted parameters, derived from the difference between pure AM1 and ab initio calculations over a test set of 2755 organic molecules.
The AM1-BCC calculation requires more computational resources given its semi-empirical nature and has been shown to provide a better estimate of binding energies. However the same study highlighted Gasteiger charges as producing better results when it comes to pose prediction, therefore there is validity to using both for different contexts[573].

Knowledge based software instead prioritises the information gained from the wide array of experimental complexes available. They take advantage of the availability of data by using measurements of the interatomic distances to create a statistical approximation of the underlying atomic force, hence their alternative name of statistical potentials. Initially developed for early protein-structure prediction tasks, the scope of these methods was later expanded to cover docking forces once a substantial number of docked structures were released[574].
The general concept is to consider a Potential of Mean Force (PMF) $w_{ij}(r)$, calculated from the observed densities of pairs of ligand atoms of type i and protein atoms of type j from experimental bound structures[575]:

$$w_{ij}(\mathbf{r}) = -k_B T \ln\left(\frac{\rho_{ij}(\mathbf{r})}{\rho_{ij}^*}\right) \tag{3.2.3.4}$$

where r is the distance between atoms i and j, ρ_{ij} is the experimental density, and ρ_{ij}^* is a reference density.

Scores for ligands can be obtained by summing this potential over all ligand and local pocket atoms, giving an estimation of the mean force binding the small molecule. Calculation of ρ_{ij} is relatively simple, obtained via measuring the interatomic distances across the used dataset, however calculation of ρ^*_{ij} proves more difficult. Defining how to calculate this reference density is therefore the major theoretical difference between different knowledge based methods, along with the dataset and atom types used.

In the form in Eqn. (3.2.3.4) as function of interatomic separation r, the PMF can be viewed as proportional to the logarithm of the radial distribution function. Sippl's method for parametrising PMFs, first published in 1990, forms the basis of many modern PMF-based approaches although it was first conceived for protein folding[576]. This led to approaches like DrugScore, which evaluate the reference density using the distribution of all atomic distances within a short spherical radius of the binding pocket across all examples. Subsequent advances like ITScore/SE turned to supplementing this with additional terms correcting for solvation and entropic effects[577]. The method named simply *PMF* instead introduces an additional factor accounting for the volume of the ligand[578]. The theory underlying this method additionally finds widespread usage across biomolecular simulation for deriving free energy profiles over MD derived averages of a reaction coordinate. The main issue with this approach is again data availability, as many atomic pairings can be significantly under-represented in a chosen dataset and are at risk to overfitting[579,580]. Despite this it is an incredibly efficient approach once the PMFs have been derived, and has a statistical backing unlike many in silico derived methods. They have also been shown to have good level of accuracy, and have benefitted from the significant growth of the PDB with the latest version of *PMF* using a reference set of 7152 complexes[581,582]

In recent times, the most popular scoring functions used for screening purposes have become empirically derived[583]. This is a direct consequence of the quality and magnitude of datasets vastly increasing over the past few decades, like PDBBind and BindingMOAD collecting binding affinities for over a thousand ligand-protein complexes each[584,585]. Rather than deriving from structural criteria like the density estimation used in PMF methods, empirical methods look to use regression methods to directly estimate the experimental binding affinity as a FF

of various approximate terms[586]. These combine approaches from both previous methods, as physically meaningful interaction terms can be supplemented with heuristic terms relating to more complex phenomena such as solvation, hydrophobic interactions, or PMF derived terms. For screening purposes, these cover a wide range of computational loads depending on the complexity of the algorithm used which can vary considerably. As examples of empirical methods and comprehensive docking strategies, the rest of this section will explicitly consider the methods used by two popular open-source approaches in the AutoDock family, AutoDock4 and AutoDock Vina. Both these packages find significant usage in the community and the rest of this thesis.

AutoDock4

AutoDock4 is a docking suite providing both an empirical FF and a Lamarckian genetic algorithm for pose-optimisation[587]. In its flexible ligand approach, it aims to approximate Eqn. (3.2.3.1) through comparison of the potential energy V of the system across bound and unbound structures[588]:

$$\Delta G = (V_{\text{bound}}^{\text{Ligand}} - V_{\text{unbound}}^{\text{Ligand}}) + (V_{\text{bound}}^{\text{Complex}} - V_{\text{unbound}}^{\text{Complex}} + \Delta S) \tag{3.2.3.5}$$

This uses the potential energy differences between the conformation of just the ligand when it is bound and free, and the complexed system with and without the ligand present. In the unbound state the term $V_{\text{unbound}}^{\text{Complex}}$ can be assumed to zero given sufficient distance between the receptor and ligand. The ΔS term approximates the entropic contributions, which are calculated using a simple model dependent on the number of rotatable bonds N in the system:

$$\Delta S = W_E N \tag{3.2.3.6}$$

The functional form of the potential energy V takes inspiration from non-bonded evaluations in full MD FFs:

$$V = W_{VdW} \sum_{i,j} \left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} \right) + W_{hbond} \sum_{i,j} E(\theta) \left(\frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}} \right) \tag{3.2.3.7}$$

$$+ W_{ES} \sum_{i,j} \frac{q_i q_j}{\epsilon(r_{ij}) r_{ij}} + W_{\text{sol}} \sum_{i,j} (S_i V_j + S_j V_i) \exp\left(\frac{-r_{ij}^2}{2\sigma^2} \right) \tag{3.2.3.8}$$

The Van der Waals force contribution is identical to its $12-6$ widely used form in MD, and the parameters $A_{..}$ and $B_{..}$ are taken directly from an older version of the AMBER FF[589]. Unless alternatives are provided at runtime, atomic charges are calculated using the Gasteiger method. Electrostatic calculation include a $\epsilon(r)$ term designed to emulate screening effects in proteins[590]. The $12-10$ hydrogen bonding term is also present in this version of AMBER, offering a potential well which tends to zero quicker. Here it aims to represent the importance of hydrogen bonds for bonding interactions, and the atom-type dependent constants $C_{..}$ and $D_{..}$ are chosen to replicate the known energy values for the frequently observed $O-H$, $N-H$, and $S-H$ atomic interactions. The angle θ measures the deviation from the ideal hydrogen bond geometry, and the function $E(\theta)$ quantifies how much this changes the energy contribution[591]. A solvation term is calculated using the products of solvation parameters $S_{.}$, determined for each atom type, and the effective volume of each atom $V_{.}$. Its exponential weighting heavily weights nearby atom pairs, aiming to replicate the energy contribution from displacing waters in the pocket[592]. The indices i and j enumerate all heavy atoms and explicit polar hydrogens within a user defined bounding box, excluding ligand-ligand interactions corresponding to 1-2, 1-3, and 1-4 interactions. Non-polar hydrogen atoms are considered during the assignment of atom types, then removed from the ligand for further calculations assumed to be adequately represented by the 33 included types[593]. The empirical part comes from the $W_{.}$ coefficients, weighting the different contributions. In its initial form, these were based upon fitting to 188 experimental complexes annotated with binding affinities, and verified by a supplementary test set of an additional 100 complexes.

For computational speed, the method adopts a grid-based system where a regularly-spaced lattice, with points typically separated by $0.2-1.0$Å, is placed

across the binding site. At each point a probe atom of each time is placed and V is evaluated between this sole atom and the protein. Continuity is retained by calculating trilinear interpolations at grid points when evaluating the energies, and this grid formulation turns many FF evaluations into much more efficient summations. Calculation of the grid values are required before docking can be performed, and are stored on disk in an associated `.gpf` file. Efficient sampling of the position of the ligand is achieved via a Lamarckian genetic algorithm. It uses a $(7 + N)$ dimensional representation of a ligand: a $3D$ translation vector, a $4D$ unit quaternion representing the rotation as discussed in Section (2.2.1), and a single real number angle for each rotatable bond angle. The initial stage samples M conformations, taking each coordinate from a uniform distribution across the bounds of the lattice. This is then put through several steps of the Solis and Wets local optimisation algorithm, a non-gradient based scheme which provides small adjustments to the location[594]. Once a pre-determined number of local search steps have been evaluated the generic algorithm redistributes the M runs as offspring of the previous set, with the amount derived from each proportional to the fitness of that position. The Lamarckian aspect of this comes from each offspring's ability to mutate via probabilistic addition of a random Cauchy-distributed number.

In the advent of recent progress in gradient based GPGPU optimisation methods, the AutoDock4 algorithm was revisited in 2021 as a target for GPU-optimization[595]. An average run of the original method requires 250 million calls to the scoring function Eqn. (3.2.3.5), despite the innate parallelism between the M samples in the described method. The grid based nature of these evaluations also lends itself well to GPU hardware, drawing parallels with the benefits the hardware has brought to grid based PME schemes discussed in Section (1.3.5). This process was locked to a single thread on a single CPU core, significantly limiting the scope of the program for large scale screening. Revising this process led to the development of AutoDockGPU, a GPGPU fork of the program using the same FF and grid-based Lamarckian optimisation scheme. This implementation moved score evaluations and the Solis and Wets local search algorithm to the GPU, achieving speed-ups from factor of 30× to 350×, depending on the GPU and the complex investigated. It additionally implemented the ADADELTA

methodology as a gradient-based local-search algorithm[596]. This took the gradient of the score function with respect to the $(7 + N)$-dimensional vector used to represent the molecules position. While overall performance between the two local search methodologies was comparable, ADADELTA was capable of achieving a 50% success rate using 6× fewer score evaluations. This highlights the robustness of modern gradient based algorithms for efficiently solving difficult optimisation problems.

AutoDock Vina

AutoDock Vina is the most recent addition to the AutoDock family, created with large scale screening purposes in mind[597,598]. Compared to the CPU-bound version of AutoDock4 it is two orders of magnitude faster, making it quicker than even the recent AutoDockGPU implementation. It achieves such a speed increase through innate parallelism, utilising a multithreaded design, and a heavily simplified potential and free energy function.

The predicted binding affinity is calculated as the intermolecular contribution to the potential energy V:

$$\Delta G = \sum_{i \in \text{Lig}} \sum_{j \in \text{Prot}} \frac{V_{ij}}{1 + wN} \tag{3.2.3.9}$$

where as before N is the number of rotatable bonds and w is an empirical weighting for this entropic consideration. For atomic types, Vina uses a simple scheme requiring just the effective radius of the atom R_i and 3 booleans determining whether the considered atom is hydrogen bond donor or acceptor and is hydrophobic. This allows definition of the surface distance d_{ij} between atoms:

$$d_{ij} = r_{ij} - R_i - R_j \tag{3.2.3.10}$$

The basic potential energy V used is not representative of an actual energy like in a full FF or AutoDock4, and instead consists of 3 steric terms and 2 non-bonded

interactions:

$$V = \left[\sum_{i,j} w_{G_1} \exp\left\{ \left(- \frac{r_{ij}^2}{0.5} \right) \right\}^2 + w_{G_2} \exp\left\{ \left(- \frac{r_{ij}^2 - 3}{2} \right) \right\}^2 + w_F \mathrm{F}(d_{ij}) \right] + \sum_{k,l} w_H H(d_{kl}) + \sum_{m,n} w_P P(d_{mn}) \tag{3.2.3.11}$$

where the indices i, j run over all atom pairs except 1,4-interactions, k, l run over all hydrogen bonding capable pairs, and m, n run over hydrophobic atoms. The first three terms in Eqn. (3.2.3.11) are steric, modelling the physical interactions between atoms.

The function F is designed to heavily penalise intersections between atomic radii:

$$F(d_{ij}) = \begin{cases} d_{ij}^2 \text{ if } d_{ij} < 0 \\ 0 \quad \text{otherwise} \end{cases} \tag{3.2.3.12}$$

The H function models hydrogen bonding:

$$H(d_{ij}) = \begin{cases} 1 & \text{if } d_{ij} < -0.7 \\ -\frac{10}{7} d_{ij} & \text{if } -0.7 < d_{ij} < 0 \\ 0 & \text{if } d_{ij} > 0 \end{cases} \tag{3.2.3.13}$$

The middle case simply linearly interpolates between the two outer cases, providing a purely distance based contribution ignoring directionality of the bond. The P function models hydrophobic interactions:

$$P(d_{ij}) = \begin{cases} 1 \text{ if } d_{ij} < 0.5 \\ 1.5 - d_{ij} \text{ if } 0.5 < d_{ij} < 1.5 \\ 0 \text{ if } d_{ij} > 1.5 \end{cases} \tag{3.2.3.14}$$

Again, the middle case linearly interpolates between the others.

The values of the weight terms $w_{\cdot}$ are empirically derived from the PDBBind dataset, and all interactions are cut-off at a distance of 8Å[585]. To optimise the

ligand positions, Vina again utilises a grid based scheme and a $(7+N)$-dimensional representation, although with the simplicity of V this can be calculated easily and stored in RAM. Rather than utilising a genetic algorithm, it opts for a global Monte-Carlo search operated inspired by the ICM method[599]. This is augmented with a local gradient based method based on the BFGS algorithm[600]. While the form of Vina is exceedingly simple, it remarkably provides better results than AutoDock4 for binding pose estimation[601]. This observed strength of empirical methods, able to reproduce experimental data from such a simple form, has led to them becoming some of the most widely used non-ML modern approaches to docking.

3.2.4 MMGBSA/MMPBSA

Docking is a useful process for screening, and modern software has the capabilities to accurately predict and rank poses. However the scores are notorious for being inaccurate and unreliable when compared to pharmacological measurements of binding affinities[602,603]. Alchemical Free Energy (AFE) methods have been developed to provide significantly more accurate estimations of binding strength through performing full atomistic MD calculation[604,605]. Methods such as Free Energy Perturbation (FEP), Thermodynamic Integration, and Nonequilibrium Work look to estimate ΔG through comparing solvated protein-ligand complexes and free ligands[606–609]. These methods all share a similar approach by evaluating the free energy difference by constructing a non-physical (or alchemical) pathway between a ligand bound state and free ligand state in absolute calculations, or alternatively between two different bound ligands for a relative difference. It is conventional to parametrise this pathway using a single parameter λ, from an initial state when $\lambda = 0$ and final state when $\lambda = 1$. The free energy from the two states can subsequently be expressed as an integral over λ:

$$\Delta G_{0\to 1} = \int_0^1 d\lambda \left(\frac{dG}{d\lambda}\right) \tag{3.2.4.1}$$

The error in evaluating this integral is inversely proportional to the overlap in phase space between the two states. In ligand binding scenarios the overlap is

generally non-existent, and thus several different simulation windows must be sampled over the path to ensure accuracy. Each window gradually alters or removes different contributions to the energy, such as electrostatics or VdW interactions. Once the simulations are long enough to provide an accurate energy estimation, these results can be used to extract the full or relative binding energy. The main difference between the different methods available is the estimator used to predict ΔG. Without turning to QM methods these offer possibly the greatest degree of accuracy for estimating binding energies, however this comes at a large computational cost. All AFE methods require several simulations to be run, and their accuracy is directly linked to the number of independent windows used[610]. Therefore while these simulations can be run in parallel and benefit greatly from GPU acceleration, they still require a significant committal of computational resources to reach the lengths required. Even advanced proprietary software like Schrödinger's FEP+ can manage just over 13 compounds per day[611]. In the situations this work considers, where there is little to none SAR information available, this is simply too low a number to identify ligands from first principles using high-throughput methods. Therefore there is a significant need for general methods with a level of accuracy accuracy above docking calculations yet below these alchemical free energy methods.

Molecular Mechanics with Generalized Born and Surface Area solvation (MM/GBSA) or with Poisson-Boltzmann (MM/PBSA) are two widely used methods which strive to fill this gap[612]. First introduced in 1998, these methods aim to substantially reduce the computational load of AFE methods by estimating binding energy while requiring sampling of fewer simulations[613]. Both methods originally decomposed the free energy of a single state into 6 terms:

$$G = U_{\text{bonded}} + U_{\text{EM}} + U_{VdW} + E_{\text{pol}} + E_{\text{np}} - TS \tag{3.2.4.2}$$

The first three terms are identical with the classical FF potential energy terms discussed in Section (1.3), quantifying the contributions from the bonded, electrostatic, and Van der Waals forces. Latter terms supplement this with two solvation terms for the polar E_{pol} and non-polar E_{np} contributions, and an entropic contribution proportional to the temperature and a calculated energy value. It uses a

basic estimate of the binding energy as a 3 term equation:

$$\Delta G = \langle G(\text{complex}) \rangle - \langle G(\text{protein}) \rangle - \langle G(\text{ligand}) \rangle \tag{3.2.4.3}$$

Either 3 simulations can be run, generating an estimate for each term directly from the dynamics of that system, or a single calculation can be run to simulate the complex. Somewhat counter-intuitively, the 3 simulation method has been seen to be less accurate than running a single system, which along with its significant computational savings make it the general choice[614]. Values for all the terms in Eqn. (3.2.4.2) which are identical to the FF used can be extracted directly from the MD if forces are saved, however techniques to estimate the next 3 create some diversity in the approaches taken.

Several strategies have emerged for the calculation of the entropic term, including normal mode analysis or a term dependent on the number of rotatable bonds similar to Eqn. (3.2.3.6)[615]. Many studies simply omit the term, although others highlight it as non-negligible, leading to significant variation between studies and discussion across the literature. The two solvent terms estimate the energy required to displace water molecules, decomposed into a polar term representing the change in electrostatics and a non-polar term modelling the work required to form the cavity.[616] In order to calculate these, the simulation is stripped of all solvent molecules and methods usually used to describe implicit solvents are introduced to estimate the solvents contribution to Eqn. (3.2.4.2). The non-polar solute contribution can be estimated using a simple scheme, whereby it is assumed to be linearly proportional to the Solvent Accessible Surface Area (SASA):

$$E_{\text{np}} = \gamma \times \text{SASA} + b \tag{3.2.4.4}$$

where γ is commonly known as the tension, and is implementation dependent. This is a relatively simple calculation, as the SASA can be computed directly from the Van der Waals radii of the atoms. Several newer schemes augment this with a dispersion term, although comparisons have shown it to make a small difference to the overall accuracy.

Calculation of the polar term leads to the nomenclature difference between the two methods, with both using different continuum approximations of the electrostatic contributions. In the MM/PBSA method, the electrostatics are modelled using the Poisson-Boltzmann differential equation[ii]:

$$\Delta\epsilon(\mathbf{r})\Delta\phi(\mathbf{r}) + 4\pi\rho(\mathbf{r}) = 0 \tag{3.2.4.5}$$

where $\epsilon(\mathbf{r})$ is the dielectric distribution function, $\phi(\mathbf{r})$ is the electrostatic potential, and $\rho(\mathbf{r})$ is the charge density. This can be solved using any finite difference method and is commonly evaluated on a grid cover of the solvent-solute boundary. Solving this equation requires substantial computational resources, and in response to this the Generalised Born method was developed. Used in MM/GBSA, it instead calculates the effective Born radii for each atom, an estimate of the effective radius of an atom given the screening effects caused by its local environment. There are several different schemes to calculate this, with most relying on rudimentary evaluations of simplified versions of the Poisson-Boltzmann equation[618,619]. Once these are calculated, evaluating E_{pol} reduces to a summation over the boundary atoms:

$$E_{\text{pol}} = -\left(\frac{1}{\epsilon_{\text{solute}}} + \frac{1}{\epsilon_{\text{solvent}}}\right)\sum_{i,j}\frac{q_i q_j}{C_{ij}} \tag{3.2.4.6}$$

$$C_{ij} = \sqrt{r_{ij}^2 + \alpha_{ij}^2 \exp\left(\frac{r_{ij}^2}{4\alpha_{ij}^2}\right)} \tag{3.2.4.7}$$

where α_{ij} is the average Born radii between atoms i and j.

Both methods have seen significant adoption by the drug discovery community as a step in screening pipelines, as one of the few free energy calculations requiring samples from a single state. Several extensions have been proposed for both performance and accuracy. By far the most time-consuming part of the method

[ii] Although this form was used in the original description of the method, it does not provide an adequate description of mobile ions and therefore alternate forms are widely used today[617]. The same is true for the form of the Generalised Born method, however in both circumstances this is simply a more general form and so the theory is unchanged.

excluding the initial MD run are the electrostatic calculations. However their grid and sum based nature has led to them being critical targets for GPU acceleration, and several publications have shown great speed-ups in comparison to CPU bound solvers. For Poisson-Boltzmann calculations this has led to an observed $10\times$ increase in one study and another implementation showcasing a $5 - 50\times$ increase across a wide variety of systems[620,621].

Additionally in recent years, these calculations have been run with small QM regions defined around the ligand in order to improve accuracy. The initial introduction of this method showed how a QM-enabled MM/PBSA (QM/MMPBSA) was able to effectively reproduce the results of a similar QM-FEP run, showing promising results[622]. This has been extended to several systems, with the QM/MMGBSA methods commonly showing an increased sensitivity to chosen parameters although scoring highly when supplied significantly more data than the non-QM method[623,624].

3.3 Case Study 4: CACHE2 Drug Discovery Challenge

3.3.1 Overview

This work marked the beginning of my PhDs turn to applying my knowledge towards the formal process of drug discovery. Requiring a significant amount of programming and big data processing, over the course of this project I learnt how to design, develop, and implement a DDP from first principles- suitable for situations when no prior knowledge of any SAR is available. It is the most computationally focused project, and the description below highlights the successes of achieving the required performance criteria for the many high-throughput calculations used. Designed to take small molecules from large VS libraries of SMILES strings up to performing free-energy binding calculations using QM/MMGBSA, the implemented DDP was used to screen billions of molecules under a tight time deadline. It revolved around the CACHE2 drug discovery challenge, a competition to identify binders to SARS-CoV-2 NSP13, the target of earlier investigative work described in Case Study 2. Further information and resources about the structure, experimental procedures, and results of the challenge can be viewed at the website:

- `https://cache-challenge.org/challenges/finding-ligands-targeting-the-conserved-rna-binding-site-of-sars-cov-2-nsp13`

My group learnt of the challenge in late August 2022, narrowly before it began, as a collaborator invited us to participate due to our previous work. The aim was to target compounds specifically to the RNA-site and a small stalk pocket in an attempt to find compounds capable of competitively binding with RNA to halt the replication process of the disease. These would be experimentally tested by the organisers, before later being publicly released with the hope that the community could use these as a basis for LO in future development of treatments against COVID-19. The challenge involved several competing teams with a staggered 2 round process of submitting compounds for experimental binding assays. The approach utilised known fragments identified in experimental

structures, which served as the basis for a large scale similarity search across a VS library. Compounds passing the screening tests were subsequently put forward for verification by the medicinal and pharmaceutical chemists comprising the team. I was responsible for coding and managing the pipeline which interfaced the different software leading molecules from the VS libraries to the point of QM/MMGBSA calculations, which were handled by my colleagues. It required a substantial amount of learning about this new area of study, and as such I view the preceding review of the area as part of this work. As a group effort, there was a significant amount of teamwork and collaboration between the group on how to improve each part of the process. This section aims to provide an insight into my contributions to the overall strategy.

3.3.2 Background

While the previous Case Study 2 focused on characterising the dynamical and functional properties of SARS-CoV-2 NSP13, this work looked to progress this study towards finding small molecules capable of binding to the protein. Subsequently, the following description of the work will assume much of the knowledge provided in the previous work.

Existing antiviral treatments against SARS-CoV-2 have shown great success in targeting the RTC complex. These treatments have achieved notable successes in reducing cases of severe hospitalisation amongst the treatments currently available on the NHS[625]. The RdRp is targeted by the prodrugs molnupiravir and remdesivir, both of which metabolise into NTP-competitive binders in the cell capable of avoiding NSP14's proofreading mechanisms[626]. Metabolised molnupiravir is incorporated by the RdRp into the product ssRNA as a cytidine, however it has two tautomeric forms which alternate between being interpreted by later polymerases as a cytidine or a uridine[627,628]. This introduces significant transcription errors to the product ssRNA strand which compounds through subsequent mutagenesis, leading to a destruction of the viral payload[629]. Remdesevir has a similar method of action via metabolising into an adenosine analogue[630]. While the exact physical mechanism is unclear, theorised to be due to simple steric clashes, introduction of this molecule to a product strand results in termination of the transcription

process after a few more nucleotides have been processed[631]. This has been used as a viable therapeutic treatment in both SARS-CoV-2 and the Ebola virus[632]. The Mpro is targeted by a combination treatment of nirmatrelvir and ritonavir[633]. Ritonavir serves as a CYP3A4 inhibitor, preventing the natural degradation process of nirmatrelvir increasing bioavailability[634]. Nirmatrelvir itself is the active compound, which covalently bonds to the Mpro active site to prevent the binding and slicing pp1a and pp1ab[635].

Despite the relative success of these treatments, all 3 have been subject to significant criticisms. The strategy of molnupiravir to encourage a high level of mutations has been shown to have a statistically significant effect on the prevalence of new strains in areas where it has been introduced and in patients with prolonged infections[631,636]. Remdesevir effectiveness is contentious, with several trials showing it to have no significant outcome improvement when compared to placebo studies, although there are contradictory reports to this.[637–640] Nirmatrelvir and ritonavir have instead been associated with occurrences of "rebounds", where the initial treatment works but the disease later comes back with potentially increased severity[641–643]. Again this is controversial and there is literature stating this mechanism is just as prevalent in control groups highlighting it as a feature of the disease[644]. While only time and further studies will tell the validity of these criticisms, they undoubtedly provide ample support for continuing research to find potentially safer and more effective therapeutic alternatives.

As outlined in Case Study 2, NSP13's critical role in moving the viral payload during the replication process highlights it as an alternative target. Helicase inhibition has already been proven to be a valid anti-viral strategy for targets such as the Herpes Simplex Virus (HSV) and Hepatitis C (HepC)[645]. Their critical function makes them highly sensitive to mutations or interference and critically highly conserved, showing promise for avoiding mutated resistance[646]. In treatments for the HSV, a new class of compounds known as Helicase Primase Inhibitors (HPIs) have shown a significant improvement when compared to existing treatments[647,648]. The helicase primase is a complex including the SF1 UL5 DNA-helicase, and as the 3D structure of this complex has not been resolved a full structural understanding of the mechanism for inhibition is unavailable[649,650]. Despite this research on HPIs continues, with inhibitors like amenamevir already

approved in Japan for shingles (Herpes Zoster), and pritelivir undergoing Phase III clinical trials for generic HSV-1/2 treatments[646,651]. The SF2 RNA-helicase NS3 of the HepC virus has also been identified as a key target for therapeutic intervention, leading to a large array of treatments in various states of availability[652,653]. However the literature is inconclusive as to how much these directly target the helicase domain and instead inhibit NSP3's protease function[654]. Direct HepC helicase inhibition is still actively researched, and several early stage results show some promise at identifying prospective inhibitors[655,656].

Similar treatments for NSP13 have not yet received as much attention, despite it being the most conserved NSP in across coronaviruses. Any treatment targeting this protein should in theory be effective against further strains while bypassing some of the risks involved with Mpro or RdRp targeting. The organisation of the CACHE2 drug discovery challenge looked to organise a concentrated effort to find small molecules capable of binding to NSP13. Aiming to mobilise interdisciplinary teams of computational researchers in order to accelerate the development of inhibitors, it was set up to provide a motivating challenge alongside providing access to experimentalists capable of testing compounds. As the RNA unwinding procedure is mechanistically coupled to ATP hydrolysis in NSP13, both the ATP binding pocket and RNA pockets provide promising targets for inhibition. Previous sequence analysis studies made it is clear the ATP site of NSP13 is significantly less conserved across coronaviruses than the RNA binding pocket, leading to the latter being chosen as the target[657]. Such consistency points towards a high functional importance, and therefore it is hoped that drugs binding to this site would be more effective against future pandemic threats from the Coronavidae family viruses. This posed a difficult task as inhibition of RNA binding by small molecules is a fledgling field with almost no information available for SF1B type helicases[658]. As shown in Fig. (3.1), the project would consist of 2 sequential stages of ligand selection. The first would culminate in a selection of 100 molecules for initial testing, before a further final round of testing for 75 further optimised compounds[iii]. This staggered procedure would provide a time frame to reevaluate the strategy used and rerun the DDP with the experimentally confirmed binders

[iii]This was changed from the 50 shown in the diagram at a later time by the organisers.

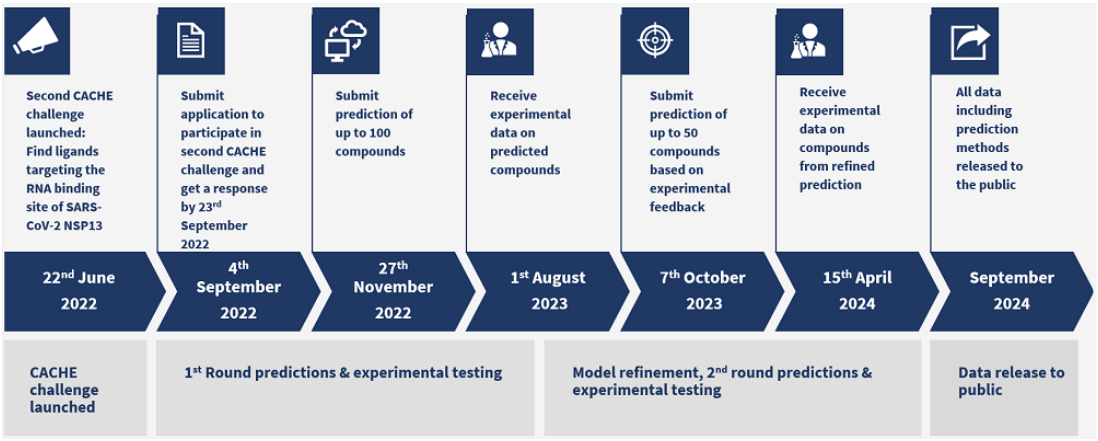


Figure 3.1: Figure showing the outline of the CACHE2 challenge. Further information about this timeline, the results, and methods used can be found on the website linked in Section (3.3.1).

to perform some LO on hits from the first round, as all groups' round 1 molecules would be publicly released.

The experimental procedure was to use both a Surface Plasmon Resonance (SPR) and ATPase assays to measure the binding and ATP processing activity respectively. SPR-based experiments measure how much the tested ligand is able binds to a target protein by providing an estimate of it's binding constant K_D[659]. Immobilizing several copies of a ligand on a dextran surface coupled to a thin metal sheet and applying constant light source to it will trigger a surface plasmon polariton to travel parallel to the sheet. The magnitude of this wave is very sensitive to the refractive index of the material, which in turn is sensitive to whether the protein is bound[660]. Measuring this value as a dense solution of NSP13 is covered across the surface will result in a curve as the plasmon flow increases when binding takes place, before decreasing as complexes disassociate[661]. The gradient of this downward and upward curve, allow for the equilibrium dissociation constant k_d and association constant k_a to be calculated. The binding constant K_D is defined as the ratio of these values. As the brief focused on RNA-binding, ATPase assays are then used to determine whether the binding occurs in the ATP-binding domain. Each occurrence of ATP hydrolysis liberates a phosphate group[662]. Therefore by comparing the concentration of free groups after a period of time between an ATP-NSP13 mixture to a an ATP-NSP13-ligand mixture, it is possible to assess the impact the ligand has on the frequency of ATP

hydrolysis. Molecules binding to the ATP site should inhibit hydrolysis, giving a lower concentration of free phosphates compared to a control group. Therefore in this scenario, it was preferred to find ligands showing significant results on the SPR assay which were relatively ineffective on the ATP hydrolysis results. Such a result was taken to indicate a ligand which favoured the RNA-binding site, although it would not indicate any clear inhibition of translocation or therapeutic value.

3.3.3 Methodology

At the time of the start of the contest, few papers had investigated RNA-binding small molecules for NSP13. The best source of information was a recent fragment screening study by Newman et al, which crystallised 65 small fragments bound to various pockets in the hope of kickstarting a structure based drug discovery campaign[663]. Many of these identified binders were located near the ATP site, however 5 showed clear potential for the CACHE2 challenge. 4 of these binders were found near the RNA cleft, from PDB IDs 5RLH, 5RLZ, 5RML, and 5RMM. An additional binder from model 5RLB was found near the stalk pocket, identified in the previous work[401]. As the best example of experimental data on RNA binding molecules, these served as the starting point for the constructed DDP. The fragments shown above were used as a starting point for a large scale similarity search. This included the fragment from 5RLB, as binding to the stalk pocket would also provide viable experimental results given it should not affect the ATPase assay. As compounds were to be ordered from the pharmaceutical company Enamine, the EnamineREAL VS database was selected as the source of molecules. This database consists of 6.75×10^9 compounds provided in a `.CXSMILES` string format containing associated metadata about each molecule, with all included molecules satisfying Lipinski's Rule of 5 and the Veber criteria[664].

At this stage with no SAR or in-depth knowledge of important chemical properties, the simplest assessment of general molecular similarity would be found by using a circular fingerprint capable of providing a comprehensive assessment of the total similarity between two molecules. Identified crystallographic fragments were fairly small, ranging from 13-17 atoms in size, so ECFP4 fingerprint vectors

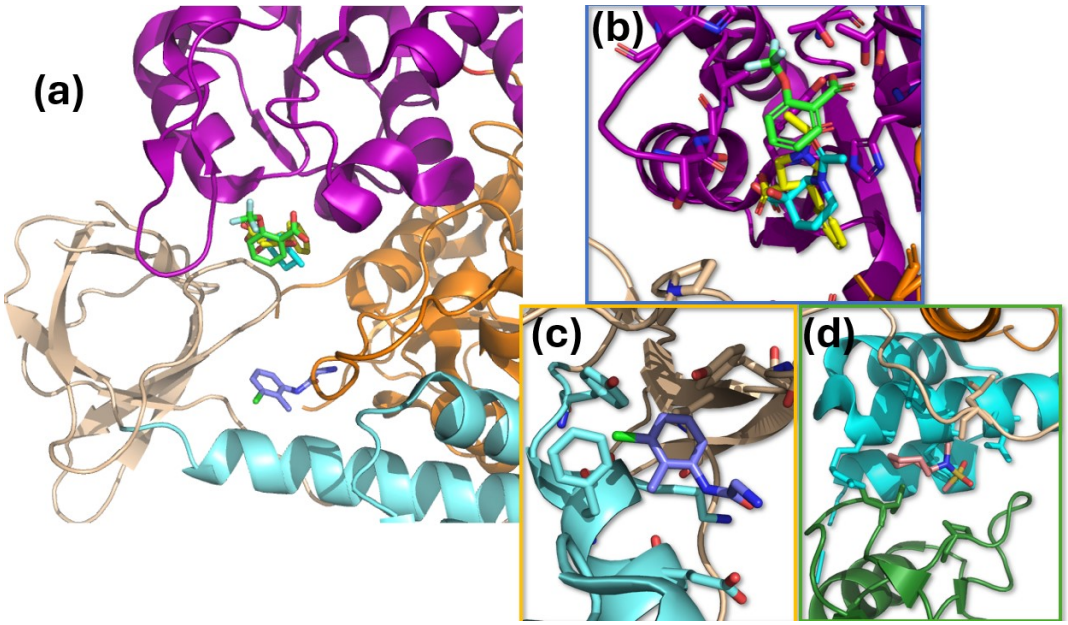


Figure 3.2: Fragments use the following colour scheme 5RLH: Green, 5RLZ: Cyan, 5RMM: Yellow, 5RML: Purple.
***(a)** Shows fragments bound to the entrance of the RNA site between RecA1 and RecA2, close-ups with contacts are shown for the **(b)** fragments bound closer to RecA2, **(c)** fragments bound closer to RecA1, and **(d)** stalk pocket bound structure.*

were chosen to provide a suitable neighbourhood matching criteria. This fingerprint has been shown to be one of best for small molecule VS purposes[665].

Parsing such a large number of molecules posed a challenge, and the size of the data meant moving it to a cluster was infeasible. Generating fingerprints would require uncompressed formats, and this meant processing around 4TB of data. Running searches locally was therefore more viable, as additionally the cluster compute time could be better utilised for more resource intensive downstream tasks. To ensure searching was computationally feasible on the local NVIDIA RTX A4000 with 16GB VRAM, a bit-vector size of 1024 was chosen. This is also known to not introduce a significant number of bit collisions[666]. This allowed 10^8 ligands to fit into the GPU at once, along with some leeway to accommodate for variable molecular size and computational overheads. Meaning a search of the EnamineREAL would require 68 individual searches for each binder.

The Tanimoto similarity was used for maximum efficiency. Before this could be calculated, the initial major computational hurdle was generating fingerprints for all molecules in the EnamineREAL. Performing this efficiently required a

staggered approach, as all molecules could not be fit into the RAM on any available machine. For consistency with the GPU search limitations, each file was split to contain 10^8 molecules. To ensure all SMILES strings used an identical notation they were first reprocessed by RDKits canonicalisation function. The RDKit Morgan fingerprint implementation was used, which is CPU-bound as no suitable open-source alternative was found which was quicker or GPGPU-enabled[667].

This implementation is fairly fast, although this is ultimately dependent on the size of the molecule, taking on average around 1ms per molecule. Extrapolating this to the 6.75 billion calls required a simple sequential strategy for this taking just over 11 weeks without accounting for reading and writing of files or varied molecular complexity. Given the tight time schedule, a dedicated computer was put to use for over two weeks processing 8 separate files at a time in parallel using the Python multiprocessing library. Each fingerprint-SMILES pair was then written to a binary format, allowing for interfacing with the GPU-Similarity software. Performing this one file at a time meant it was possible to begin searching before all SMILES strings had finished processing, enabling downstream tasks to begin much sooner.

The next step was to perform the similarity search. The code for this was written in C++ as a fork of the open-source version of the Schrödinger GPU similarity program, utilising most of its key functionalities[668]. This codebase provided the critical interface between the binary fingerprint files and the GPU, allowing each fingerprint database to be loaded onto the GPU and easily searched. While the original program provided an interactive web-based interface to search the molecules, this was converted to instead iterate over all 68 fingerprint databases and write the results to file. Loading each file sequentially to the GPU was the major bottleneck, as once any file was loaded the search was near instant for 10^8 molecules.

A search for all 5 fragments took an average of around $74s$ per file, giving a near immediate results to begin downstream tasks with once any fingerprint files finished processing. Later once all files were generated, a full run searching all 6.75×10^9 molecules could be done in just under 75 minutes. The 1000 most similar compounds from each file were taken to form docking sets. Molecules were

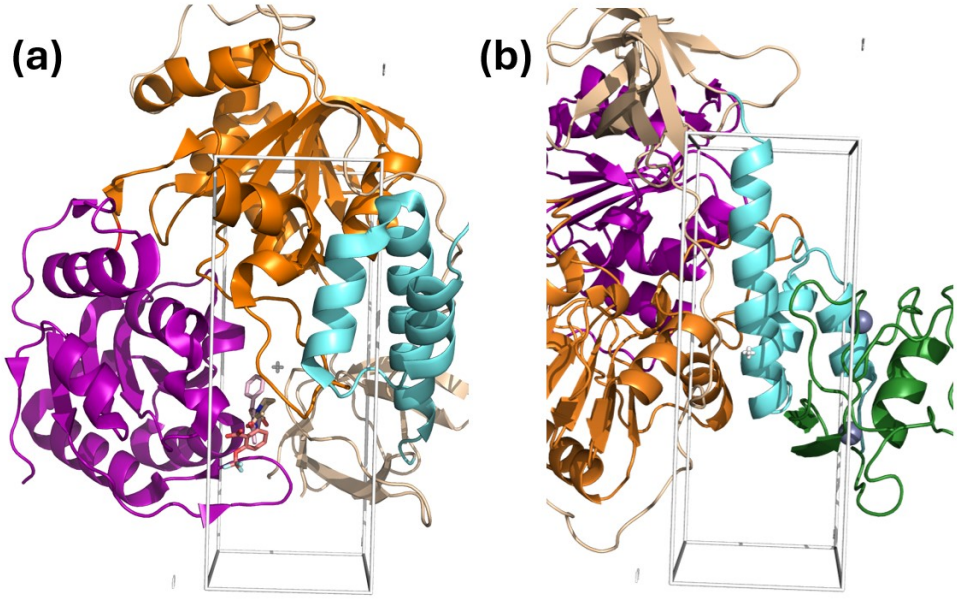


Figure 3.3: The docking structures and boxes used for ***(a)*** *The RNA cleft and* ***(b)*** *the Stalk pocket.*

additionally checked for promiscuity at this stage, using both the PAINS and BADAPPLE filtering SMARTS string algorithms[479,480]

Compounds passing the similarity screening were parametrised and 3D structures were generated prior to docking. Parameterisation was performed using the Schrödinger LigPrep/ConfGen pipeline[528]. This used Epik for tautomer generation with a pH of 7.4 ± 0.5. To prevent the number of conformers generated from becoming unmanageable, each molecule was limited to producing at most 8 tautomers and 8 subsequent stereoisomers. The `OPLS_2005` FF was used for the final FF optimisation step[521]. Molecules were docked to two structures, both prepared by my colleague to represent the two non-ATP pockets considered. The binding pockets for both were chosen to encompass the whole of the binding site, and are shown in Fig. (3.3). Experimental structures were used as the basis for these models to avoid simulated artefacts. For the RNA pocket, the structure was based on the 6ZSL model crystallised in the same study as the fragments, which showed an Apo structure with a wide RNA cleft and no ATP-analogues bound. The stalk pocket structure was based on the 5RLB structure, as this was the only experimental model available that had anything bound to this pocket.

A consensus docking strategy was used where molecules were ranked based on the scores from both AutoDock Vina and the AutoDockGPU implementation of AutoDock4[595,597]. The docking process was the most computationally difficult part of the process, as the batch processing systems for each software used were unsuitable for the speed and scales required. A Python based multiprocessing wrapper was written around them capable of docking both in a highly parallel fashion. Using both CPU and GPU enabled software allowed the tasks to be distributed across different systems able to take advantage of the different requirements. As Vina was CPU-bound, this was run on ARCHER2 system which featured 128-core nodes with 256GB of RAM. This was capable of achieving a superb throughput of ligands, allowing for 1 million complexes to be formed in around 4 hours ($\approx 2s$ per ligand) running separate threads on each core. Due to this capability, Vina was used to initially dock all structures and its ranking was used to order the ligands put through AutoDockGPU, prioritising the better predicted ligands.

Even on the GPU, the AutoDockGPU implementation was significantly slower than Vina, requiring around $7s$ per ligand. In addition to support this required the calculation of `.GPF` files which contained the pre-calculated grid parameters. Fortunately due to the Vina calculations happening on clusters, grid files were able to be generated asynchronously without detracting substantial resources from docking calculations. The generation of these files is CPU bound but fast, and could be optimised by pre-generating all grids for every atom type and then forming the necessary file at run-time. Each instance of AutoDockGPU required 400MB of VRAM, meaning 60 could fit into a 24GB RAM card and run in parallel providing a substantial speed-up but still substantially lower than Vina. The ADADELTA optimiser was used in AutoDockGPU due to its better rate of convergence, and a grid spacing of 0.25Å was used. As direct comparison between docking scores is mostly ineffective, instead molecules were assigned a rank depending on how each software scored them relative to all docked molecules. The average rank of these was used as a final way to select compounds put forward for QM/MM-GBSA calculations.

Before this however, in the first round iterations an additional clustering step was included to ensure there was sufficient variety in the molecules. This was to

counteract the relative lack of diversity amongst molecules found using a pure Tanimoto-based measure, as it significantly biased large numbers of molecules that are solely similar to the fragments. This could skip over potentially more informative results capable of providing a different binding scaffold. While running enough iterations of this process should slowly converge to molecules with features scoring highly on the computational tests, this may happen too slowly to observe over the time given for the challenge. To help aid this and give a more diverse range of molecules for the binding calculations, a clustering method was included to promote compounds which scored highly while also showing distinct features from other hits. For this, from the top ranked molecules had their fingerprint distances calculated using a Euclidean metric. This distance matrix was then clustered using the FasterPAM implementation of K-Medoids into 200 sets[669,670]. For more variety, the highest scoring ligand from each cluster was included, alongside the next top molecules based solely off rank.

As Vina has been shown to produce results more consistent with experimental results, these coordinates would be used for QM/MMGBSA[601]. Due to the software removing hydrogens, a atom-matching graph isomorphism strategy developed by my colleague was used to correct the hydrogen locations before running MD. The MMGBSA was performed using the GROMACS implementation, utilising a QM/MM region covering the associated binding site[671]. A comparison of the QM regions used for each model is shown in Fig. (3.4). Ligands were parametrised using the AM1-BCC method implemented in Antechamber to attain more accurate partial charges[672]. A single-run QM/MMGBSA strategy was used, where MD was performed using the pocket which ranked the highest. All simulations used PME for long-range electrostatics, Langevin dynamics for temperature coupling set to 300K, and the Parinello-Rahman barostat. For each complex, first 10,000 steps of energy minimisation were performed, followed by 50,000 steps of NPT and 50 steps of NVT equilibration. Production MD was performed in the NPT ensemble for 1ns, which was later sampled every 50ps into 20 separate frames used for the binding energy calculations. Entropic contributions were calculated using the interaction entropy method, and surface area calculated using the LCPO formula. The QM region used the PM6-DH+ Hamiltonian. All steps and set-up for this

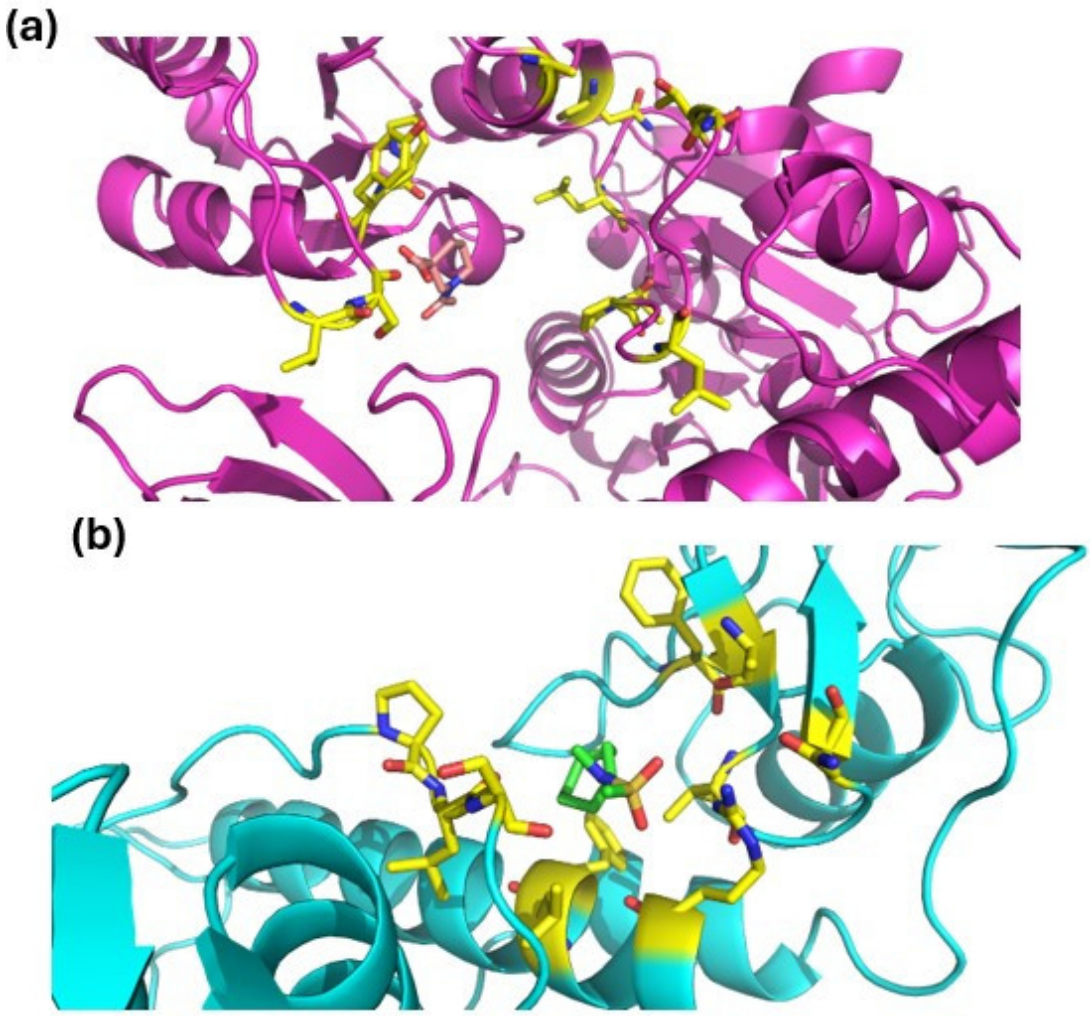


Figure 3.4: Comparison between QM regions for QM/MMGBSA calculations used for the different pockets. Residues in yellow are featured in the QM Hamiltonian. ***(a)*** *Shows the RNA pocket model with fragment 5RLZ shown for clarity, and* ***(b)*** *shows the Stalk pocket with 5RLB bound.*

part were performed by my colleagues, while I provided the selected molecules and then parsed and analysed the final results.

Molecules with the most preferable estimated binding energies would then be collected into PyMol for visual inspection by medicinal chemists on the team, looking for signs of a realistic binding. Alongside this, our collaborators outside the group would run 2 other pipelines using the proprietary Schrödinger software and and Pharmacophore approach. Schrödinger involved its own GLIDE docking software and Prime MMGBSA-based calculation[673]. The best molecules from these approaches would also be put through our pipeline to generate a consistent dataset to inform the final decisions. Molecules experimentally verified through

the first set of results, were also used as the basis for an additional runs through the pipeline after experimental results were received.

3.3.4 Results & Discussion

Preliminary calculations shown in Fig. (3.5) show the Tanimoto similarity between the experimental binders, with a maximum of 0.27 observed between 5RLZ and 5RMM. This gave promise to the strategy, as each molecule was capable of finding a relatively distinct set of molecules. The known binding fragments were additionally converted to SMILES strings and then put through structure generation and docked to each pocket using both software to provide a baseline for comparing docking scores. The values corresponding to the best scoring models are shown in Fig. (3.6). Fortunately all passed through the process and featured a realistic score, although neither software was able to correctly identify 5RLB as the most stable in the Stalk pocket. Instead 5RMM was consistently found as the best scoring ligand by all software across all pockets. This was unsurprising as it is the largest of the fragments, therefore of a more similar size to the compounds used for optimizing these algorithms.

The first rounds similarity search across all files provided a group of 340,000 molecules known as Similarity Search 1 (SS1), consisting of the top 1000 most similar and unique ligands in each file to the 5 fragments. Once this had been put through conformer and tautomer generation, it resulted in 1,102,621 individual molecules for docking. This was docked with both docking programs and the results for the top 25% are shown in Fig. (3.7). The docking scores on average showed good agreement with each other having a Pearson correlation of 0.514, with Vina taking slightly lower average values. Both methods had a substantial number of ligands scoring far lower than the known binders, highlighting the fact that many good experimentally performing ligands may not be clear from either strategy.

In the time frame available as many as possible of these were run through QM/MMGBSA, resulting in binding estimates for 869 compounds. However the results at this stage were difficult to interpret. A distribution of the scores attained for the first round are shown in Fig. (3.8), showcasing the large variation

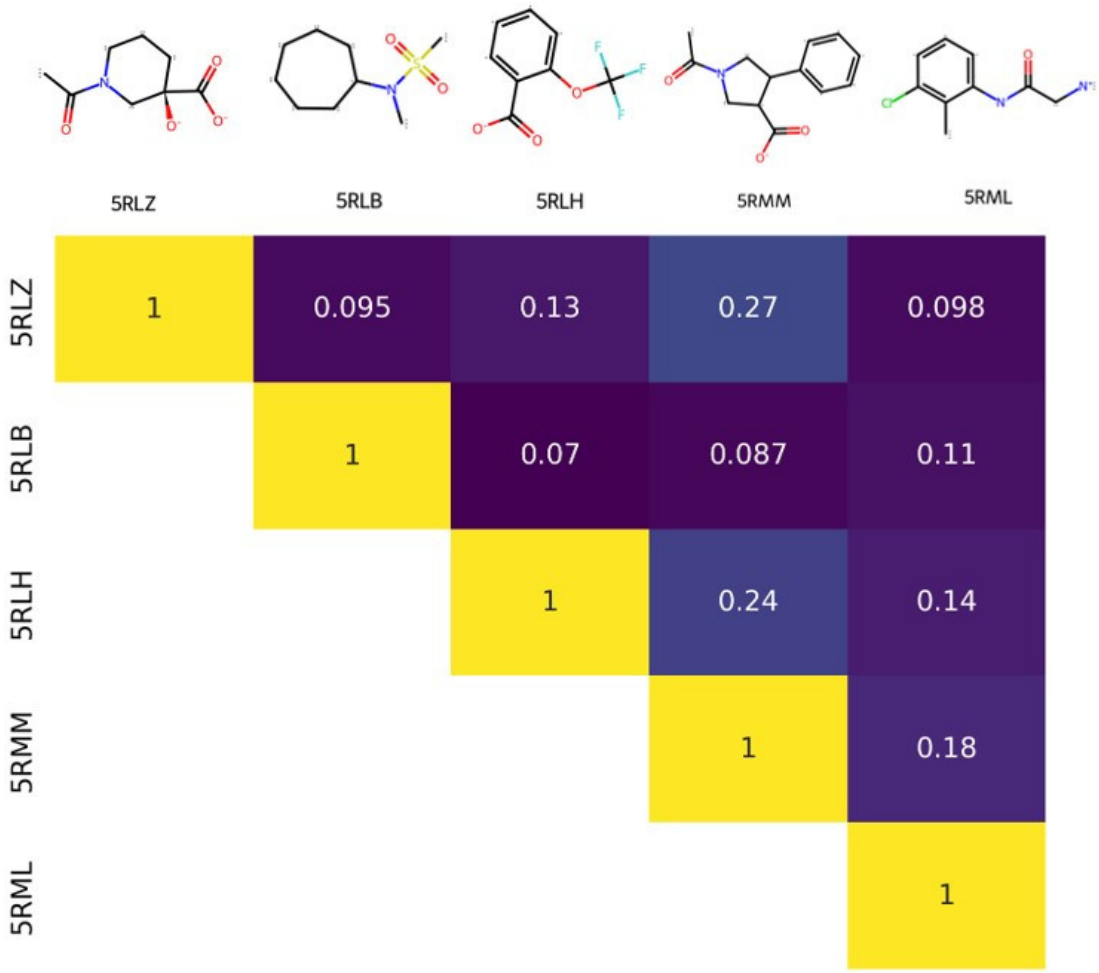


Figure 3.5: The top row shows molecular drawing for each molecule, generated with RDKit. Below is an upper triangular similarity matrix annotated with the inter-molecular Tanimoto similarity between the molecules using a ECFP4 1024-bit fingerprint vector. As crystal structures are unable to pick up hydrogen atoms at the experimental resolution, the protonation states are most likely wrong.

in values attained. Stretching from -36.98 kcal/mol to a huge maximum of 117.22 kcal/mol, including 226 positive binding coefficients indicating unfavourable binding. The average value was 1.65 kcal/mol, with a standard deviation of 23.25 kcal/mol. Considering a value of around -5 kcal/mol to indicate some notion of preferential binding, gives 387 results suitable for inspection and a hit-rate from docking as just 40%. The cause of this would prove difficult to pinpoint, as such variance is not necessarily a bad thing or indicative of an error. Differentiating an extremely poorly binding ligand from an erroneous result requires both detailed numerical and visual inspection, and even then it is not always clear. Due to the simplifications used, docking software is known to yield a basic estimate and

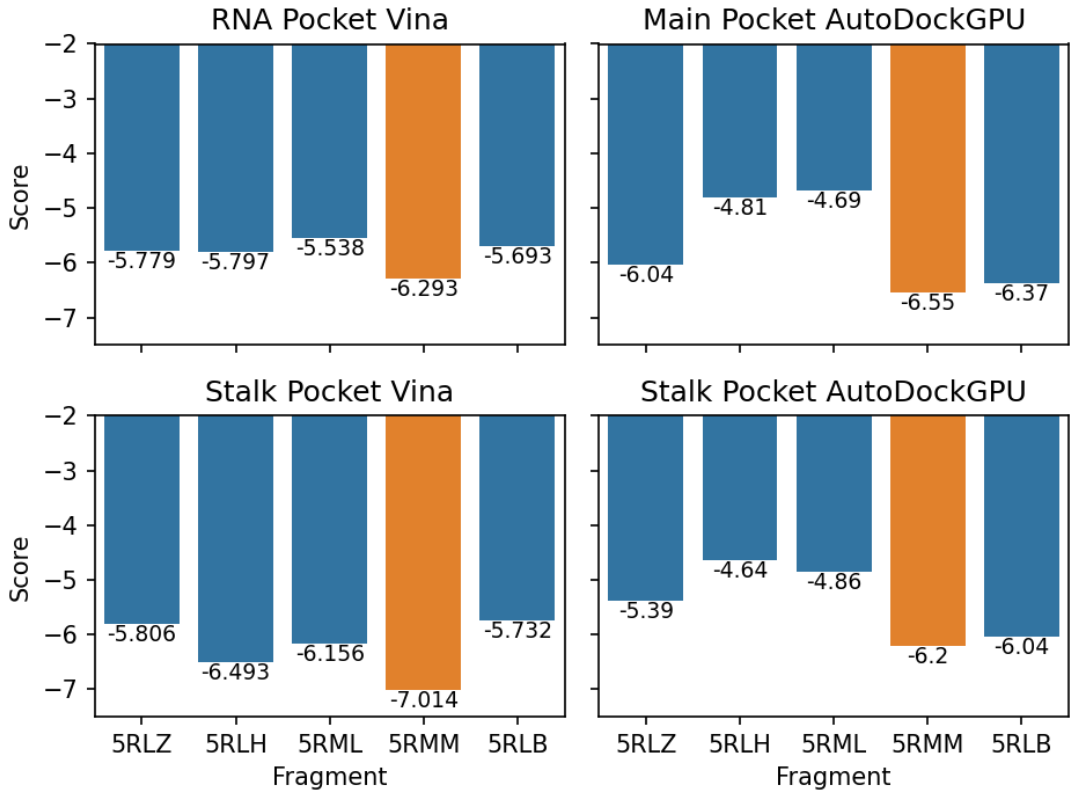


Figure 3.6: Comparison of docking scores attained for each known fragment with AutoDock Vina and AutoDockGPU for each binding pocket used. All molecules were put through the conformer generation pipeline leading to multiple structures, and the best scoring models score is displayed 5RMM is consistently ranked the top and is coloured in yellow.

therefore expected to contain some poor predictions- however 60% of a portion of the lowest results actually being screened does point towards some greater error with the QM/MMGBSA methodology, which shall be investigated later. At this stage, it was unclear whether this was simply working as intended and the docking software put forward mostly poor to egregious ligands.

From inspection of the QM/MMGBSA results and the other members methodologies, 100 molecules were chosen by the team to be experimentally tested. The experimentalists had some issues with the ATPase activity assays, leaving many molecules without results for these. Despite this 93 molecules produced experimental results shown in the appendix in Fig. (3.6), along with chemical drawings of each molecule in Fig. (3.13). One molecule, `CACHE_1451_96` which originated in our method, showed good experimental results on both SPR assays and was automatically put forward for further testing in round 2. The docking positions

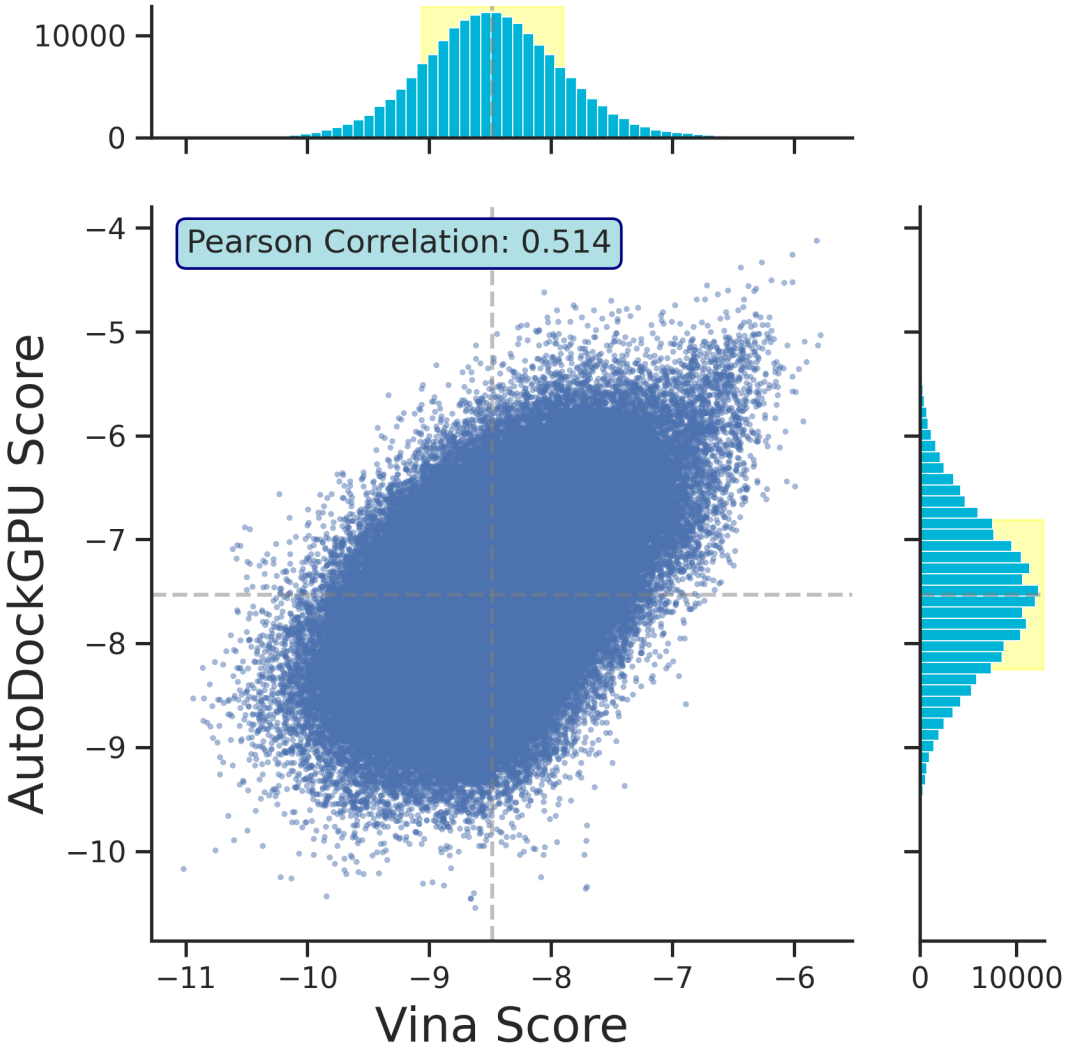


Figure 3.7: Comparison between the docking scores for the top 25% of molecules in SS1. Marginal distributions for each variable are shown around the edges, with their means marked by gray dotted lines and the region one standard deviation away coloured in yellow.

for this molecule from Vina, used for QM/MMGBSA, are shown in Fig. (3.9). An additional summary of the numerical results obtained for it are shown in Table 3.5. Although the experiments are inconclusive as to which pocket this molecule bound to, it was consistently ranked higher on the stalk pocket giving an inclination it may be binding away from the RNA site. From a visual inspection, it is clear the docking position in the RNA pocket is less favourable, featuring a large number of negative electrostatic interactions with the binding pocket while the stalk pocket position had more good hydrogen bonds. The QM/MMGBSA

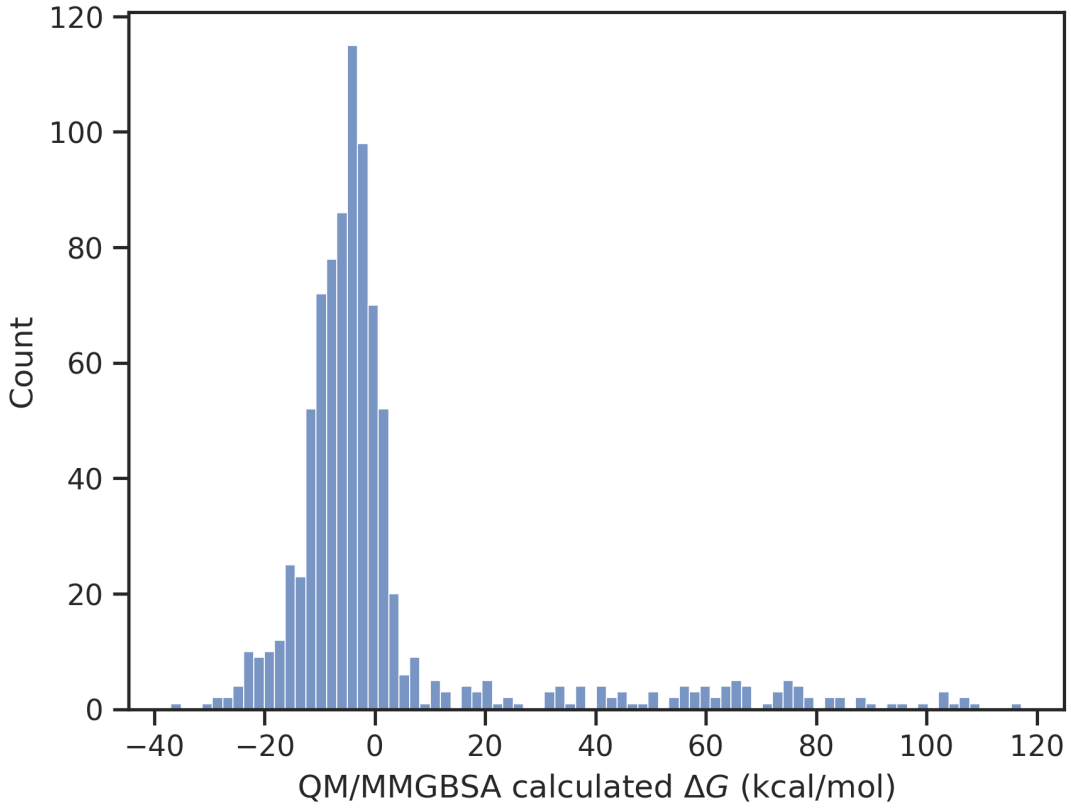


Figure 3.8: The distribution of the binding energies calculated using QM/MMGBSA for 869 ligands screened from SS1 during the first round.

result was significantly better on this site, and while it was not one of the lowest observed, the result for either pocket would have made it to the hand-checking phase. Overall round 1 seemed to provide confirmation that there is viability to this methodology, although it did already indicate some of the critical issues faced. It is also important to note the experiments were performed at a very high concentration ($50\mu M$), which could lead to saturation of the immobilized ligands or even non-specific binding to occur. This means the observed results may be simply due to the experimental conditions, and further thorough testing is required before this can be confirmed as a definite result.

Before commencing round 2, as the other groups molecules were publicly released this was used to verify our strategy. All other groups molecules were docked using both software, and QM/MMGBSA calculation were performed using both docked positions. Results are shown in Fig. (3.10). The docking results reiterated much of what had already been seen. There was no discernible

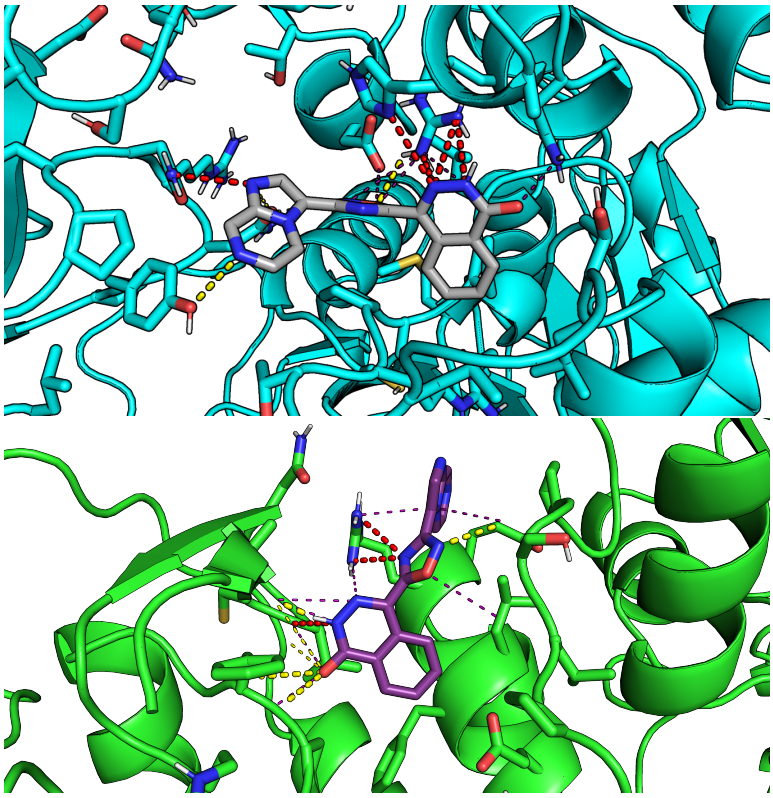


Figure 3.9: RNA-pocket docked (above) and stalk pocket docked (below) structures of the best experimentally performing molecule `CACHE_1451_96` *from round 1. Identified contacts are highlighted with dotted lines. Yellow indicates good hydrogen bonds, red indicates electrostatic clashes, and purple are suboptimal contacts.*

difference between the software in terms of identifying the good binders, although interestingly the correlation between docking scores for these molecules was slightly higher at 0.6. This gave some justification for the consensus strategy being the best approach.

Scores were sufficiently low, with the highest confirmed good binder being $\approx -6.9/-5.4$ (Vina/AutoDockGPU)- somewhat mimicking the results of the fragment tests. Unfortunately there was additionally a large buffer of ligands at the bottom which did not show any binding preference at all but occupied very low-ranking positions. It seems this is unavoidable, and most of the confirmed binders lay quite far above the bottom. For a scale-based strategy such as ours to work this must simply be compensated for by calculating as many QM/MMGBSA results as possible. The QM/MMGBSA comparison showed the Vina positions did provide an on average a slightly lower estimation of the binding energy for confirmed binders, confirming the basis for our usage of those positions. However

Docking Scores	RNA	Vina	-8.252
		AutoDockGPU	-6.77
	Stalk	Vina	-9.004
		AutoDockGPU	-8.35
QM/MMGBSA Results	RNA		-11.9 kcal/mol
	Stalk		-14.57 kcal/mol

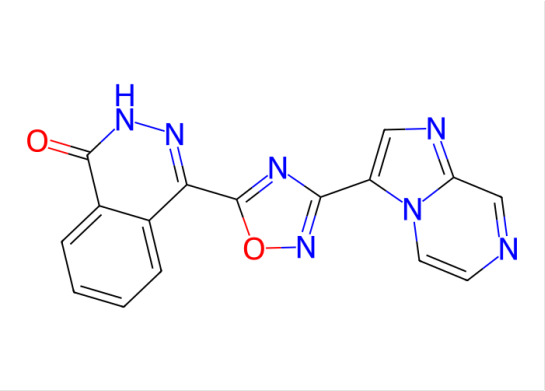


Table 3.5: Summary of the computational results obtained for `CACHE_1451_96`*, alongside a chemical diagram of the molecule. Experimental results can be found in the appendix Fig. (3.6).*

both averages were near to 0. Taking a relevant binding to be a value below -5 kcal/mol, this again results in a 40% identification rate, which is quite low. As even confirmed binders provided a large number of positive values, this continually implied that there may be some issue with either the method for this system or the used implementation.

Moving to round 2, the aim was to perform more of an LO approach to the known binder. For this, all databases were searched again to provide every molecule from the EnamineREAL with a similarity to `CACHE_1451_96`. In addition, this search was supplemented with other molecules showing a reasonable binding result but not carried through to round 2, decided upon by my collaborators. This included `CACHE_1451_97` and `CACHE_1451_85` in the previously tested molecules. Altogether this created a supplementary dataset of 107,861 additional ligands known as Similarity Search 2 (SS2), and its relatively small size is due to the majority of similar molecules already being put through the pipeline. Putting these through LigPrep/ConfGen resulted in 430,765 ligands which were again docked, ranked, and queued for QM/MMGBSA.

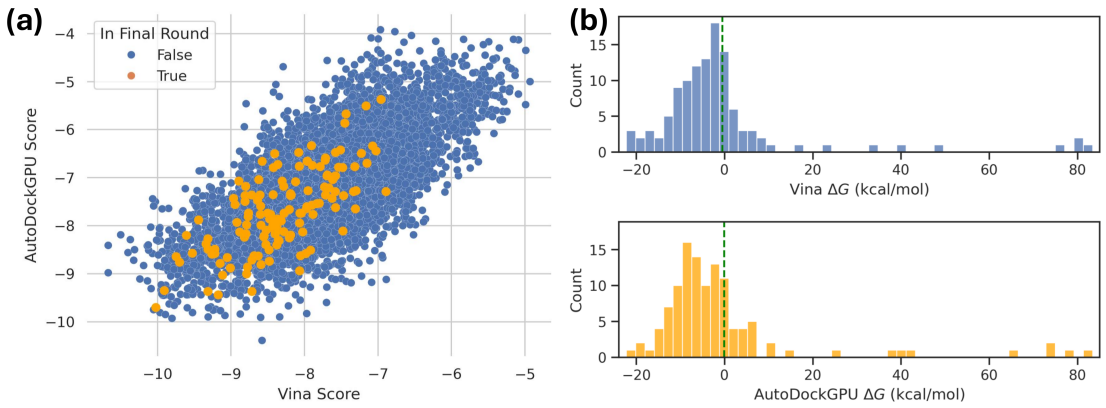


Figure 3.10: Comparison of the docking scores and QM/MMGBSA predictions attained for all groups round 1 molecules. ***(a)*** *Shows a scatter plot of the docking scores for all submitted molecules, taking the lowest score from either lowest binding pocket. Those in orange made it through to round 2 while the blue compounds did not show good enough binding. For those in round 2,* ***(b)*** *shows the distribution of QM/MMGBSA calculated ΔG values using either the Vina docked position (above) or AutoDockGPU coordinates (below). The dotted green line indicates the mean of each distribution.*

The docking distribution for this set along with the correlations between various molecular properties for the top 25% are shown in appendix Fig. (3.12). For the scores, there is no discernible difference when operating at these scales. Both docking scores showed a relatively large negative correlation with the molecular weight, showing a preference for smaller molecules in this process. The similarity to any of the experimentally confirmed binders showed a difference in correlation between the two programs. For Vina it was relatively negligible (0.17), but for AutoDockGPU it was significantly higher (0.4). Neither of these were particularly high however, highlighting the LO strategies approach to deviate from this compound. Interestingly the distribution for similarity shows the dockings preference to score molecules distant from the known binders much higher.

This set was continually updated with molecules from the other pipelines, and similarity searches to good results confirmed by visual inspection. Subsequent additions however greatly decreased in magnitude, as the set began to comprise of the majority of most similar ligands in EnamineREAL. For this round 2319 new QM/MMGBSA compounds were performed on SS2 molecules, and the distribution across these is shown in Fig. (3.11). Again the method provided a wide variation of binding energies, with results this time spanning from a maximum of 195.2 kcal/mol to a minimum of -146.7 kcal/mol. The average for this round was slightly

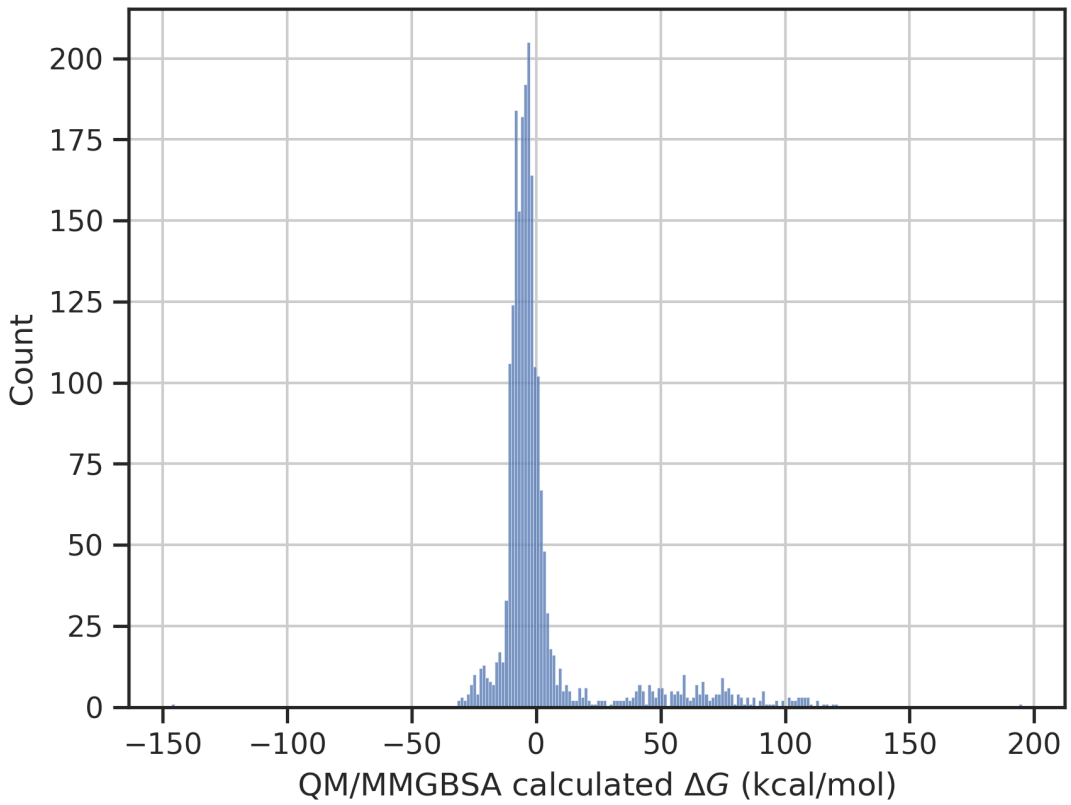


Figure 3.11: QM/MMGBSA distribution for SS2 molecules.

higher, sitting a 2.65 kcal/mol with a higher standard deviation of 24.37 kcal/mol. With results still clustered around 0, the strategy was once more to take the lowest scoring ligands and give them to the experts for inspection. There were 926 new molecules with a binding energy below -5 kcal/mol, giving a pool of 1213 ligands to select from.

The 75 chosen molecules for LO are shown in Fig. (3.14). Of these, only 28 managed to produce any experimental results, and none of these showed any sign of binding. While this was surprising given the lead optimization strategy from a known binder, it is an unfortunate reality of the drug design process. Any alterations made to the original compound had counteracted its original affinity. The numerically-focused approach to LO was too coarse, and clearly achieving any improvement requires more rational changes appropriate to the ligands key characteristics. By simply using numerical results to evaluate whether a stochastically selected change was beneficial, the augmentations considered were

already biased towards what scored well on calculations which clearly had some major errors.

3.3.5 Conclusion & Future work

For a first approach of a group to a drug discovery challenge, I think getting any hit is a positive result and the results show there is some utility in the methodology used. This project was a learning experience and a functioning pipeline had to be constructed under a tight time deadline. The one conceived is good at screening molecules from huge databases down to manageable amounts, and finding an experimentally confirmed binder from a database of over 6 billion is a great result. However while I am proud of the strategy presented and my contribution to it, I believe there are several key shortcomings of the approach.

Firstly, I think a consensus docking method requires more than 2 different programs, and especially ones derived using different strategies. Augmenting the docking pipeline with a fully FF or knowledge-based approach would have benefitted it greatly, providing a different way of confirming any binding potential. Both programs used were empirical and used similar databases (such as PDBBind) in their construction and verification, leading to too high a correlation between the results to really highlight any potentially different compounds. This can be seen in the docking results shown in Fig. (3.7), and the subsequent SS2 search showing the exact same correlation further highlights this. I believe this played a large part in why we were unable to find effective LO compounds, as both Vina and AutoDock4 would suggest similar ligands which actually moved further away from what made the original molecule bind so well. It additionally meant many compounds were disregarded due to mid-level scores, which could actually have been very good binders such as those seen in Fig. (3.10)(a) and the fragments. Potentially there is is room for a more bespoke selection procedure than simply taking the lowest, as those nearest to the fragment scores seemed much more promising for the experimental results. However with the correlation between the chosen software so high, such ligands were easily overlooked using this strategy.

The next issue is with the QM/MMGBSA calculations. This was a major source of problems due to its scale, accuracy, and frequency of errors. I think

using it as a step straight after docking was a mistake, as it is quite a taxing method which could have been saved for the output of an additional screening process. Introducing another process would also be much more effective if it could disregard even a few of the low docking scoring ligands which would go on to perform very poorly at QM/MMGBSA, such as those seen in Fig. (3.10)(a). These calculations dominated the time spent overall, and as previously stated a $\approx 40\%$ hit-rate at this stage is too low to justify it as it means 60% of the available computation time is wasted. This could be augmented with a non-QM MMGBSA method, all it would have to do is run substantially more ligands than we were able to in order to save some time. In my opinion, introducing the QM aspect not only made it much slower, but also introduced additional errors. The error rate observed in Fig. (3.10)(b) in retrospect makes the method unsuitable for a final stage process as it cannot effectively identify an overwhelming majority of confirmed results. I also believe that we did not give them sufficient simulation time, as using the QM implementation has been shown to require more frames for an effective result[622]. Subsequently reanalysing the output files shows clearly that many times it is the entropy contributions that yield massively unrealistic values. This is a known problem with using the interaction energy method to calculate entropic terms, and using such a method in a situation where generality is so vital was a mistake[674].

Finally, I think our approach to LO was fundamentally flawed, and the methodology we used is not suitable for such a delicate task. In retrospect already from SS1 there were enough docking structures to work with, and introducing hundreds of thousands more in a non-structured way during LO meant the rate of data generation was far greater than what could be parsed. If a hand-checking strategy was to be used, it was clear that even if we could process another 1000 QM/MMGBSA molecules there would not be time to hand-check them all. Introducing molecules with potentially a mere 0.8 similarity to this process was not helpful, and I think a better strategy would have been to be more systematic with the changes made to the molecules. This could have been in the form of further analysis on the other groups molecules to identify possibly beneficial changes, or simply mapping out the extremely local space around the known binder.

Overall, the CACHE2 drug discovery challenge provided a great challenge and resulted in the identification of a novel binder. It provided a first experience of the work involved in finding novel binders and alerted me to many current issues facing drug designers alongside the ways computational researchers can alleviate many of these problems. This became the basis for the next Case Study, where focus turns to quantifying how much a similar pipeline can be improved through introducing ML strategies to the ligand selection stage. While an experimentally confirmed inhibitor of NSP13 still eludes the scientific community, perhaps one of the molecules from this challenge can provide the LO basis for a new treatment. Since the contest has concluded, a novel experimentally confirmed SF1B helicase inhibitor for PIF1 points towards hope in this direction[675].

3.3.6 Appendix

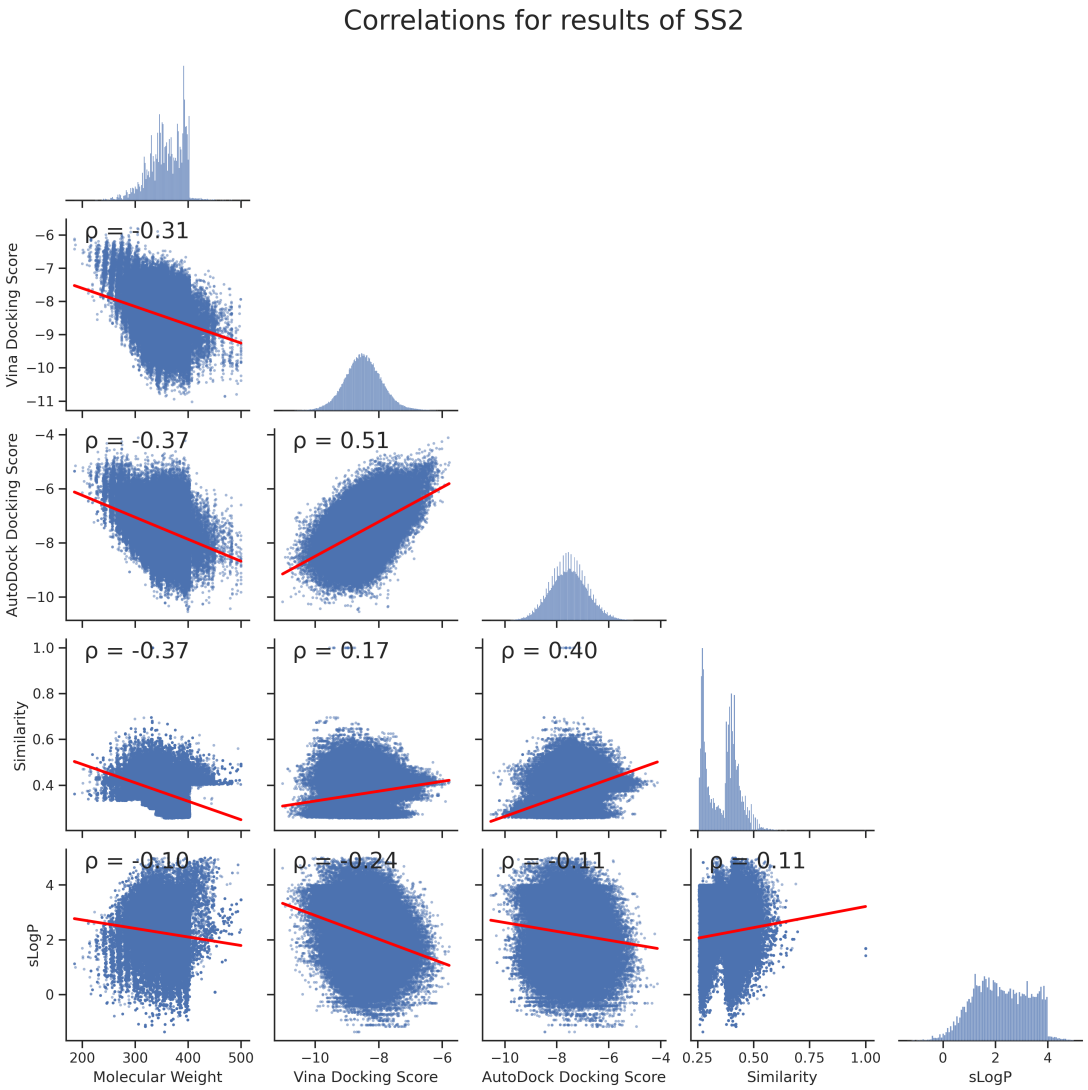


Figure 3.12: Scatter plots, distributions, and correlations (ρ) across various molecular properties and docking results for the top 10% of SS2 molecules.

			SPR Binding Assay: Primary screening at $50\mu M$				SPR Binding Assay: Dose response				Catalytic ATPase Assay				
ID	**Molecular Weight** (Da)	**cLogP**	**Adjusted relative** $50\mu M$	**Slope**	**Slow Dissociation**	**Rmax > Expected Rmax**	**NSP13 KD** (μM) **1**	**% Binding 1**	**NSP13 KD** (μM) **2**	**% Binding 2**	**ATPase % inhib at 50** μM **1**	**ATPase % inhib at 50** μM **2**	**ATPase IC50** (μM)	**ATPase IC50 Hill Slope**	**Advanced to Round 2**
1	373	2.7	0.2	No	No	No	0	0	0	0	1				
2	309	1.4	0.26	No	No	No	0	0	0	0	20				
3	433	2.4	0.4	No	Yes	No	0	0	0	0	0				
4	393	2.7	0.26	No	No	No	0	0	0	0	0				
5	271	-0.7	-0.31	No	No	No	0	0	0	0	0				
6	437	3.2	0.82	No	Yes	No	>1000	>1000	0	0	0				
7	357	1.9	0.1	No	No	No	0	0	0	0	3				
8	283	0.2	0.05	No	No	No	0	0	0	0	23				
9	425	3	0.29	No	No	No	0	0	0	0	0				
10	299	1.1	0.06	No	No	No	0	0	0	0	2				
11	269	-0.2	0.05	No	No	No	0	0	0	0	13				
12	402	2.4	0.55	No	No	No	>1000	>1000	0	0	2				
13	369	2	0.1	No	No	No	0	0	0	0	1				
14	269	-0.4	0.06	No	No	No	0	0	0	0	17				
15	383	-0.1	0.2	No	No	No	0	0	0	0	4				
16	297	0.7	-0.12	No	No	No	0	0	0	0	13				
17	433	1.9	0.33	No	No	No	0	0	0	0	1				
19	283	-0.1	0.1	No	No	No	0	0	0	0	17				
20	436	0.7	0.06	No	No	No	0	0	0	0	0				
21	310	1.1	-0.34	No	No	No	0	0	0	0	2				
22	271	0	0.07	No	No	No	0	0	0	0	15				
23	364	2.8	0.34	No	No	No	0	0	0	0	32				
24	382	3.4	0.12	No	No	No	0	0	0	0	1				
25	311	1.2	0	No	No	No	0	0	0	0	14				
26	446	2.7	0.16	No	No	No	0	0	0	0	0				
27	432	1.9	0.01	No	No	No	0	0	0	0	0				
29	365	2.6	0.13	No	No	No	0	0	0	0	7				
30	341	-0.6	-0.04	No	No	No	0	0	0	0	0				
31	400	1.6	-0.15	No	No	No	0	0	0	0	9				
32	395	1.7	0.06	No	No	No	0	0	0	0	25				
33	335	0.7	-0.18	No	No	No	0	0	0	0	16				
34	355	1.4	0.18	No	No	No	0	0	0	0	0				
35	286	-1	0.01	No	No	No	0	0	0	0	13				
36	406	2.1	0.39	No	No	No	0	0	0	0	0				
37	394	2	0.01	No	No	No	0	0	0	0	1				
39	378	1.4	0.33	No	No	No	0	0	0	0	1				
40	392	2.6	0	No	No	No	0	0	0	0	1				
41	356	2	0.36	No	No	No	0	0	0	0	69	7			
42	382	2.4	0.9	No	No	No	207	404	0	0	13	80	4	0	
43	263	0.2	0.01	No	No	No	0	0	0	0	2				
44	328	0.8	0.07	No	No	No	0	0	0	0	1				
45	363	2.1	0.65	No	No	No	>1000	>1000	0	0	0	5			
46	408	2.7	1.22	No	No	No	0	0	0	0	0	2			
47	295	-0.4	0.05	No	No	No	0	0	0	0	1				
48	402	2.5	0.21	No	No	No	172	71	>1000	0	80		NI	NA	
49	334	2	0.19	No	No	No	0	0	0	0	15				
50	474	0.4	0.1	No	No	No	0	0	0	0	19				
51	417	1.8	0.28	No	No	No	0	0	0	0	1				

			SPR Binding Assay: Primary screening at $50\mu M$				SPR Binding Assay: Dose response				Catalytic ATPase Assay				
ID	**Molecular Weight** (Da)	**cLogP**	**Adjusted relative** $50\mu M$	**Slope**	**Slow Dissociation**	**Rmax > Expected Rmax**	**NSP13 KD** (μM) **1**	**% Binding 1**	**NSP13 KD** (μM) **2**	**% Binding 2**	**ATPase % inhib at 50** μM **1**	**ATPase % inhib at 50** μM **2**	**ATPase IC50** (μM)	**ATPase IC50 Hill Slope**	**Advanced to Round 2**
52	332	0.8	0.12	No	No	No	0	0	0	0	12				
53	332	0.7	0.02	No	No	No	0	0	0	0	15				
54	375	2.2	0.92	No	No	No	>1000	>1000	0	0	3				
55	322	0.4	0.08	No	No	No	0	0	0	0	9				
56	398	0.3	0.05	No	No	No	0	0	0	0	12				
57	334	1.2	0.63	No	Yes	No	0	0	0	0	14				
59	418	0.9	0.02	No	No	No	0	0	0	0	13				
60	316	2.9	0.94	No	Yes	No	0	0	0	0	17				
61	434	3	0.93	No	No	No	495	244	894	580	11	2			
62	321	1	0.27	No	No	No	0	0	0	0	22				
63	445	2.3	0.25	No	No	No	0	0	0	0	1				
64	397	3.5	0.09	No	No	No	460	52	595	55	75	82	>100	-2	
65	389	-0.2	0.06	No	No	No	0	0	0	0	13				
66	305	1.2	0.09	No	No	No	0	0	0	0	9				
67	445	3.8	0.35	No	No	No	0	0	0	0	11				
68	330	0.1	0.07	No	No	No	0	0	0	0	3				
69	393	2.2	-0.03	No	No	No	>1000	>1000	0	0	61	88	55	-2	
70	253	-0.8	-0.01	No	No	No	0	0	0	0	2				
71	403	3.5	0.31	No	No	No	0	0	0	0	0				
72	377	0.6	-0.24	No	No	No	0	0	0	0	1				
74	345	-0.3	-0.83	No	No	No	0	0	0	0	1				
75	272	3.2	0.4	No	No	No	0	0	0	0	20				
76	370	1	0.1	No	No	No	0	0	0	0	2				
77	380	0.2	-0.37	No	No	No	0	0	0	0	1				
78	399	0.2	-0.64	No	No	No	0	0	0	0	6				
79	394	2.3	0.05	No	No	No	0	0	0	0	1				
80	334	3	0.11	No	No	No	0	0	0	0	1				
81	354	1.6	0.01	No	No	No	0	0	0	0	0				
82	406	1.9	0.34	No	No	No	0	0	0	0	0				
83	367	3.2	0.58	No	No	No	>1000	>1000	0	0	1	1			
84	343	1.9	0.05	No	No	No	0	0	0	0	0				
85	347	1.6	2.09	Yes	Yes	No	115	277	0	0	92	86	ND	ND	
86	388	1.3	0.16	No	No	No	0	0	0	0	1				
87	417	0.6	0.38	No	No	No	0	0	0	0	2				
88	404	0.1	0.12	No	No	No	0	0	0	0	3				
89	310	-0.4	-0.07	No	No	No	0	0	0	0	0				
91	313	-1	0	No	No	No	0	0	0	0	11				
92	369	0.4	0	No	No	No	0	0	0	0	2				
93	381	2.2	0.03	No	No	No	0	0	0	0	3				
94	404	3.6	0	No	No	No	0	0	0	0	80	15			
95	418	2.9	0.75	No	No	No	>1000	>1000	0	0	0	1			
96	331	1	0.41	No	No	No	36	42	>1000	>1000	89		24	-1	+
97	333	1.4	-0.02	No	No	No	124	5	0	0	80	58	>150	-1	
98	336	2.3	1.1	No	No	No	>1000	>1000	0	0	0	2			
100	379	2.2	0.29	No	No	No	0	0	0	0	0				

Table 3.6: Experimental results from round 1 of the CACHE Challenge, where molecule CACHE_1451_96 showed the greatest binding affinity and was automatically put forward for another round of experiments. SMILES strings of corresponding molecules are shown in Fig. (3.7) along with molecular diagrams in Fig. (3.13).

CACHE ID	SMILES
1	C(CC(=O)OCc1cccc(C#N)c1)C1=Nc2ccccc2C(N1C1CC1)=O
2	C1C[C@@H](C[C@@H](C1)C(O)=O)NC(C1(CC2CCC1C2)C(O)=O)=O
3	C1CCc2nnc(c3ccc(cc3)NC(CCN3C(C4CC=CCC4C3=O)=O)=O)n2CC1
4	CC1C(c2ccccc2O1)NC(Nc1ccc(CN2C(CCCC2=O)=O)cc1)=O
5	C[C@@H]1CN(C[C@H]1C(O)=O)C(C1(CCCO1)C(O)=O)=O
6	C1CN(CCC1Cn1ccnn1)C(c1nc(c2cccs2)n(c2cccc(c2)F)n1)=O
7	CN1C(NC(c2c1nc(N1CCC(C1)c1nc(C3CC3)no1)n2C)=O)=O
8	C1CCC(C1)(C(O)=O)C(N1CCC(CC1)CC(O)=O)=O
9	C1CC(Cc2ccccc2[Cl])N(C1)C(C1CCN(CC1)c1cncc2nnnn12)=O
10	CC1=C(C)N=CN(CC(COc2ccc(C#N)cc2)O)C1=O
11	C[C@@H]1CN(C[C@H]1C(O)=O)C([C@H]1C[C@@H](CC(O)=O)C1)=O
12	Cc1ccc2nc(CN3C(C(C)(c4cccc(c4)n4cnnn4)NC3=O)=O)cn2c1
13	C1Cc2cc(ccc2NC1=O)OCCNC(N1CC2C(C1)C1C=CC2O1)=O
14	C1CN(CCC1C(O)=O)C([C@H]1CC[C@H](C1)C(O)=O)=O
15	CC(c1ccc2c(c1)NC(CO2)=O)NC(N1CCC2=C(C1)N=CN(C)C2=O)=O
16	CC1(C)C(C)(C)C1(C(O)=O)C(N1CCC(CC(O)=O)C1)=O
17	C1CCCN(CC1)S(N1CCN(CC1)c1ccc2nnc(C(F)(F)F)n2n1)(=O)=O
19	C1CCN(C(C1)C(O)=O)C([C@H]1CCC[C@H](C1)C(O)=O)=O
20	C1CN(CCC1C(Nc1ccccc1C(N1CCOCC1)=O)=O)c1ccc2nnnn2n1
21	Cc1cc(C#N)ccc1C(N1C[C@@H](CC1c1ncc[nH]1)OC)=O
22	CC(C)C(C(O)=O)C(N1CCC(C)(CC1)C(O)=O)=O
23	Cn1c(c2cccc(c2)NCC(NC2CCOc3ccccc23)=O)nnn1
24	CC(C)Oc1ccc(NC(N2CC(C3COCCC23)c2ccccc2)=O)nn1
25	C1CCCC(CC1)(C(O)=O)NC([C@H]1CCC[C@H](C1)C(O)=O)=O
26	CCC1C(Nc2ccc(cc2O1)C(C)NC(CCc1c(C)nc2c(C#N)cnn2c1C)=O)=O
27	CC1=C(C(N(C)N=C1C)=O)c1nc(CN2C[C@H](C[C@@](C2)(C(F)(F)F)O)OC)cs1
29	CC1COC(CN1C(c1cc(C)nnc1)=O)c1cccc(c1)C(F)(F)F
30	COCC(NCCN1CCOC[C@@]2(C1)COC[C@H]1CCCCN12)=O
31	CC1CCCCC12C(N(CC(N1CCN(CC1)c1ccc(cc1)O)=O)C(N2)=O)=O
32	CC1CN(Cc2ccccc2O1)C(Nc1cccc(c1)N1C(COCC1=O)=O)=O
33	CCc1nc(CN2CC(c3nc(COC)no3)C3(C2)COC3)on1
34	CCc1c(c(C)no1)NC(N1CCC(C1)Nc1ccc2nccn2n1)=O
35	COCc1nc(CN2CCC(CC2)(C(N)=O)O)sn1
36	C1CCc2nnc(C3CCCN(C3)c3ccc4nnc(C(F)(F)F)n4n3)n2CC1
37	COc1ccc(CCNC(c2ccc3C(N(CC=C)C(c3c2)=O)=O)=O)cc1OC
39	C1CN(CCC1c1nc2ccccc2[nH]1)C(c1cccc(c1)OCC(N)=O)=O
40	CC(C)c1cc(C(N2CCC(Nc3ccncc23)=O)=O)c2cnn(C(C)C)c2n1
41	COc1cc2c(cc1OC)[nH]c(CN1CCC(CC1)Cc1cnn[nH]1)n2
42	C1CN(CCC1=C1C(Nc2cc(ccc12)[Cl])=O)C(NCc1cccnc1)=O
43	Cc1nc(CCCN2CCC[C@@H]2c2ncno2)on1
44	CC1C(CCC(N1)=O)NC(c1ccccc1Oc1cnn(C)c1)=O
45	CC(C(Nc1cc(ccc1OC)[Cl])=O)N1CCC(CC1)c1c[nH]nn1
46	C1CCCN(CC1)c1ccc(CNc2ccnc(c2)C(N2CCC(CC2)O)=O)cc1
47	COc1cc(CN2CCCN(CC2)CC(CO)O)cnc1
48	C1CC(c2cnn(Cc3ccccc3)c2C1)NC(c1ccc2c(c1)OCC(N2)=O)=O
49	CN1C(c2cc(c3nc(C4CCOC4)no3)sc2N(C)C1=O)=O
50	CCS(N1CCN(CC1)CC(Nc1cc(ccc1C)S(N1CCOCC1)(=O)=O)=O)(=O)=O

CACHE ID	SMILES
51	CC1CC(C)CN(C1)S(c1cccc(c1)C(NCc1nnc2CCCn12)=O)(=O)=O
52	C(CN1C(CSC1=O)=O)NC(c1cccc(c1)c1ncon1)=O
53	CCS(NC1CCN(CC1)CC1CC2(C1)COCCO2)(=O)=O
54	CC1(C(NC(N1)=O)=O)c1cccc(c1)NC(c1cnn(c1)c1ccccc1)=O
55	CN1C(c2cc(C(NN3CCCC3=O)=O)sc2N(C)C1=O)=O
56	CS(c1ccc(cc1)OCCCN1C[C@@H]2CS(C[C@@]2(C1)C#N)(=O)=O)(=O)=O
57	Cc1c(CNC(c2cc3C(N(C)C(N(C)c3s2)=O)=O)=O)con1
59	CS(c1ccc(CN2CCCC(C2)c2c(CN3CCOCC3)cn[nH]2)cc1)(=O)=O
60	CCc1nc(c2cccc(c2)NC(N2CC[C@@H]([C@@H]2C)O)=O)no1
61	COc1cc(ccc1C1(CN(CCC(Nc2ccccc2OC)=O)C1)O)[Br]
62	C(CO)CSc1ccc(cc1)C(N1[C@@H]2CO[C@@H]3[C@H]1CO[C@H]23)=O
63	CC(C)N(C)S(c1ccc(cc1)NC(CN1CCCCC(C1)S(C)(=O)=O)=O)(=O)=O
64	COc1cc(cc(c1OC)OC)c1nc(c2c(csn2)[Br])on1
65	C1CCN(CC1)S(N1CCN(CC1)Cc1cnc2c(C#N)cnn2c1)(=O)=O
66	CCN1C(c2ccc(cc2NC1=O)C(NCCOCC)=O)=O
67	Cc1cccc(c1)OCCNC(CN1CCN(CC1)S(c1cc(C)ccc1C)(=O)=O)=O
68	C1COc2ccccc2CN1C(c1cn([C@@H]2COC[C@H]2O)nn1)=O
69	CC1C(Nc2cc(ccc2O1)NC(CCN1C(c2ccc(C)cc2C1=O)=O)=O)=O
70	C1COCC(C(O)=O)N1c1cnc(cn1)C(O)=O
71	Cc1ccc(c2c1CCCN2C(CCc1c(C)nc2c(C#N)cnn2c1C)=O)OC
72	Cc1cccc2C(N(CCC(N3CCC(Nc4ccncc34)=O)=O)C=Nc12)=O
73	CN1C(c2ccc(cc2C1=O)C(N1CCN(CCOC)C(C1)=O)=O)=O
74	C(C(O)=O)c1cccc(c1)c1ccc(C(O)=O)c(c1)O
75	CC1(C)C(Nc2cccc2N1C(CN1C(C2(CCCC2)NC1=O)=O)=O)=O
76	Cn1c2CC(CCc2cn1)C(NC1CCN(CC1)C1=C2C(N=CN1)=NC=N2)=O
77	C1CN(CCC1C(N1CCOCC1)=O)C(CCN1C(c2ccccc2C1=O)=O)=O
78	Cc1ccc(cc1N1C(CNC1=O)=O)NC(N1CC(Cc2ccccc12)OC)=O
79	C1CC(N(C1)c1ccc(CC(NC2CCc3ccccc23)=O)cc1)=O
80	Cc1c2C(N(CC3CCN(C3)C(c3ccccc3)=O)C=Nc2sn1)=O
81	CC(c1ccc2c(CCC(N2)=O)c1)NC(NC(C)c1ccc2c(CCC(N2)=O)c1)=O
82	C1CC(c2ccccc2C1)OCC(COc1ccc2c(CCC(N2)=O)c1)O
83	C1C(C(N(C1=O)c1cccc(c1)O)=O)Sc1ccccc1C(O)=O
84	Cc1cc(C)n(c2ccc(CNC(c3ccc4c(c3)nn[nH]4)=O)cn2)n1
85	CC(C)C1C(CCO1)C(NC1CCN(CC1)S(C1CCOCC1)(=O)=O)=O
86	C1CN(CCC12C(NC(N2)=O)=O)C(c1ccc(cc1)n1cc(cn1)[Br])=O
87	C1C(CN(Cc2ccccc2[Cl])C1=O)C(N1CCC2(CC1)C(NC(N2)=O)=O)=O
88	COCCc1nc(CN[C@@H]2CCC[C@@H]2N2CCC(C2)O)no1
89	C1C[C@@H]2C(N(C(N2C1)=O)c1cccc(c1)N1C(CCC1=O)=O)=O
91	CCn1c(CNC(N[C@H]2C[C@H](C)S(c3c2ccs3)(=O)=O)=O)ncn1
92	C1CCC(C1)n1ccc(CNC(N2C[C@@H]3C[C@H](C2)C2=CC=CC(N2C3)=O)=O)n1
93	Cc1ccc(c(c1)O)NC(Cc1c(C)nn(c2ccc(cn2)C(F)(F)F)c1C)=O
94	CCN1C(C(Nc2cc(ccc12)C(N1CCC(CC1)c1nc2ccccc2o1)=O)=O)=O
95	c1ccc2C(NN=C(c2c1)c1nc(c2cnc3cnccn23)no1)=O
96	c1ccc2c(c1)ncc(C(Nc1cccc3C(NNC(c13)=O)=O)=O)n2
97	C1CC(Nc2ccc(cc2C1)NC([C@H]1C[C@@H]1c1cccc(c1)O)=O)=O
98	CC(C(NC(N)=O)=O)Nc1ccc(CCn2nc(c3ccccc3)nn2)cc1
100	

Table 3.7: SMILES strings for the molecules experimentally tested in round 1 of the CACHE Challenge. Our selected molecule, CACHE_1451_96, which showed promising results is highlighted in green.

3.4 Case Study 5: Active-Learning to Improve Ligand Selection

3.4.1 Overview

While Case Study 4 provided a practical introduction to the world of large scale screening, this project looked to build upon this basis to alleviate one of the key issues encountered in effective ligand selection. Docking-based scoring had shown to be unable to provide a consistent selection of optimal compounds, losing a significant amount of time to downstream testing of poorly performing molecules. To be able to develop new selection methods and provide a reliable assessment of their benefits, this work looked to conceive a robust test example of a DDP to quantify the enrichment provided by novel ML strategies. To do so, the methodology from Case Study 4 was extended to a new system, utilising a more accurate and system specific calculation designed in previous work within the group. Alongside this a new ML methodology to aid selection was implemented and tested. It culminated in comparing the GNN-based strategy against using the docking score as before. This looked to provide clear evidence of the benefit gained from involving ML training in a DDP, paving the way for a more automated and efficient drug design process while avoiding the critical selection errors observed in the CACHE2 Challenge. The RAS system had been the focus of work in the group for years, and the QM/MM testing method had been already developed leading to my initial introduction on the project to test using ML to select further ligands. After the CACHE2 challenge, the developed DDP was brought into this project, and my role became more involved as it became clear this would be a more difficult task than anticipated. Over time this turned the project from application driven to a more method development contribution, with the aim to provide an effective testing framework and quantitive result highlighting the benefit of ML. During this project I was responsible for designing, implementing, and training the ML experiments alongside data processing and analysis, finding the ligands to test, and docking calculations. My colleagues performed all other work, including parametrizing ligands alongside developing, running, and parsing the QM/MM calculations. It shows a clear extension to our previous work in

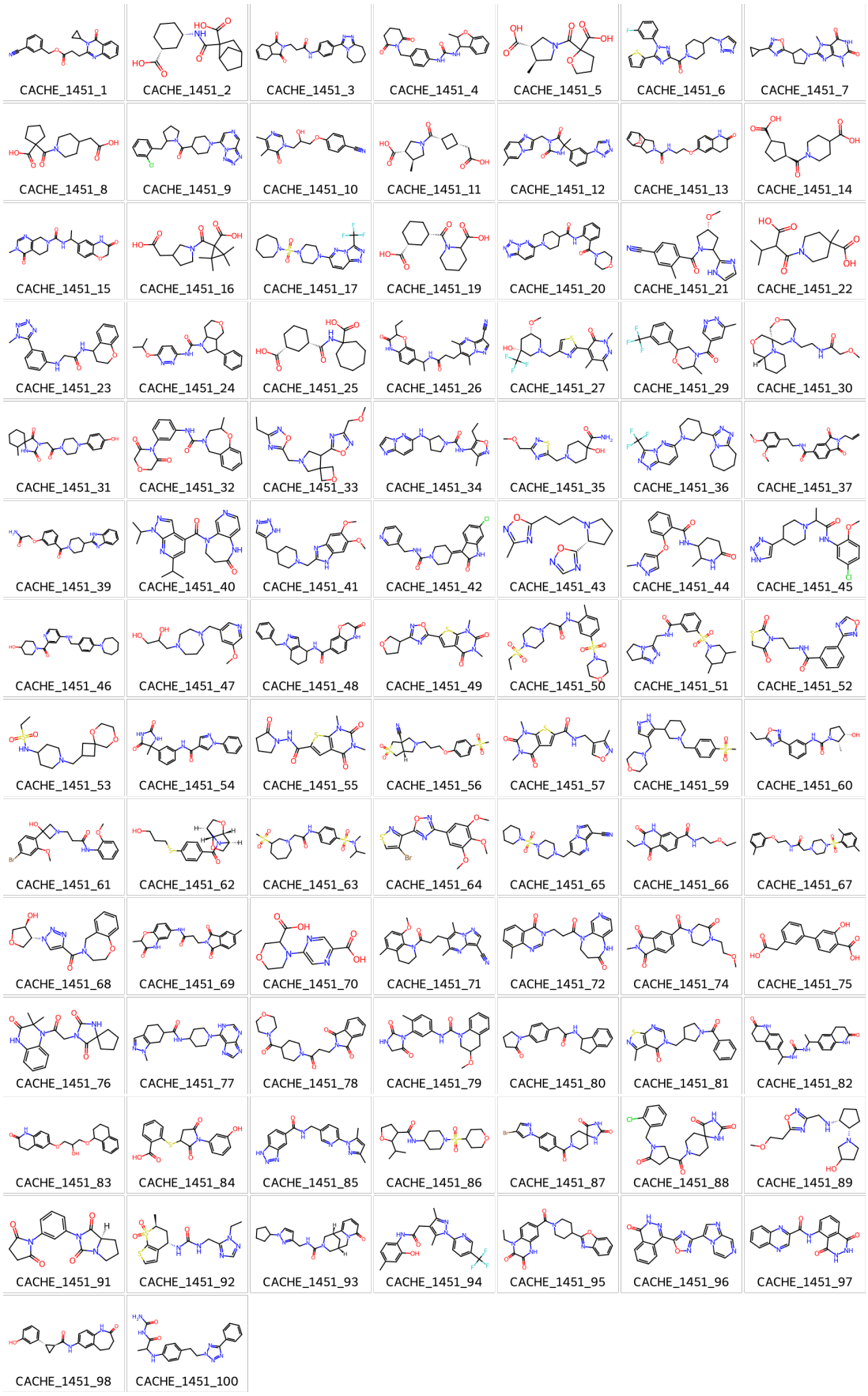


Figure 3.13: Molecular diagrams for the round 1 selected molecules from SS1.

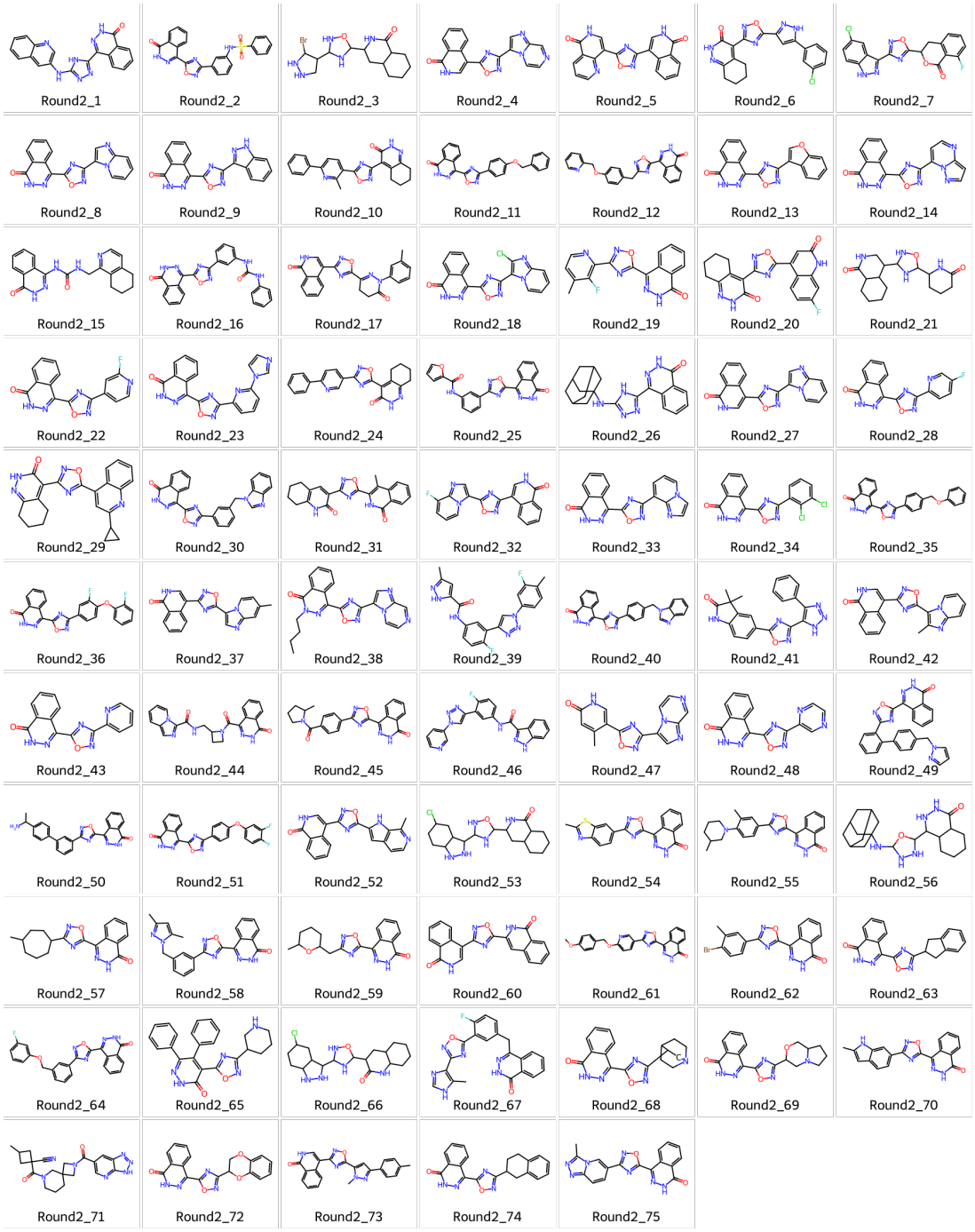


Figure 3.14: Molecular diagrams for the 75 molecules selected for round 2 of experiments. No ligands showed any experimental activity.

the area and aimed to serve as a quantitative example of the predictive power of GNNs in DDPs, highlighting the ease of incorporation at a relatively light computational cost.

3.4.2 Introduction

Many of the strategies and techniques required to work with small molecules have already been introduced in this chapter, but these are continually being improved. Drug discovery is a multi-billion dollar industry, and continues to attract significant resource investment from both the private sector and research communities. As it has been seen, the process of finding new therapeutic compounds is incredibly unforgiving and demanding, requiring some of the highest capital, time, and skill investments in modern science while having a very low hit rate. Successful design processes can take decades, and regularly research and development costs for newly discovered compounds spiral into the billions[676]. However the opportunity to provide therapeutic remedies to the many diseases that plague humanity, along with the seemingly limitless financial upside to owning an in-demand patented drug, keeps researchers and companies interested in finding the next set of BACs. Leading to a significant motive for furthering existing methodologies to be more effective and efficient.

One of the main current issues in this process is the screening of large VS libraries, as performed in Case Study 4. DDPs form the most general modern approach to this problem. While an approach to screening high volumes of ligands has already been introduced, there are still several issues with this process and the community is always looking for ways to improve. The clearest shortcoming is the disconnect between the rankings provided by the different methodologies in a pipeline, as the results of one calculation can often be a poor indication of performance in the next[677–679]. This task is made even more difficult considering the incredible degree of selectivity required not only for the ligand to bind to the target, but to also have the functional effect required.

Purely docking-based procedures are not able to discriminate between pharmacological active and inactive compounds, meaning a significant proportion of computational time is wasted on ligands with little to no significance. However they are often the only available techniques capable of scaling to the sizes required. Therefore being able to predict a ligand's potential activity without performing costly simulations is of the upmost importance. The recent rise of ML strategies provides one possible approach to improving a DDP by enabling a

more informed selection to be made. As downstream data becomes available, this can be utilised to increase the selectivity of the pipeline by using ML to predict the costly methodologies final results from easily accessible data. An effective predictive system would allow for more expensive calculations to be saved for molecules with a greater potential, lowering some of the scaling requirements for DDP processes.

Such a system would also enable more complex calculations to be included in a DDP. Often drug discovery is needed in systems where a targeted process is already well-understood on a chemical level, in addition to there already being known binders available as discussed. In these scenarios, quantum chemists can create a schematic of the reaction catalysed, and QM methodologies can be used to investigate the free-energy barrier associated with this. Utilising this as system specific screening step in a DDP offers a replacement for the generic binding methods outlined in Section (3.2.4), whose efficacy can vary greatly between systems[680–682]. While this gives a much better estimation of a ligands potential, the significantly higher computational cost means fewer ligands are capable of being selected for this process. Therefore maximising the selection efficiency is of paramount importance. Predicting such a complex calculation is a difficult task, and beyond the scope of many classical models due to the complex interactions between the protein and ligand.

Recent advances in GNNs have helped to improve NNs ability to learn from point-cloud data in several areas. They have provided great results in very similar applications across the areas of biology, such as protein structure prediction, molecule generation tasks, and compound property prediction[683]. This lays the groundwork for an even more automated DDP procedure where NNs are capable of selecting what BACs should be tested next. Additional advances in Active Learning (AL) schemes show promise for developing novel strategies to train NNs iteratively alongside DDPs, learning directly from calculation results as more information becomes available[684,685]. Implementing a pipeline specific NN as part of a DDP may become the future of drug discovery, as even a small improvement in efficiency would warrant inclusion if it came at a low enough computational cost. However there is a delicate balance to strike, and there is need for a critical evaluation of the potential upside these methodologies can provide.

RAS shows great potential as a test system for such methods due to an established set of known binders, a well characterised chemical reaction process, and the need for a small molecule inhibitor capable of affecting its various isoforms and mutations. As a kinase it is active in signalling pathways, playing a key role in the MAPK/ERK-pathway dictating cell mitosis upstream from BRAF studied in Case Study 1. RAS is a family of GTPases, and it is natural to talk about the RAS isoforms as one in this context due to the high conservation of their active sites. They bind GTP with a picomolar affinity, and a subsequent GTP-hydrolysis reaction modulates the behaviour of the wild-type[686]. Its bound state is active and propagates downstream proliferation signals, while after the reaction it relaxes into an inert GDP-bound conformation. This reaction mediated by the binding of GTPase-activating proteins (GAPs), which provide natural control over signalling behaviour[687,688]. GAP plays an active role in controlling the reaction with the complexed binding site, shown in Fig. (3.15), being formed of residues from both proteins. Most notable from GAP are a key arginine finger which coordinates the GTP. The active-inactive transition is characterised by the motion of two loops named the Switch I and Switch II located near the GAP interface[689]. These loops are shown in the appendix in Fig. (3.24).

RAS is the most frequently oncogenic mutation across kinases observed in human cancers, present in over 30% of all cases[690]. This has led to its widespread study as a therapeutic target. However difficulties finding RAS inhibitors have earned the system a reputation as the *"un-druggable"* protein[691]. Similar to BRAF, its regulatory mechanism is heavily influenced by mutations local to the active site. Remarkably 80% of all mutations are found at two glycine residues located near the binding site and GAP arginine finger. The 4 most prevalent substitutions are G12C, G12D, G12V, and G13D[692]. These mutations have been shown to increase the energetic barrier to GTP hydrolysis, locking RAS in an active GTP-bound conformation despite GAP binding.

Recent years have delivered some breakthrough in targeting mutant RAS, especially with covalent inhibitors specific to the G12C mutant[693]. Crystal structures show these bind to the pocket formed by the Switch II loop, near to the GAP interface. Covalent binding to the mutated residue in the GDP-bound state reduces the affinity to GAP, locking the protein in an off conformation. This

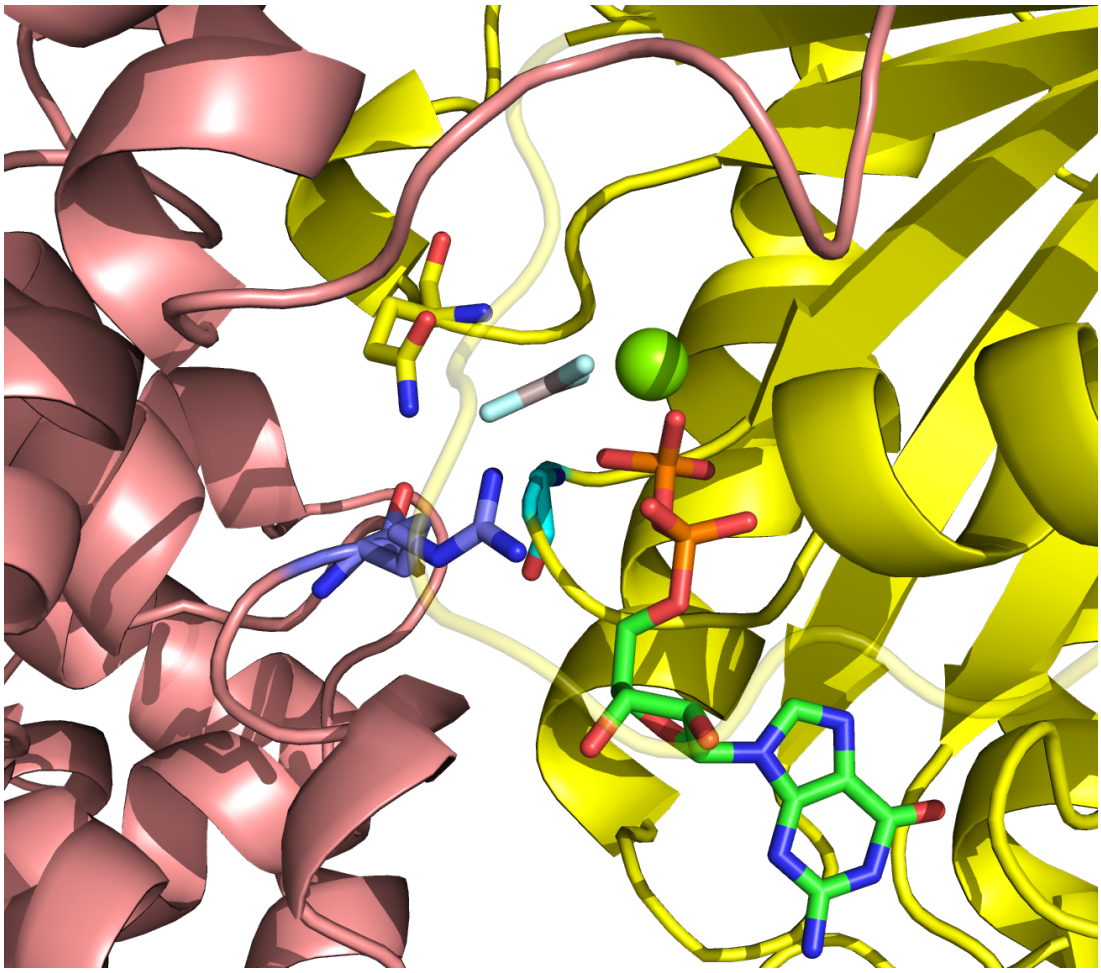


Figure 3.15: A GDP-bound RAS-GAP complex (PBD ID: 1WQ1) showing a close-up of the GTP binding site. RAS is shown in yellow and the p120-RASGAP is shown in pink with the key arginine finger in purple. In the binding site, ADP and a magnesium ion can be seen in green, with a free aluminium fluoride coloured in grey. In cyan on RAS the G12 residue is shown and the glutamine Q61 residue active in the chemical reaction is shown in licorice.

cysteine mutation provides an ample target for covalent inhibition, however this is selective to only a single RAS mutation. Treatments taking this approach include the approved drugs adagrasib and sotorosib, and ARS-853[694–696].

A few non-covalent approaches have also found success. MRTX1133 has shown selective inhibition of the G12D mutation, and is currently ongoing clinical trials[697]. Experimental structures show this to also bind to the Switch II pocket in an almost identical location to G112C covalent inhibitors. BI-2852 is another non-covalent inhibitor which has been observed binding to the G12D mutation which is the most prevalent in cancers[698,699]. Studies indicate this induces an inhibitory

dimerisation of RAS molecules. Unlike the other binders considered, it has been crystallised in the region between the Switch I and Switch II loops, known as the Switch I/II pocket. Despite being non-covalent, both these binders operate using the same mechanism as the previous covalent inhibitors by reducing RAS-GAP affinity.

While compromising the complex formation capabilities of RAS-GAP has been shown to be a highly effective strategy for inhibiting oncogenic activation, it is not applicable to all RAS isoforms and mutations. An alternate strategy for returning to wild-type function in mutated systems is to instead look to lower the mutation-induced increased barrier to hydrolysis. This was partially achieved for G12C by using a non-native GAP domain RGS3, and earlier work explored the possibility of engineering a mutant-specific GAP based on computational screening[700]. There has been less work on using non-covalent small molecules for this purpose. With the large number of known treatments providing good set of scaffolds to start from, the biological aim for this work then became to investigate small molecules capable of reducing the RAS G12D catalytic barrier in the ambitious search for a pan-RAS inhibitor.

In this work using RAS-G12D as an example we aim to show how a small GNN can be used to markedly improve the quality of selected ligands. It looks towards developing a quantitative assessment for a methodology implementing a GNN in a DDP as an additional screening step. It begins with a short theory section introducing some of the recent key developments for this methodology, which is used to inform our approach. Next the methodology for this work is shown, where a recently developed fast hydrolysis barrier calculation technique is used to asses the influence of ligand binding. While there is no substitute for simply running these complex calculations, this work looks to investigate if the available technology is already enough to notably help with the selection of drug-like molecules in a practical workflow. This is the focus of this final case study, as GNNs are applied to predicting RAS hydrolysis barriers to see if the top performing molecules in a dataset of around 10,000 can be found more efficiently. Further application of this work may be used in the hunt for the elusive pan-RAS inhibitor.

3.4.3 Theory

Active Learning

AL has emerged as a paradigm for learning in a continual process, rather than the canonical train, validate, predict pipeline[701]. It focuses on training models using the least amount of data possible, focusing its usage on situations where data-labelling in prohibitively expensive[702]. This enables more complex training strategies to emerge in large iterative pipelines like in drug discovery, rather than solely building predictive models from datasets of final results. Such a strategy is alternatively known as online learning, and has found popularity under this name in usage on recommender systems for online services, where large quantities of user data are continually processed[703,704].

All AL methodologies are split into two stages, training and querying. Training consists of the standard backpropagation of error gradients, as with all ML methodologies. It is separated from usual approaches by the small size of the initial set of data, chosen to be as minimal as possible while still providing adequate information to the NN for some level of prediction. After a predetermined number of epochs, appropriate to avoid overfitting on such a small dataset, the querying stage begins. During this process model inference is performed across all other potential data points, and some selection strategy is used to augment the training set with some further samples from this. It is an iterative process, as training can be performed repeatedly and queries can continually select new data points for testing based on some pre-defined inclusion criteria.

Generalised ML strategies built from small datasets are prone to major errors, as stable areas of parameter-loss space are sometimes unable to be found when using little data. This can be caused by random fluctuations in performance due to model initialisations, general numerical instability during the early training process, or a poor support given by the subset of the dataset, although it is impossible to pinpoint an exact reason. Parallels can be drawn with the issues observed in Case Study 3, where they were circumvented by training a large number of models initialised over a large portion of parameter space. AL approaches implement a similar strategy, under the name of an ensemble approach, to mitigate this by training a number of models simultaneously. Unlike the parallel minimisation

runs performed in Case Study 3, in ensemble strategies each model contributes to a general prediction which can be interpreted as the total model. Ensemble models have found wide usage across modern ML in both the physical sciences and beyond, and is not just limited to AL as a strategy[705–707].

For active learning, one of the most common approaches to this is known as a query by committee, first conceptualised in the early days of the field[708,709]. While this early work is very theoretical and looks for an analytical formula to maximise the entropic gain of each chosen sample, modern implementations are significantly simplified. In most current strategies, the number of added data samples per query step are predefined and the next set of additions to the training set are decided using elementary statistics across the predictions. Common choices include the maximum variance, the minimum average, or the some combination across the ensembles predictions. These techniques have the potential to greatly aid molecular searches, where a stream of generated data can be used to iteratively update the predictions across the sets of other untested ligands. A dominant strategy in this area has yet to emerge, however there is existing work in the area focusing on applications of such techniques which shall be summarised in the remainder of this section. This highlights some of the work which inspired our approach.

For establishing a strategy for training ML FF potentials, Smith et al. utilised an AL strategy to choose test cases for labelling and subsequent training[710]. In this scenario, the final models looks to be as widely applicable as possible, so the strategy looked for models which the FF was incapable of accurately predicting. Therefore inclusion was based on whether the observed variance per atom ρ_i in the calculated force for test case i was greater than a set constant value $\hat{\rho}$, calculated using:

$$\rho_i = \frac{\sigma_i}{\sqrt{N_i}} \tag{3.4.3.1}$$

where N_i is the number of atoms in system i. This allowed the training process to hone in on systems where the ensemble models disagree, focusing computational time on the most informative examples. The paper states this led to a 90% reduction in the data required to train an equally performing ML FF for basic organic molecules.

In a similar approach to Case Study 4, Khalak et al. used AL to search for binders for the PDE2 system[711]. Their approach stuck to simple 1D molecular representations, testing a variety of approaches as they searched for molecules performing well on general AFE calculations. Docking software was avoided by generating complexes through direct replacement of ligands found in crystal structures with those from a VS library, and minimised using MD. This work tested a selection of 5 various inclusion criteria:

- **Random** ligand selection.
- **Greedy** selection, taking only the best predictions.
- **Narrowing** where several models are trained on different representations for the first 3 iterations, the models with the lowest RMSE are chosen to continue, and greedy is used for subsequent epochs.
- **Uncertain** selects the ligands with greatest variance in predictions.
- **Mixed** identifies 3× the selection size using the greedy approach, then selects the most uncertain from these.

The work identified several molecules showing good binding affinity with a common pose and structural scaffold, however there was no conformation of this beyond some AFE calculations. In terms of maximising the performance of the AL strategy, it was found that introducing 3D coordinates to the descriptors performed the best, indicating the vital importance of structural information. It additionally identified a greedy strategy as being able to locate the binders more efficiently, although identifying the risk associated with the leading to the NN becoming trapped in a local minimum if the chosen set became too uniform.

Yang et al. look to establish a methodology for navigating VS libraries vast chemical space using an AL methodology and SMILES-derived features[712]. It focused on efficiently predicting GLIDE and DOCK software docking scores across the ZINC VS library. This work identified using a combination of taking the most uncertain ligands from the top predicted subset as the most effective in finding optimal ligands. It found pure uncertainty based criteria to be less effective than greedy ones. Testing of different selection sizes each iteration pointed towards

larger selections being more effective, especially when considering the early stages of AL. In this study, AL based methods were shown to outperform similarity measures to provide a high (> 0.6) degree of correlation with docking scores.

In their approach to a similar screening methodology, Gentile et al. appended active compounds sourced from the DUD-E library to the ZINC15 VS library[713]. The 1.36 billion entries were then screened using 12 prominent drug targets. The performance of the model was evaluated using the enrichment E_N^*:

$$E_N^* = \frac{[\text{Proportion found via AL}]_N}{[\text{Proportion expected from a random selection}]_N} \tag{3.4.3.2}$$

where N is a chosen size indicating the top N ranking molecules. The work found introducing AL strategies to consistently enrich selections, although the degree of improvement was target dependent.

Thompson et al. looked to build a predictive model capable of predicting the results of relative FEP calculations on the TYK2 kinase[714]. The study utilised a dataset built from a single scaffold of a known inhibitor, significantly restricting the molecular space required as support for the model. In a thorough benchmark, they tested the selection size, ML architectures, various inclusion criteria, and initialisation schemes for the initial set while using ECFP6 fingerprints as input features. Their study never extended to GNN models, but found the choice of model and inclusion criteria to be largely insignificant on performance. Instead the selection size was found to provide the greatest impact on model performance. They found their methodology was capable of finding 75% of the top 100 molecules by sampling only 6% of the overall database.

ML Approaches to small molecules

Representing small molecules in ML methodologies is a difficult task, and there have been several proposed improvements to GNNs in order to better represent small molecules.

At its core, the data is point-cloud data with node-level features associated to the underlying atom types. This is perhaps the most natural representation, and has seen wide-spread usage throughout the area for molecular property prediction tasks[715]. It allows for general GNN strategies to be used, yet leaves

significant freedom for case specific descriptors which may not be non-trivial to find. This allows the application of generic GNN layers, such as GCNs, GATv2, and GraphSAGE as outlined in Section (1.2).

In addition to these generic GNN layers, specific graph layers have been proposed for use with small molecules[716,717]. Molecular fingerprint layers were devised to best emulate the properties of circular fingerprints, detailed in Section (3.2). They mimic the generation process by using different layer weights not only at each depth but also depending on the degree of the node considered $\deg(v)$. This allows the passed messages to be simply the concatenated feature vectors, meaning each subsequent layer receives information from nodes one atom further away. When using this method, the number of layers then becomes analogous to the neighbourhood size of a circular fingerprint. In a message passing formalism, they can be viewed as using the following functions for message passing and vertex updating:

$$M(h_i^t, h_j^t) = [h_0^t || \dots || h_j^t], \quad \{0, \dots, j\} \in \mathcal{N}(v) \tag{3.4.3.3}$$

$$U_t(h_i^t, m_i^{t+1}) = \sigma((W_t^{\deg(v)})^T m^{t+1}) \tag{3.4.3.4}$$

Alongside this they are commonly implemented with skip connections across each layer, which can also be weighted. While the number of weights in this methodology are significant, many of the properties of molecular fingerprints are similar to what is desired from a screening GNN so this is seen as a reasonable computational expense. It has become a common tool used in many areas dealing with small molecules[718,719]. One benefit of this is that it gives the network another way to differentiate between different atomic types at a local level, allowing it to form some sense of functional groups. However in scenarios such as this, where data is very limited, the number of samples for some atom types and the large number of weights may be too small for a reliable estimation over all degrees when extrapolating.

When testing NNs on molecular datasets, the general randomised approach to forming train-test data splits can have drastic consequences on the accuracy of models. The similarity between ligands in the train and test set has been observed to have a large impact on model performance, as having more similar

sets can be seen as a manifestation of data leakage. To avoid this and provide an adequate challenge for ML methodologies, ligand datasets may be batched into test-train splits based on the similarity of their underlying scaffold[720]. This places similar groups of ligands into separate sets, meaning the network will be tested on its ability to extrapolate effectively. Such a strategy was first introduced in DeepChem package of utilities for working with deep NNs on chemical datasets[721]. However these approaches have been criticised for creating artificial datasets leading to benchmarked performance being weaker than observed in practical usage[722]. Therefore their implementation must be justifiable on a case specific basis, and researchers should be careful not to report performances unachievable at an application level.

3.4.4 Methodology

The aim is to find ligands able to restore the activity of G12D RAS to that of the wild type. This requires a lowering of the hydrolysis barrier to its native level, increasing the rate of hydrolysis reactions to match the wild-type and preventing the oncogenic continual activation. The enhanced sampling QM/MM simulation was designed in previous work to accurately simulate the reaction mechanism, and a detailed discussion of it can be found in previous published paper[723]. To summarise, the methodology initially built a clear reaction mechanism for RAS hydrolysis. It then defined a small QM region around a bound GTP, and used enhanced QM/MM sampling methods to derive a minimum free energy profile for the reaction over a RC representing the reaction. This barrier along with chemical diagrams of the process are shown in the appendix in Fig. (3.25). To improve the performance of this methodology, subsequent simulations could be performing by assuming the QM wave function is unaltered while changing the MM region and re-optimising coordinates using MM minimisation across the RC. Final free energy estimates could then be made by performing a single-point QM energy evaluation at the value of the RC corresponding to the main catalytic barrier giving a scalar estimate of the barrier height. In the original study, this simplified method was used to evaluate barriers across the various RAS mutations.

The simulation method was shown to be consistent with the decreased hydrolysis rates of the RAS mutants. This provided a clear indication of its physical accuracy and more confirmation of the mutations increasing the reaction barrier. In this work, the QM/MM methodology was extended towards instead varying the MM region by introducing small compounds rather than mutated residues. Such an extension is capable of scaling to thousands of ligands in a feasible time, relying on a couple of critical assumptions:

- The apo path is appropriate to model the reaction with the ligand in place, implying the ligand has no affect on the reaction RC.
- The QM region is not re-optimised with the ligand which is docked in the MM region, assuming the ligand has no structural affect in the catalytic region.

Both of these are fairly strong assumptions. However a chemical analysis of ligand affects would put this approach outside the realm of a large scale automated approach, and involving further QM optimisations would be very restrictive on time. Therefore this is viewed as a suitable compromise to utilise the method as part of a HTS strategy, allowing for a large number of simulations to take place. It is clear that in a different study aiming to directly determine the suitability of found ligands, there is still a requirement for a more thorough approach.

Preliminary work, mostly performed before my introduction to the project, began this process by investigating the effects of small compound binding. In the time following the first paper a set of 4512 ligands were processed using a previous methodology distinct from the screening methodology introduced in Case Study 4. This involved docking using DOCK software to a region encompassing both the Switch I and II pockets[567]. Ligands were selected from the ZINC VS library[29]. All these ligands had associated The dataset was repurposed to provide ligands distinct from the final experiments set, which were used in preliminary testing to develop the feature generation strategy.

At this stage testing focused on developing a robust featurization process that was capable of representing ligands effectively and optimising for correlation with the simulation results. To maximise the challenge placed on the GNN, these ligands were scaffold split to ensure there was fewer similarities between the

sets. This was performed using a strategy similar to the clustering method used in the CACHE2 challenge, by calculating Euclidean fingerprint distances and clustering using K-Medoids into 200 sets,[670]. All ligands were separated using a 90/10 test-train split discretised using these clusters. A validation set was deemed unnecessary as this was preliminary data not to be used in the final assessment of model performance.

As a baseline, performance was measured against a baseline ChemProp model, an alternative GNN approach used to predict molecular properties from SMILES strings[724]. Comparison was used to show if the tested featurization processes provided a valid performance benefit over simpler existing methodologies. This method includes an automated hyper-parameter optimisation protocol, which was performed prior to the final training to ensure this was the best performing model. It selected using 3 GNN layers of hidden size 256, a dropout of 0.05, and 3 linear layers with a hidden size of 128.

Based on previous studies into AL screening procedures, our features looked to incorporate 3D-coordinates from docking complexes into the node features[711]. The basic concept was to use atoms as nodes supplemented with various atomic properties. As the network was fairly small bond connectivity was not incorporated, instead assumed full connectivity between all nodes in a ligand graph. This was chosen as information can propagate poorly in GNNs with few layers over chemical structures such as rings. Several quantities were used or calculated for each ligand atom, indexed by i:

- 3D positions ($\mathbf{r}_i$).
- Vectorised docking FF parameters ($\mathbf{p}_i$), taken from the AutoDock4 FF[593]. This included the atom charges q_i.
- Heavy-atom distances ($\mathbf{d}_{ij}$) to selected pocket and catalytic residues, further explained in Appendix Fig. (3.26).
- Electrostatic potentials (ESPs) at the positions of QM region atoms, calculated using q_i.
- Atom type encodings (λ_i) as one-hot vectors, categorised based on the atomic element.

All features tested used atom-type encoding using a one-hot vector. Internal coordinates were used to account for equivariant properties of the system. ESPs were included as this was theorised to be the main mechanism a bound ligand could interact with the QM region to affect the single-point calculation. A series of combinations of these node features were tested to find the optimal descriptors. The simplest consisted of just the 3D positions and atom types:

$$h_{v_i}^0 = [\mathbf{r}_i \,||\, \lambda_i]$$

Such a simple scheme was used as a baseline measurement. It provided a quantitative assessment of the postgenital impact of using rotationally invariant coordinates and additional atomic properties in subsequent approaches. These features were named *"XYZ Coordinates"*.
The next feature set used internal distances, along with all FF parameters:

$$\begin{aligned} h_{v_i}^0 &= [\mathbf{d}_{ij} \,||\, \mathbf{p}_i \,||\, \lambda_i] \\ &j \in \{\text{pocket residues}\} \cup \{\text{catalytic residues}\} \\ &k \in (\text{QM atoms}) \end{aligned}$$

FF parameters were easily available for all docked ligands, and therefore provided a relatively straight forward approach to convert a docked structure into input for a GNN strategy. This strategy was denoted as *"Internal Distance with NoESPs"*. Subsequently introducing the ESPs was tested:

$$\begin{aligned} h_{v_i}^0 &= \left[\mathbf{d}_{ij} \,||\, \mathbf{p}_i \,||\, \frac{q_i}{\mathbf{d}_{ik}} \,||\, \lambda_i\right] \\ &j \in \{\text{pocket residues}\} \cup \{\text{catalytic residues}\} \\ &k \in (\text{QM atoms}) \end{aligned}$$

Testing these features provided an assessment of our hypothesis about the impact of electrostatics on the single-point calculation result. This features set was named *"Internal Distances with Per Atom ESPs"*.
Finally, just the internal distances, atom types, and solely the charges from the

FF were tested:

$$h_{v_i}^0 = [\mathbf{d}_{ij} \,||\, q_i \,||\, \lambda_i]$$
$$j \in (\text{pocket residues}) \cup (\text{catalytic residues})$$

Simply referred to as *"Internal Distances"*.

The GNN architecture used consisted of 4 separate stages, and hyper-parameters were kept consistent across all performed experiments. An initial featurizer, consisting of two linear layers, was used to process the node features into a node embedding of dimension $d_{\text{hidden}} = 64$. Prior to being fed through the linear layers, atom type encodings were converted into 16-dimensional trainable embeddings[463]. These were then concatenated with the other atom features, independently fed through the featurizer, and inputted to a sequence of 4 GNN layers. This modelled each ligand as a fully connected graph and also used a dimensionality of d_{hidden} and an output size of $d_{\text{embedding}} = 32$. Each GNN layer used batch normalization to enable quicker model convergence and utilised skip connections[725]. The variously sized ligand graphs must then be converted to a consistent dimensionality before further processing is possible. Therefore the output of the GNNs was converted to a ligand embedding of size $3d_{\text{embedding}}$ by concatenating minimum, maximum, and average pooled representation across each node. This was then fed into a simple 4 linear layer network using an internal representation of $d_{\text{linear}} = 32$, and an output size of 1 for regression against the barrier heights. The architecture of the GNN model tested is shown in Fig. (3.16).

To verify an ensemble approach every network was trained 10 times in parallel. For the feature testing experiments this avoided issues due to poor parameter initialisations, and this was an intrinsic part of the AL strategy. The learning rate was set to 10^{-6} and cosine annealing was used to regulate the learning rate over the training process with a factor of 1 and a period of 250 epochs[726]. An MSE loss function was used between the model output and the QM/MM calculated barrier height. GATv2 layers were used for the feature comparison over the ZINC set of ligands and final AL experiments. All experiments were performed locally on a NVIDIA GeForce RTX 4090 GPU with 24GB of dedicated VRAM.

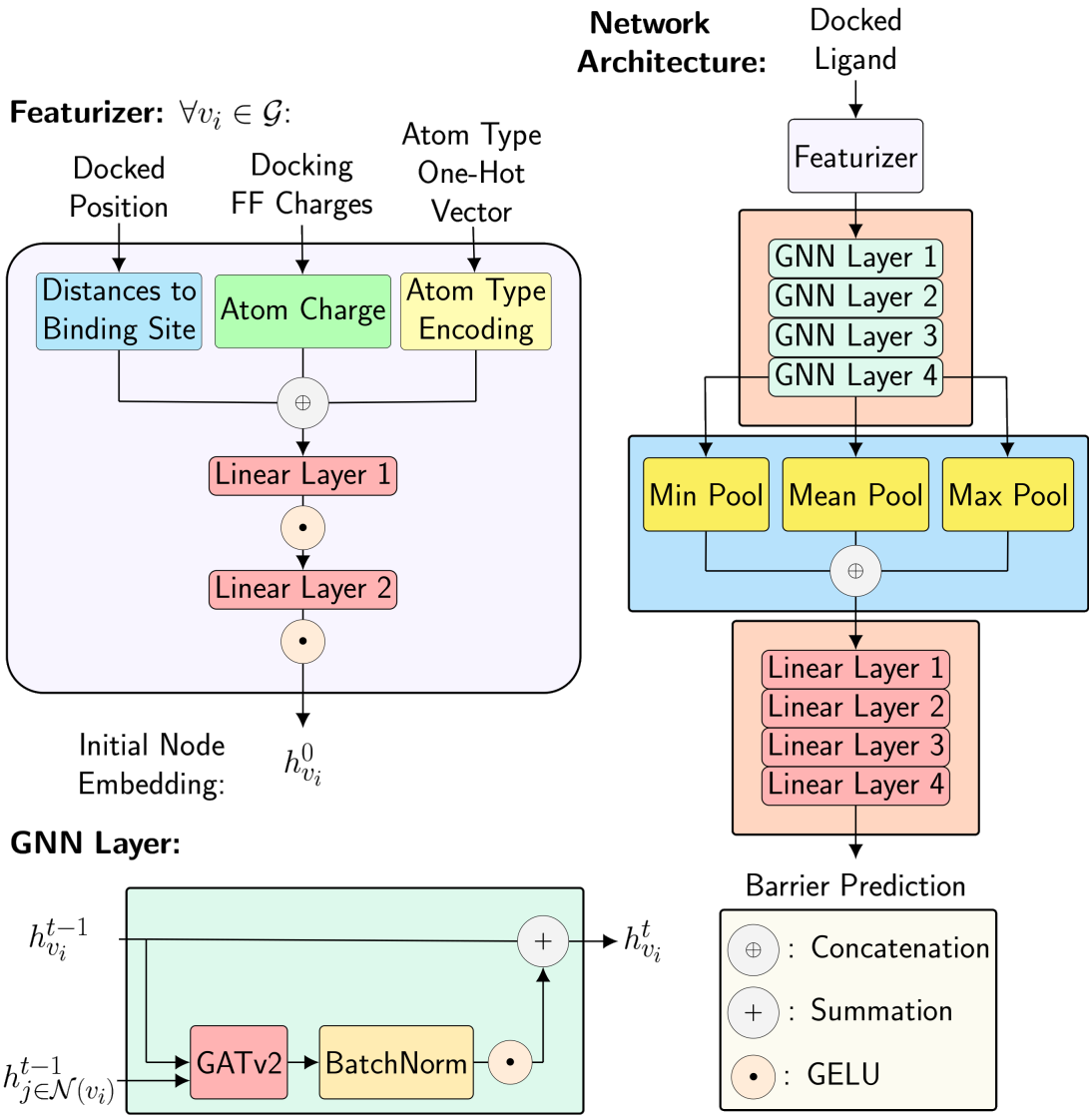


Figure 3.16: The GNN architecture used for RAS Barrier Prediction with GATv2 layers. This schematic shows an overview of the network used when using the internal distances feature set.

The next step in this study was to build a larger and more refined set of molecules for testing AL strategies. Similarity searching was performed against the 5 aforementioned known binders of RAS: adagrasib, sotorosib, ARS-853, MRTX1133, and BI-2852. Adagrasib and sotorosib were chosen as they are both approved drugs, therefore their scaffold must have both a high affinity and RAS selectivity. Despite being another covalent inhibitor ARS-853 had a distinct structure from the other compounds, indicated by the Tanimoto similarities shown in the appendix Fig. (3.27). Its inclusion provided molecules distinct from the

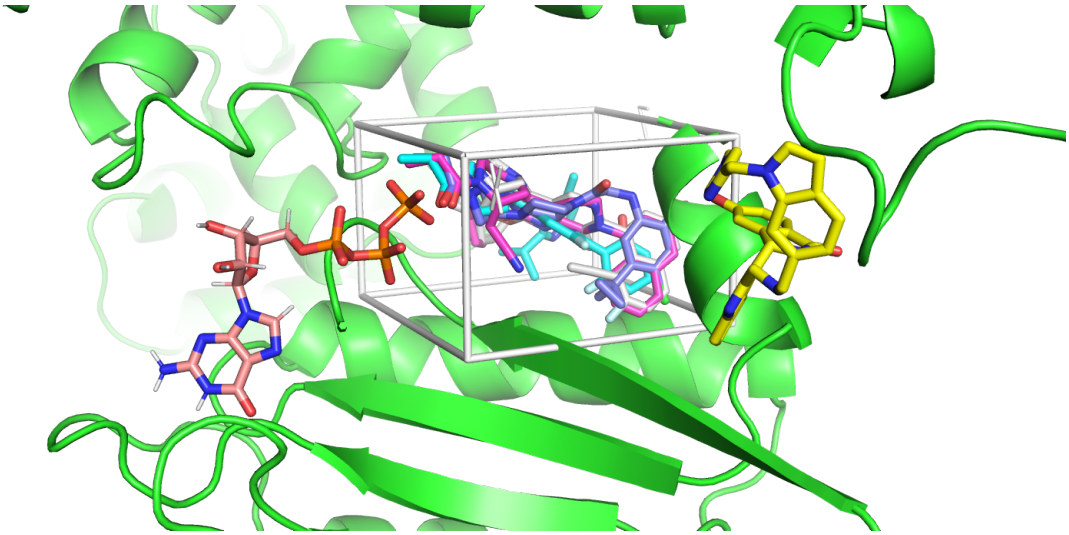


Figure 3.17: Visualisation of the known binders bound to RAS with GTP bound. Adagrasib (pink), sotorosib (cyan), ARS-853 (purple), and MRTX1133 (white) are viewed in the Switch II pocket. BI-2852 (yellow) can be seen further away from this, towards the Switch I loop in the Switch I/II pocket. The small docking box encompassing the Switch II pocket used by Vina is shown as a white rectangle.

relatively conserved structures seen across the other chosen binders. MRTX1133 and BI-2852 are both non-covalent inhibitors with experimentally observed affinity for G12D, and so they were possibly the most relevant molecules for our purposes. Experimentally bound structures for each of these molecules are shown in Fig. (3.17), and molecular diagrams are shown for each in appendix Fig. (3.28). Covalent inhibitors had their reactive groups removed prior to searching to avoid searching for this capability. The DDP outlined in Chapter (3.1) was used as the basis for the screening strategy, and the methods used are identical if not mentioned. Docking was only performed with AutoDock Vina to make a direct comparison with a single software. The box for Vina was centred upon the Switch II pocket, and taken to encompass all experimental binders excluding BI-2852. Ligands were put forward for QM/MM barrier calculations in a combination of the order of their docking scores.

To quantitively assess the improvement AL provided over docking, a testing framework was created to emulate a standard DDP implementation. Using a large set of pre-calculated QM/MM results, an initial set of N ligands was chosen by ranking ligands using the obtained docking score. This set was then used to train the outlined GNN for $10N$ epochs before performing inference over the

rest of the dataset. Querying then used a greedy selection criterion to choose an additional N ligands with the lowest average predicted barrier height across the 10 GNN ensemble. This process was repeated until the training set encompassed all available ligands. A full run of this methodology was performed for various values of N: 250, 500, 750, 1000, and 1500. All AL experiments used GATv2 GNN layers.

Methods were evaluated by their efficiency in finding the lowest 10% of the QM/MM calculated barriers. Comparisons were drawn between each of these and results from considering using the docking score or a control random selection instead. Inspired by Gentile et al., an enrichment value was also calculated to rank the models performance[713]. Rather than comparing to a random selection as in Eqn. (3.4.3.2), our strategy was to quantify the improvement gained over a pure docking-based strategy:

$$E_N = \frac{[\text{Proportion found via AL}]_N}{[\text{Proportion found via docking scores}]_N} \tag{3.4.4.1}$$

In addition to this experiment, a prior test between the various types of GNN layers was performed to assess their suitability for the task. This split the dataset using a 80/20 scaffold split using the same methodology as before. GATv2, GraphSAGE, GCN, and molecular fingerprint layers were subsequently trained using the same hyper-parameters as the AL strategy.

3.4.5 Results & Discussion

Moving to the preliminary experiments on the ZINC compound dataset, results for the baseline ChemProp model are shown in appendix Fig. (3.29). The hyper-parameter optimized model was able to obtain a PCC value of 0.445 alongside an average test set MSE of 4.825, and can be viewed in appendix Fig. (3.29). Featurization tests for our GNN model are shown in Fig. (3.18). Surprisingly by a wide margin the best performance was achieved by the Internal Distances feature set. This only slightly beat the optimised ChemProp model in terms of accuracy, with an average MSE of 4.80. However in terms of correlation it performed significantly better, achieving a PCC value of 0.643. For our usage, the PCC value was more significant as the accuracy of a prediction was relatively

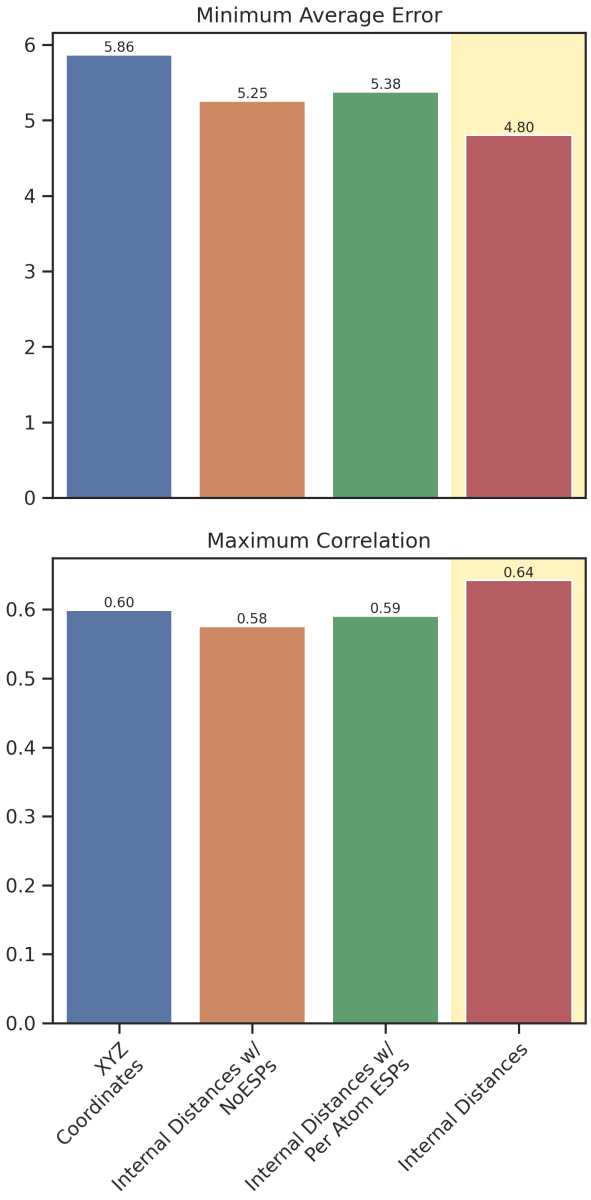


Figure 3.18: Average error and correlations for the different featurizations tested over the test set for the ZINC compounds. The measured values are taken from the best performing model in terms of correlation trained over the 10 parallel runs.

unimportant as long as they matched the order of QM/MM results. Therefore this was a promising result.

A scatter plot of the barrier height against internal distances predictions is presented in Fig. (3.19). The performance of this feature set is notable due to its simplicity when compared to the other internal distance representations. Both other approaches show a similar performance to using the XYZ coordinates

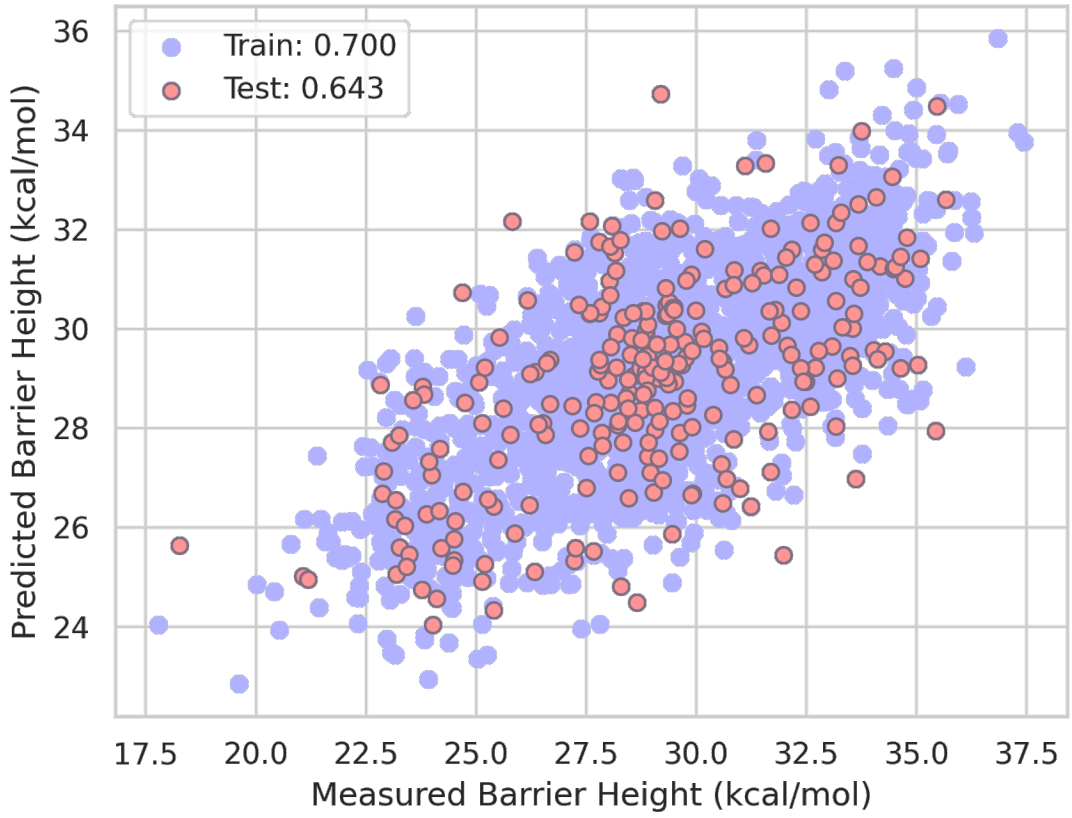


Figure 3.19: Scatter plot of the QM/MM calculated barrier against the best GNN model training using internal distances features. This model provided

in terms of correlation, even though they are supplemented with additional information about each atom. This highlights the importance of basic yet effective features, as the introduction of more values appears to negatively impact the network as it must learn to disregard irrelevant data. Despite the ChemProp model displaying a lower MSE than all GNN models but the internal distances, it was significantly outperformed in terms of correlation, highlighting the critical importance of structural information for effective predictions.

Further investigation was performed on the ZINC set to assess the reasons why including coordinates improved performance by such a factor. The centre of geometries for all ligands in this set are shown in the appendix Fig. (3.30). It is clear the large binding pocket used for the DOCK calculations split the data into two discrete parts around the Switch I and Switch II pockets. Few ligands can also be observed in the region between, near to where BI-2852 has been resolved in the Switch I/II pocket. This was most likely the reason why involving distances

had such a dramatic effect on the correlation as it was able to discriminate which pocket the ligand was bound to. Additionally early Vina docking results to a box matching the ZINC set showed a significant preference for the Switch I pocket, as shown in appendix Fig. (3.31). This region had no experimentally confirmed binders in the set used for similarity searching. Therefore the possibility of finding a ligand suitable for this region from similar ligands Switch II and Switch I/II binders was deemed unlikely.

Both of these prior observations led to choosing the small region around the Switch II pocket as the Vina binding box displayed in Fig. (3.17). 178,877 ligands were docked to this and their distribution is shown in Fig. (3.20). The docking scores for the best isomer LigPrep generated for each known binders are also shown in this figure, and show a wide variation between the different known inhibitors. Adagrasib stands as by far the best scoring, being in the top 0.5% of molecules overall. Additionally both BI-2852 and MRTX1133 score highly, showing promise that some of the characteristics of these molecules may be valued by Vina. However ARS-853 and sotorasib highest scores are distributed around the mean, and would not be chosen by a docking score selection strategy until very late in the process. This provides additional justification for the need for improved scoring ligands, as even an approved drug for this system is disregarded by docking scores.

10,656 QM/MM calculations were performed on the best scoring docked ligands and the distribution of the barriers can be seen in Fig. (3.21). The bottom 10% (1065 ligands) of this range up to a maximum barrier height of 25.34 kcal/mol, just above the 24.1 kcal/mol calculated for unbound G12D[723]. 521 ligands display a reduction of the hydrolysis barrier to below the G12D value, with two ligands lower than 18.8 kcal/mol. These values are comparable to the wild-type barrier of 18.5 kcal/mol. Further testing of these compounds could more accurately predict their inhibition capabilities, and provide a basis for future work.

Comparisons between the different GNN layers were consistent with Thompson et al. study, highlighting the relative insignificance of architectural choices like the layer type[714]. Appendix Fig. (3.32) and (3.33) show the MSE values and correlations over the test and training datasets for all 10 models of each layer type. It is difficult to discriminate between them in terms of performance, with

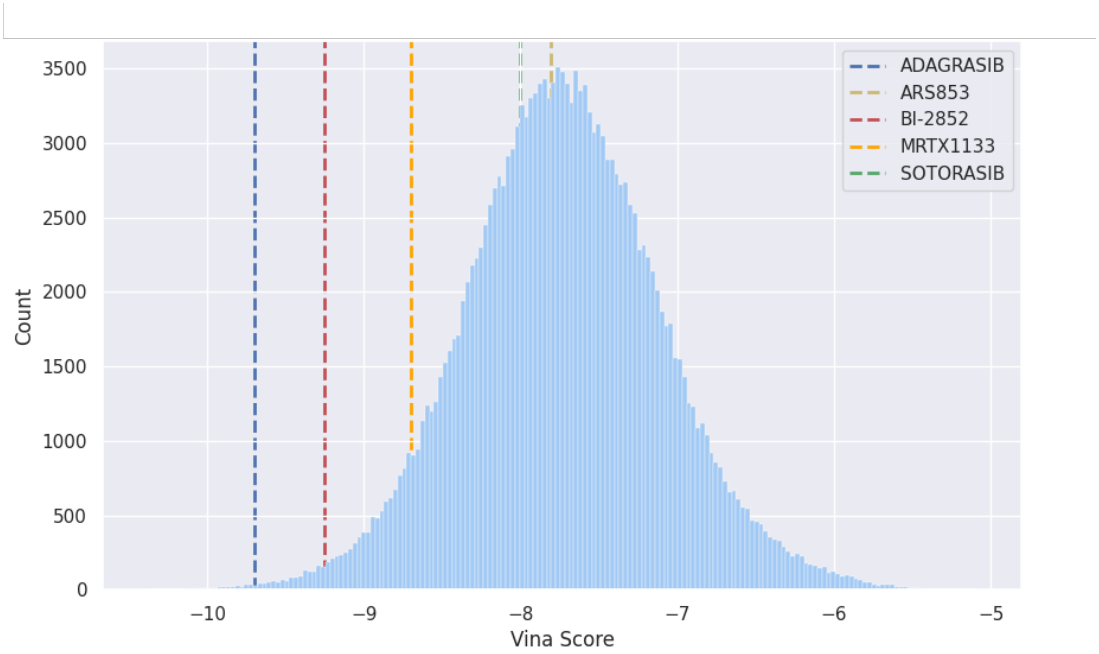


Figure 3.20: Vina docking score distribution across found similar ligands and known binders. For each known binder, the score shown is for the best scoring isomer generated by LigPrep.

all scoring very similarly across both metrics. In terms of correlation across the test set, every model was around ≈ 0.37, a significant departure from the results for the ZINC set. Counter-intuitively, this was associated with a very significant increase in prediction accuracy with nearly all models displaying near identical averaged MSEs across the test set. This was around a 3.6× increase from the accuracy observed on the ZINC compounds.

All AL experiments displayed a significant improvement over a random selection with over double a sampling rate of the ligands corresponding to the lowest QM/MM barriers in all cases. Enrichment values against docking scores are shown for all selection sizes tested in Fig. (3.22). Every experiment shows an improvement over docking, with a maximum enrichment consistently occurring when approximately 50% of the dataset has been sampled. The maximum ranges from 1.061 for $N = 1500$ to 1.10 for $N = 750$. While this is not as extreme an improvement as other results in the literature, this is a clear result for little additional computational cost when compared to the barrier calculations. Intuitively AL is consistently beaten by docking scores until around 20% of the dataset is sampled, due to the need for optimisation. This could potentially be circumvented

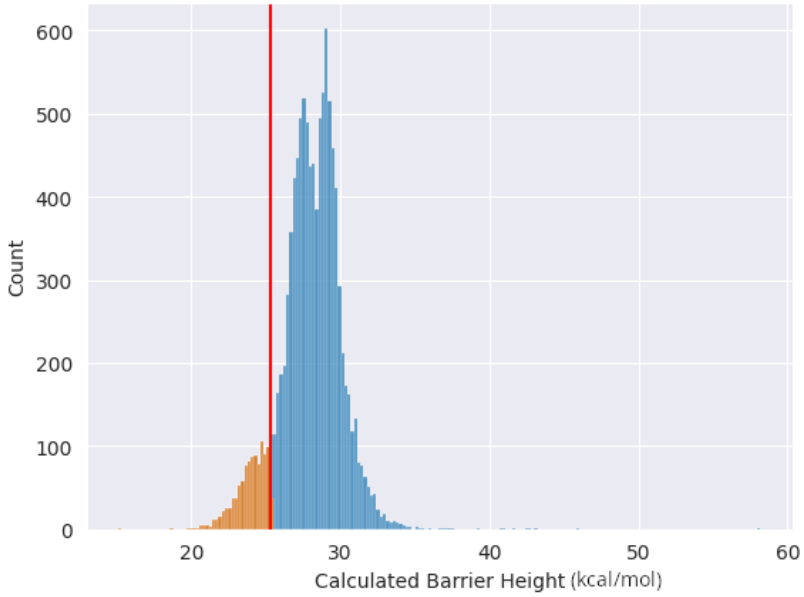


Figure 3.21: Histogram of QM/MM calculated barrier heights for the 10,656 tested ligands. The red line indicates the cut-off of 25.34kcal/mol and compounds with a barrier below this are shown in orange.

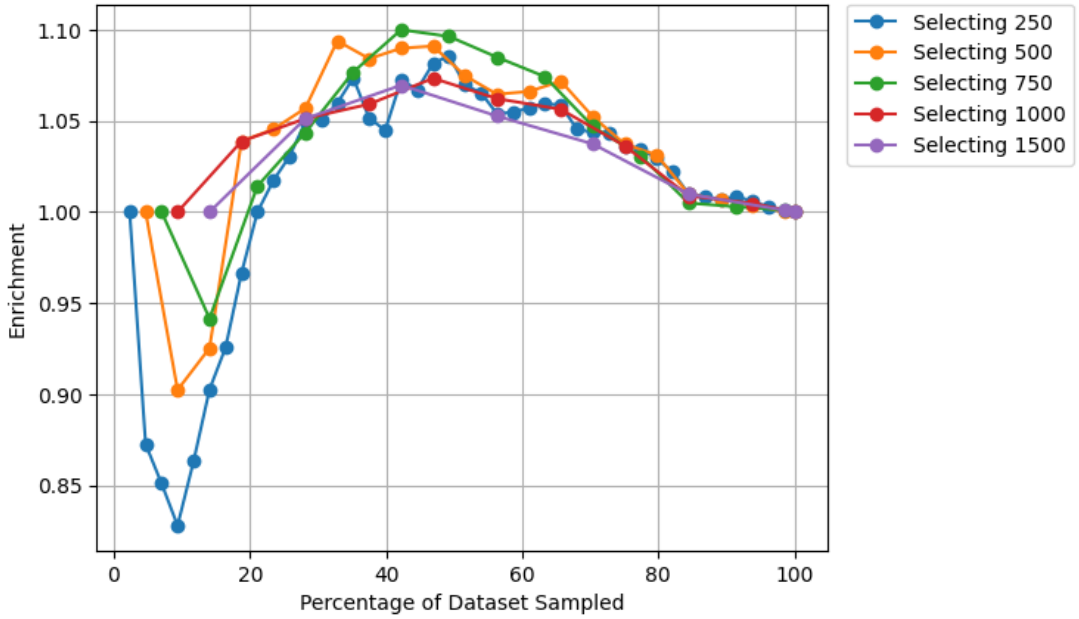


Figure 3.22: AL enrichment against percentage of the dataset sampled over a pure docking strategy calculated for all values of N tested.

for small selection sizes by using a larger initial set or more training epochs, providing an avenue for further work.

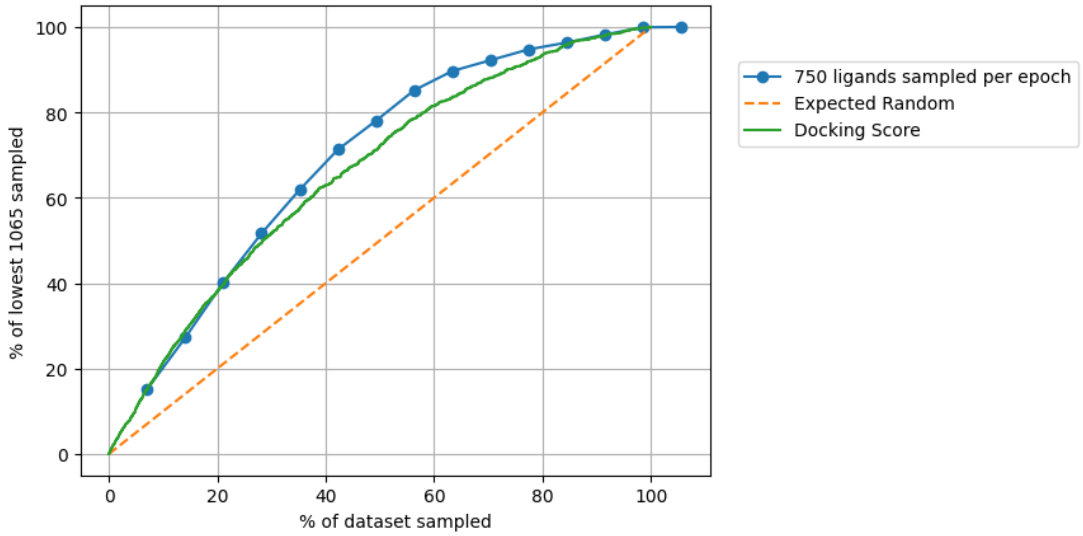


Figure 3.23: Selection efficiency comparison between the GNN with $N = 750$, docking score, and a random selection.

The best performing experiment was for $N = 750$, displaying a 10.0% improvement over the Vina score when 42% of the data is sampled. Figures showing the ligand selections for this case are shown in appendix Fig. (3.34), and a similar plot showing the selection distribution for docking scores is in appendix Fig. (3.35). Fig. (3.23) shows a direct comparison between this AL experiment, docking, and a random selection. The most unexpected result is the high performance of the docking score for this calculation, as it provides a significantly better approximation for this more complex calculation than for the QM/MMGBSA is Case Study 4. Further analysis is required to see if there is any reason why this may be. As it is surprising giving Vina displayed a high aversion to the used Switch II pocket in preliminary docking results, and the correlation between the docking score and barrier is only 0.21 almost half the GNNs.

3.4.6 Conclusion & Future Work

This work provided a definitive result that a noticeable improvement can be made to ligand selection efficiency by implementing a relatively simple AL strategy. Although the magnitude of the improvement was not as significant as other reported results in similar AL studies, a $\approx 10\%$ benefit can have a dramatic impact on computational resources at the considered scales. Obtaining barriers for 10,000 ligands was no simple feat, requiring a large amount of work by my colleagues which is not included in this Case Study. The expensive nature of gathering data makes AL a perfect strategy for this scenario. Training time is negligible compared to simulation times, and while a batch of 128 simulations can take around a day with the QM/MM method on a performant cluster, the whole training process for layer comparison studies only took 17 hours. In addition this training is inexpensive enough to be performed locally on GPU hardware. Running such a strategy in parallel to an existing DDP can therefore be done with very little additional overhead while providing a clear improvement. Over the next few years I believe techniques such as this will become a common part of approaches to drug discovery to enable a wider variety of calculation types to scale more effectively. Independent NNs can be trained at each step of a DDP to make a more effective usage of gathered data as it is generated.

The project highlights the significant amount of work required to provide even a small improvement to DDP techniques, and yet still there are many unanswered questions. The final enrichment improvement is limited by the size of the data, and I expect with a significantly larger dataset the benefit of AL would only increase. The convergence of docking scores and AL enrichment in the last half of the experiments can be associated with the lack of choice making the task significantly harder. However further investigation into this is prohibited by the cost of the data. I also think the usage of docking software is much more effective here, due to the restrictive ligand box. Vina is much more performative than what was observed in Case Study 4, and given the confusing results achieved when using a large box for RAS I think this is due to the sampling algorithm being able to optimise more effectively in a small region of space. Further analysis into

this could be used to better infer the performance of the AL and inform future docking strategies.

Another significant improvement to the work can be made with a hyper-parameter search, with little justification being provided for many of layer dimensionalities and training choices. This includes testing alternate selection strategies, although the literature does provide some justification for the greedy strategy used. While the ML system was inspired by GNN approaches across similar literature around this area, more testing is required to ensure the most optimal GNN architecture is used for this use case. However the choices for such an experiment are numerous and can require a lot of time, therefore the work had to focus on a small selection of parameters around the AL strategy and GNN layers. Additionally, the selection of ligands is biased towards those which scored low in the docking calculations, and therefore there is no clear quantitive assessment of how a GNN would predict across a whole VS library. The chemical diversity of this full space may provide significant issues for the data, however even this dataset contained a range of molecules. Alternatively performance may be improved by not relying on the docking score as a direct screening method, avoiding many of its issues. Finally, I think the difference in performance between the ZINC set and the similarity searched set requires some further study. The accuracy gain is most likely due to the smaller space available to ligands, but the reasoning for the significant reduction in correlation for reasons unclear. I was unable to come to a quantitive conclusion as to what causes this.

An unexplored aspect of this work is using GNN models to optimise the ligand positions. By predicting a complex calculation using an autodiff enabled scheme, it is also possible to take derivatives of the predicted score with respect to the atomic positions. Via an effective parameterisation, such as using rotatable bonds and quaternions like in docking software, this gradient could be used as a system specific optimiser for docking positions. This would be a true novel use of a GNN, and highlight how they are capable to provide a multifaceted benefit to a DDP.

Biologically, the work showed a viable strategy for finding lead compounds in the search for a pan-RAS inhibitor. Development of such BACs is of vital importance to improving our abilities to fight some of the most common cancers. Experimental confirmation of any binder found is required before its potential

can be assessed, but this work found The project forms the basis of an effective strategy for screening for G12D RAS, showing a clear extension to other RAS isoforms and mutations. I hope this work can be capitalised on to provide more compounds for experimentalists to test to this end.

3.4.7 Appendix

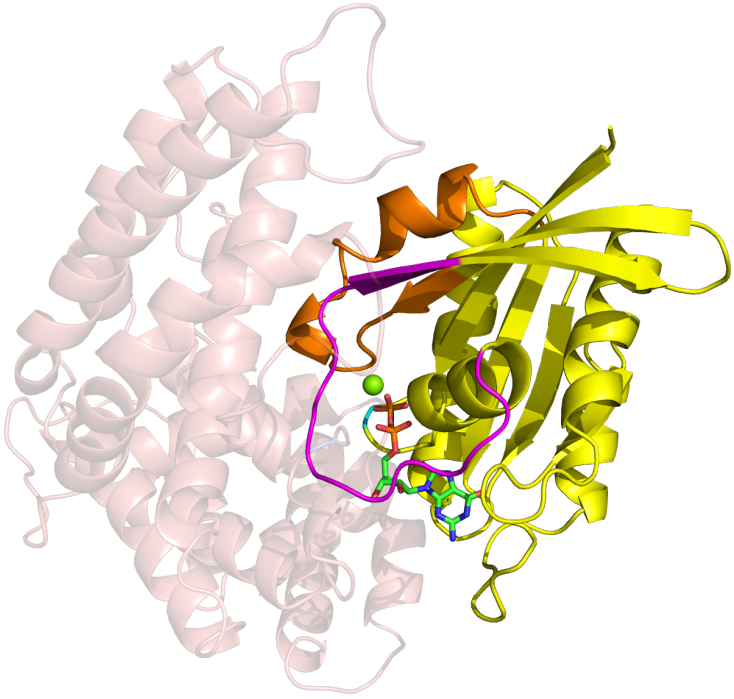


Figure 3.24: Location of the Switch I and Switch II pockets in the RAS structure (PDB ID: 1WQ1), coloured in magneta and orange respectively. A bound MG ion and GDP molecule are also show. The p120-RASGAP structure is transparent and in orange, showing the close proximity of the Switch I/II pockets to the GAP interface.

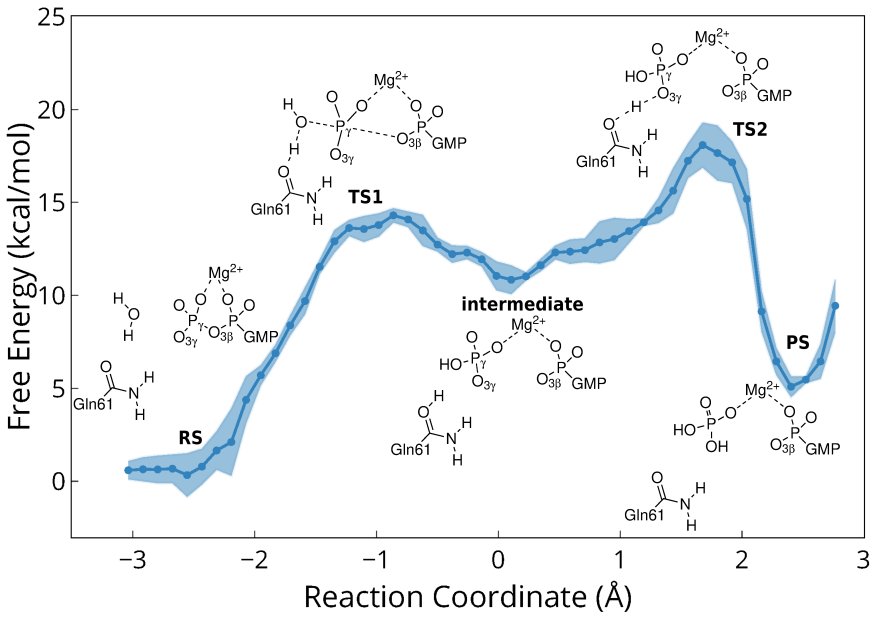


Figure 3.25: Free energy barrier calculated using the finite temperature String method in previous work by my group. The blue shaded areas represent the variation of the free-energy estimation. Figure courtesy of my colleague Dénes Berta.

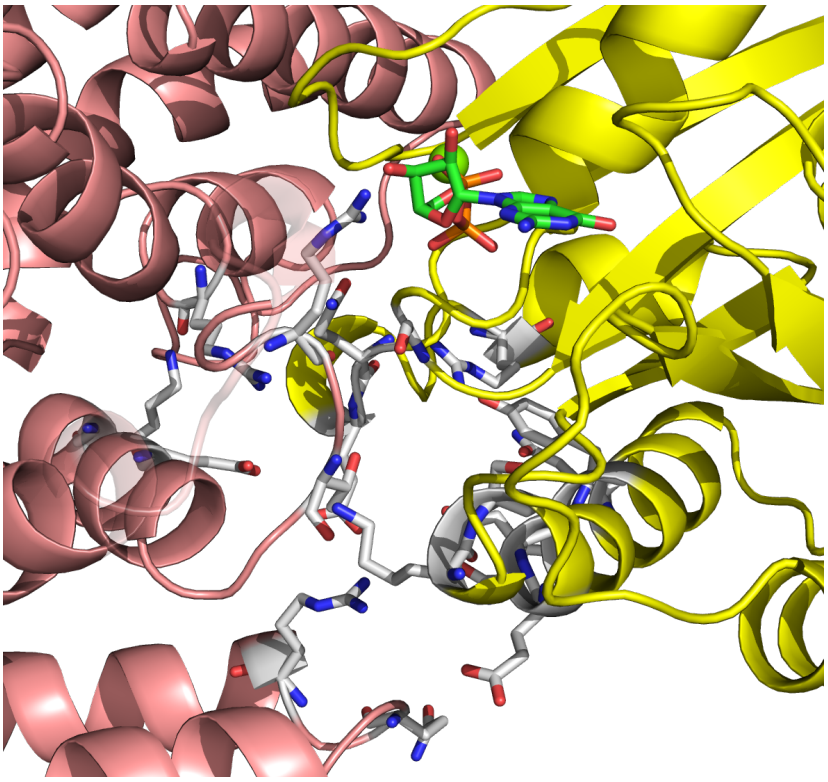


Figure 3.26: Overview of the residues selected as features to represent the binding pocket. Chosen residues are from RAS: G12, Q61, R68, K88, S89, E91, D92, Q95, Y96, Q99 and GAP: Q747, R749, R789, T791, E799, K803, R903.

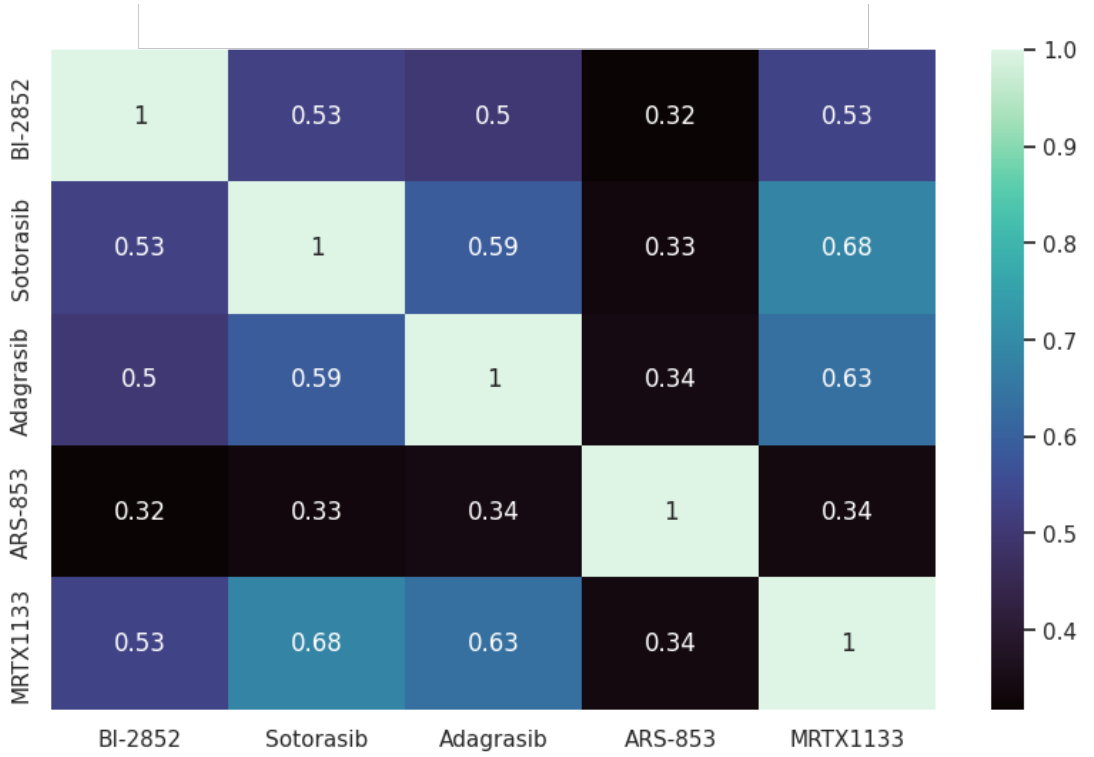


Figure 3.27: Tanimoto similarity calculated between known RAS inhibitors using ECFP4 fingerprints. ARS-853 is clearly very distinct from all other compounds, which all show $a > 0.5$ similarity.

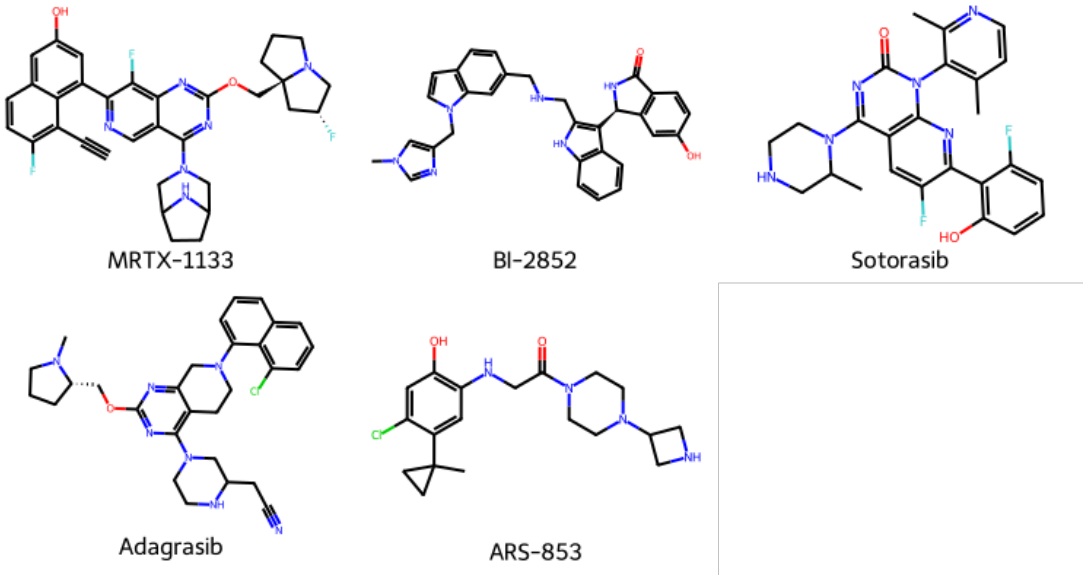


Figure 3.28: Molecular diagrams for the 5 RAS binders used for similarity searching.

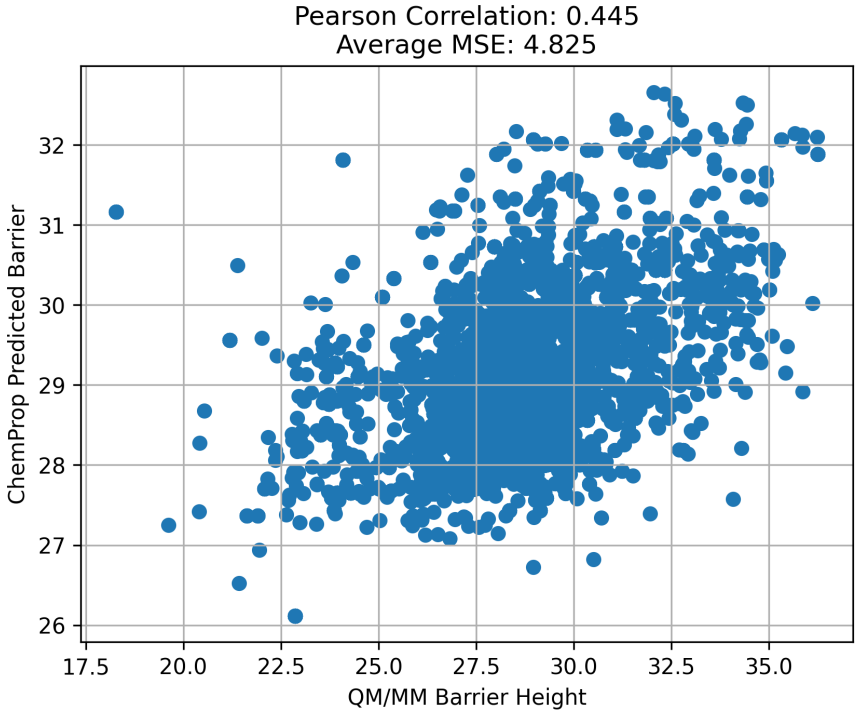


Figure 3.29: Performance of the ChemProp baseline model on the ZINC dataset. Displayed MSE and PCC values are calculated across the test set only.

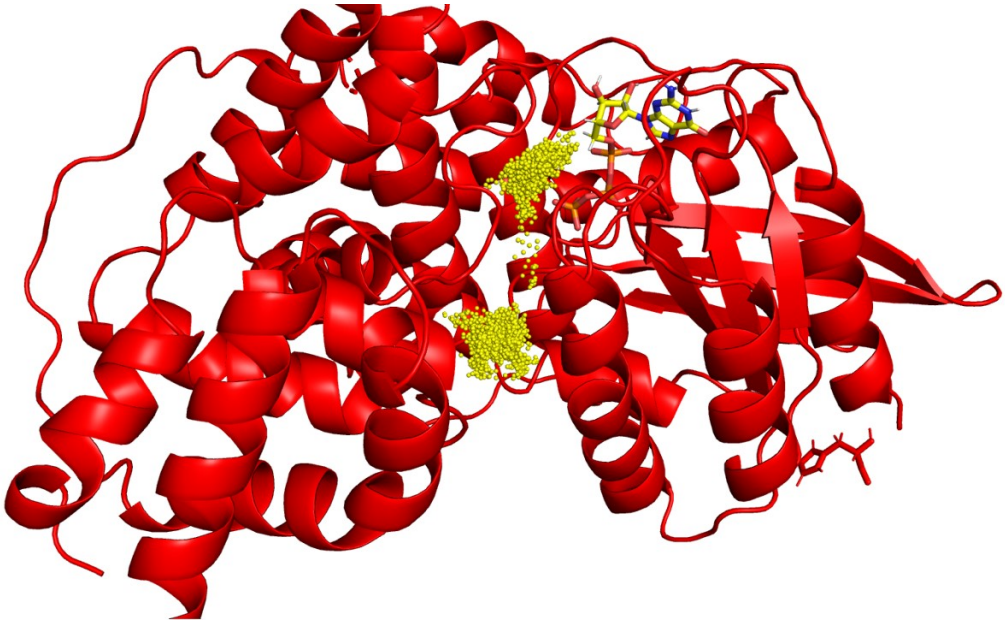


Figure 3.30: Centre of geometries for all ligands in the ZINC dataset.

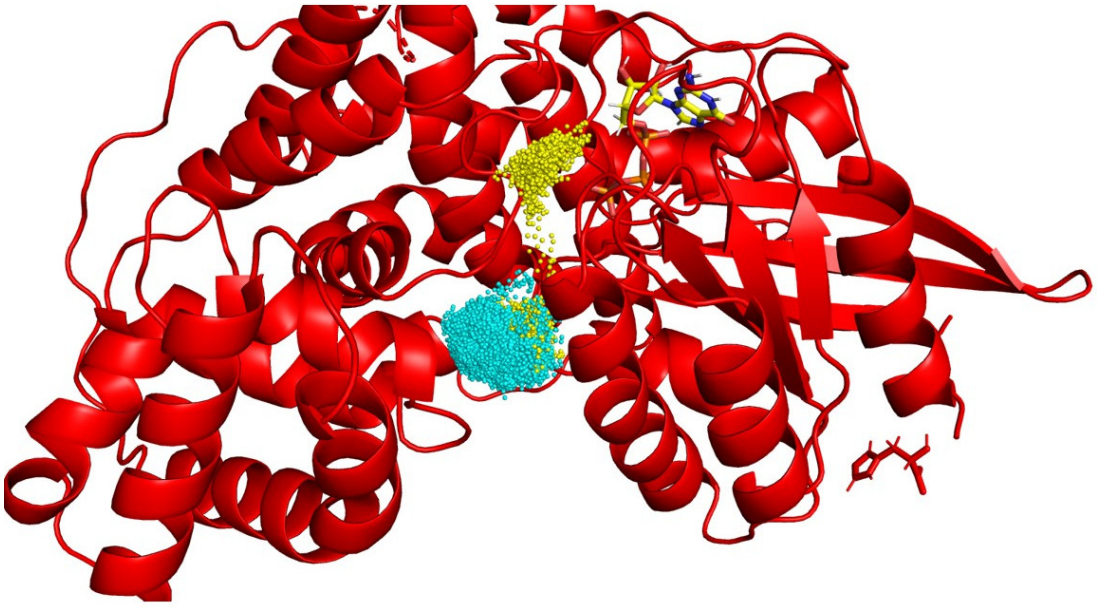


Figure 3.31: Centre of geometries for all ligands in the ZINC dataset alongside 16,257 preliminary Vina docking results to a binding box matching the ZINC set. Vina clearly display a large preference for the Switch I pocket, with no compunds being docked to the Switch II pocket.

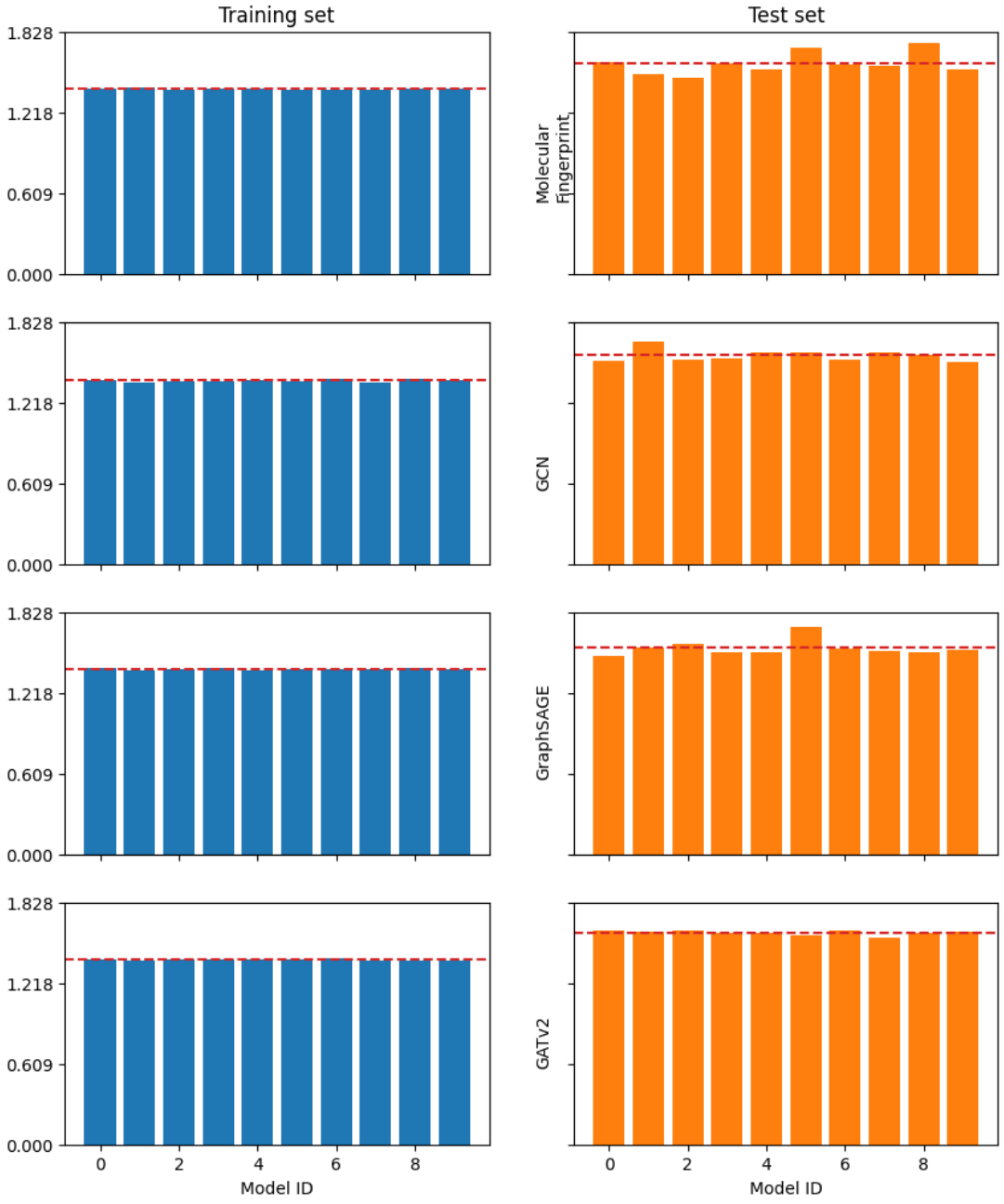


Figure 3.32: The averge MSE errors for all 10 GNN layer types across the training and test sets.

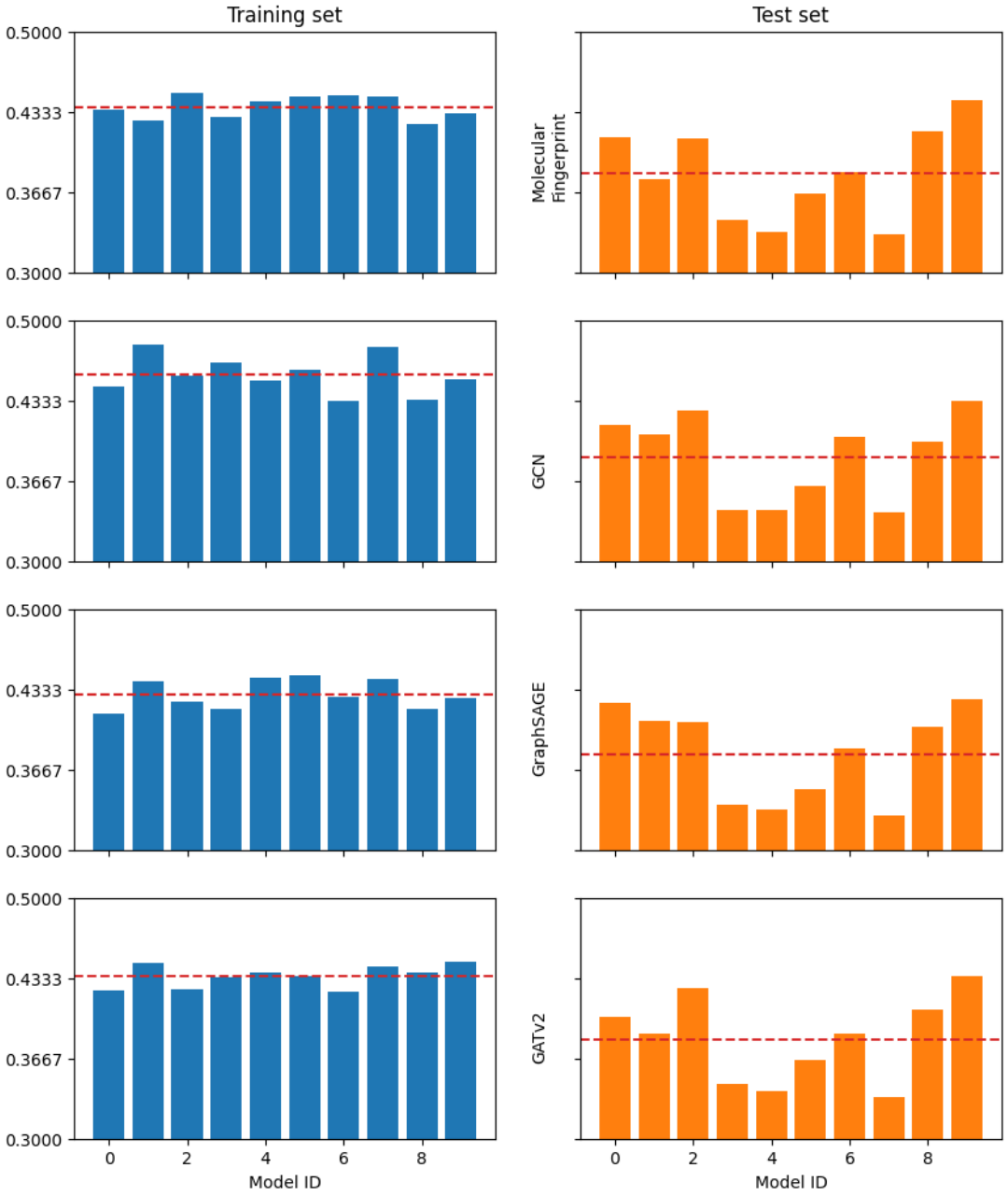


Figure 3.33: The PCC values between predicted and calculated barrier heights for all 10 GNN layer types across the training and test sets.

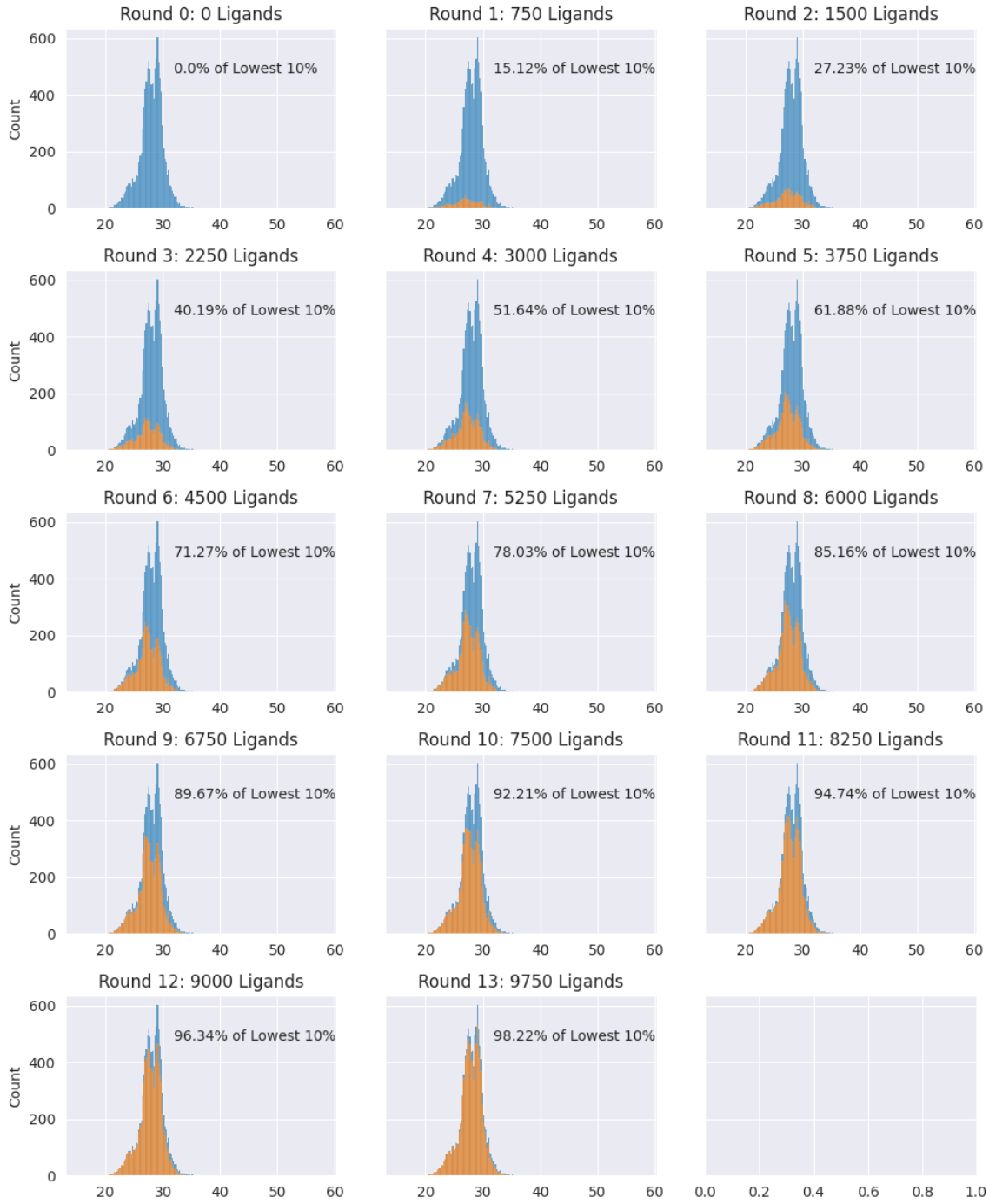


Figure 3.34: Histogram plots showing the ligand selections for the GNN model using a selection size of $N = 750$ and GATv2 layers. All barriers are shown in blue and those selected by this strategy are shown in orange.

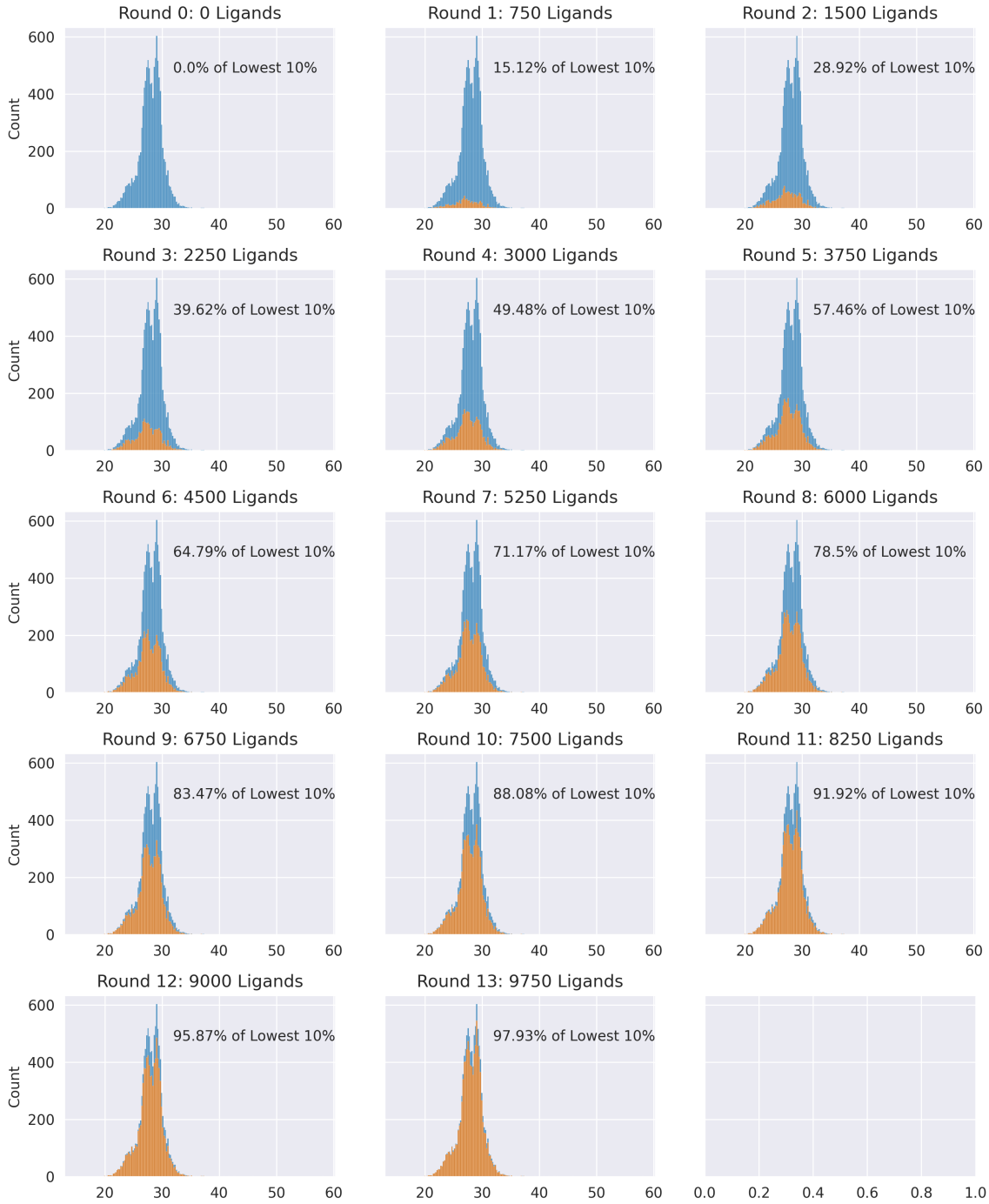


Figure 3.35: Histogram plots showing the ligand selections using docking scores to choose compounds and a selection size of $N = 750$. All barriers are shown in blue and those selected by this strategy are shown in orange.

Chapter 4

Conclusion

As a relative newcomer to this field when I began my PhD, I have learnt a significant amount about current research surrounding drug design over the course of my study. This thesis aimed to provide a comprehensive overview of this alongside technical details of the studies performed outlining my contribution to the field. Chapter 1 looked to provide the context for this, and describe my personal perspective on the field, its many underlying theories, and the reasons for its current metamorphosis. Throughout my time it has evolved rapidly, with an explosion of ML techniques, the number of identified therapeutic targets, and the availability of both data and powerful predictive models. I believe this is a direct consequence of the proliferation of graphics hardware finally being applied to process huge scientific datasets over the past 20 years, which became the unifying thread between my projects. While GPUs themselves may get succeeded by more powerful and specialised heterogenous computing architectures in the recent future, the focus on large scale parallelism is not going to change anytime soon. For this reason they provided the initial basis for the work, and their impact on both the subsequently described fields of ML and MD are clear. Much of the subsequent work focused on the development or application of methods building from this basis, forming a large portion of the scientific contribution of this work.

Both of the other fields introduced in Chapter 1 formed the computational backbone of this work, with a NN featured in 3 of the Case Studies and some type of FF-based simulation data being utilised or generated in all projects of this work. The set of techniques comprised modern MD are ingrained within this

area, and their successes are numerous. As the main numerical simulation method for biological systems its greatest strength is its wide applicability, forming the backbone of so many studies into protein behaviour and function. This includes this work, where it provided key insights into structural and functional changes in Case Studies 1 and 2 respectively, and underpinned the quantitive assessment of binding affinity in Case Study 4. In addition, the closely related FFs were utilised in Case Study 5 as part of a related yet distinct simulation method capable of improving the performance of QM level simulations. This shows how there will always be a need for numerical simulations in biological study and these methods will continue to be iteratively refined. Improvements in performance and FFs will hopefully extend both its capabilities for larger systems and the class of compounds it can accurately simulate.

Over the next decade, I only expect the influence of ML algorithms in this field to get stronger. The magnitude of data available in biology is immense, and often so is the level of specialism required to practically use it, making these experts' time is possibly the greatest resource in the area. If ML models are able to approximate the proficiency biological experts have in binding verification, pharmaceutical and catalytic chemistry, and kinetic analysis, the number of systems capable to be assessed will dramatically increase. Already the impact of protein structure prediction networks is being felt, as they have been demonstrated to surpass many world-class experts at homology modelling. Despite these being hailed as the final answer to one of the great remaining scientific problems, they are an empirical and automated solution to a process that has been performed by experts in the field for years. For drug discovery, their availability helps with a major issue faced, and expands the scope of these techniques to genome scales. This vastly increases the potential number of available structures for druggable targets, and allows preliminary work to begin without requiring experiments.

These therapeutic targets still require an effective identification and investigation process, using techniques such as those highlighted in Chapter 2. Both alignment and dimensionality reduction play a key role in this process, and Case Study 1 showcases a potential strategy for combining both these techniques using VAEs. Latent space models show great promise for improving dimensionality reduction tasks on complex data such as protein dynamics. The project provides

first-hand examples of the problems these methods need to overcome, pertaining to effective comparisons of different protein structures and mappings between the learnt and physical spaces. However once these are resolved to some degree, they can be used to build representations with awareness of structural diversity confirmed in experiments. For established targets with many available crystal structures in varying degrees of activation or inhibition, this could provide an automated way to better understand the critical changes relating to these processes. Forming accurate yet small representations of the conformational space available to proteins is of great importance to developing realistic models of their behaviour, as emphasised in the next project.

Here a more effective parameterisation of the translocation process could have circumvented many of the issues arising in the final part of Case Study 2. Trying to build CVs from contact analysis of unbiased simulations proved ineffective, possibly due to insufficient sampling the conformational space relevant to the ATP hydrolysis driving the process. As it stands this work exemplifies a common approach to the early stages of drug design, consisting of reviewing literature, analysing basic MD, and attempting to simulate biological functions. The literature review to begin provides a comprehensive overview of the current theories on the translocation process across helicases. The techniques used to elucidate useful information offers examples of the strategies researchers can use to analyse MD. The great difficulty observed in simulated translocation is indicative of what often arises when further applying this information to observe functional processes. Biological phenomena are very sensitive and truly push the fidelity of numerical simulations to their limit. Despite this, the initial work provides a clear example of much of the early stage involved in assessing targets, through analysing MD simulations for critical contacts, correlations to measured data, and correspondence with theory. It culminated in two published papers providing data, analysis, and a review of the relevant theory to aid other researchers target this system.

The coordinate free methodology developed in Case Study 3 instead looks to circumvent many of these issues by using a state-based view of dynamical processes. While using MCs for this process is a well established techniques, there are several ways to construct such models from situation data. I believe it

provides a relatively novel usage of GNNs and hope the published method can provide some benefit to researchers faced with interpreting or condensing large-scale dynamical biological data. Using a state-based view allows for kinetically important conformations to be better identified, and the used loss implies the clusterings provided by the model have a greater dynamical relevance. Extracted states can be used to provide selected structures for downstream processes like functional simulations in Case Study 2 or docking in Case Study 4. Additionally I hope this method can inspire further applications of GNNs as effective Ansatz for difficult optimization problems on graphs.

Moving onto the next stage of the drug design process requires assessing introducing small molecules to protein structures. The methods section of Chapter 3 provides the necessary information to work with ligands in a computational capacity. It encompasses numerous approaches to working with small molecules effectively, and attaining a relatively well-rounded understanding of them was a substantial task. This was one of the major learning experiences of this work, hence its the comprehensive nature of the associated literature review. Culminating in strategies to evaluate molecules capacity as BACs by building a DDP with an informed selection of techniques, as observed in Case Study 4. As a direct application of some of these strategies, this Case Study serves as an example of a strategy capable to find an experimentally confirmed molecule capable of binding to NSP13. It highlights both the interdisciplinary nature of the work and the need for expert collaboration in the process. The ability of the network to process billions of ligands was a great computational result, and this contributes a practical starting point for DDPs to apply across biological targets.

Researchers in the field are always looking for ways to improve DDPs, and Case Study 5 shows a basic extension to this able to noticeably improve selection performance. The expensive nature of gathering data in these tasks provides the perfect environment for developing and testing AL algorithms. However there are few systematic studies looking to evaluate the benefit of including these strategies, and hence this became the focus of this project. Inclusion of this strategy displayed a clear 10% benefit when compared to using docking scores as compared, and offers a method that can be used as part of a DDP immediately. This method

also provides several avenues for future development, and I believe serves as a great example of the ML-enabled future of drug discovery.

From a biological perspective, this work contributes to the drug development process for 3 separate protein targets involved in many critical and common illnesses. COVID-19 is causative agent of the most widespread modern pandemic. The RTC forms its replicative complex, vital to proliferation of the disease and hence a promising therapeutic target. For one of the comprising proteins, helicase NSP13, a comprehensive overview of a possible drug design process is presented across Case Studies 2 and 4. This includes the presentation of a potential strategy for functional simulations alongside evaluating compounds for binding affinity. From initial target evaluation to developing and confirming experimental binders, this provides a textbook example of computational drug discovery. Both of these are a functional basis for two independent directions to take this research in the future. Translocation simulations could provide a way to build an effective and specialised DDP as presented in Case Studies 4 and 5. The discovered compounds could also be used for further LO to establish an even more selective compound targeting NSP13 In the future, both of these approaches could aid the development of novel drugs to fight COVID-19 by attacking a novel target. Providing a contribution to the emerging field of helicase inhibition, and avoiding many of the issues with existing treatments.

Mutations in the MAPK/ERK signalling cascade are key oncogenic drivers across many cancers. This pathway controls the rate of mitosis within the cell, and faulty activation can result in carcinogenesis. Case Studies 1 and 5 target BRAF and RAS, two consecutive proteins in the early stages of this cascade with both proven to contain some of the most prevalent mutations. While BRAF has an established treatment for its most prevalent oncogene, V600E, there are still many characteristics of the system, such as paradoxical activation and monomeric activation, that are poorly understood. The work presented on this system highlights an ML enabled strategy for better understanding the conformational space of one of its key motifs regulating activation. Many of the issues facing such an approach are discussed, and the final result shows an effective separation of functionally distinct crystal models. Improving these models to sample realistic conformations

would provide an automated system for better understanding proteins with a large number of related structures.

Several RAS mutations continue to evade effective treatment, and Case Study 5 presents a screening strategy specialised to one of these -the G12D mutation. When combined with the previous work in the group, this enables a full lead compound search leveraging data from established G12C covalent binders and some other known binders. However treatment is much more difficult without the mutated cysteine target. Discovery of a pan-RAS inhibitor, returning the hydrolysis barrier to wild-type levels, would provide a novel treatment for over 30% of kinase-caused cancers. While this work provided conclusive evidence that pipelines can be improved by including AL, for drug discovery purposes it was never run with the goal of discovering these compounds. Running the methodology this way could provide even more compounds showing the potential to lower the mutated hydrolysis barrier. In the future, the proposed strategy can be easily generalised to other mutations, and become part of a larger DDP targeting finding leads for such a compound.

Overall, this work has attempted to explain the context underlying my work and describe the contributions I have made to the field. While some of these projects may not have provided the ambitious solutions originally envisioned, I believe every one included presets a valuable perspective on a computational issue encountered during drug discovery. It also provides a working basis for many future drug design tasks, alongside detailed explanations of many of the related techniques and literature. The process of understanding biological process and modulating them is exceedingly complex, and this stands as a testament to the small benefits we can all make towards improving the employed strategies. As a community over time, we can only hope that the sum of our individual efforts can contribute to real-world outcomes on patient welfare somewhere down the line. I would like to thank the reader for making it this far, and I look forward to seeing how this field develops and many of these issues are resolved.

Bibliography

(*1*) Cowles, H. M., *The Scientific Method: An Evolution of Thinking from Darwin to Dewey*; Harvard University Press: 2020; 385 pp.

(*2*) Gauch, H. G., *Scientific Method in Practice*; Cambridge University Press: 2003; 458 pp.

(*3*) Gower, B., *Scientific Method: A Historical and Philosophical Introduction*; Routledge: London, 2012; 288 pp.

(*4*) Nola, R., and Sankey, H., *Theories of Scientific Method: an Introduction*; Routledge: London, 2014; 240 pp.

(*5*) Michelson, A. A., and Morley, E. W. (1887). On the relative motion of the Earth and the luminiferous ether. *American journal of science 3*, 333–345.

(*6*) Halliday, S. (2001). Death and miasma in Victorian London: an obstinate belief. *BMJ 323*, 1469–1471.

(*7*) Natarajan, V. (2008). What Einstein meant when he said "God does not play dice ...". *Resonance 13*, 655–661.

(*8*) BF, G., *Digital computers in research - an introduction for behavioral and social scientists.* Mcgraw-Hill: New York, 1963.

(*9*) Foster, J. G., Rzhetsky, A., and Evans, J. A. (2015). Tradition and innovation in scientists' research strategies. *American sociological review 80*, 875–908.

(*10*) Gandomi, A., and Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management 35*, 137–144.

(*11*) Davenport, T. H., and Patil, D. (2012). Data scientist. *Harvard business review 90*, 70:76.

(*12*) Google Open Dataset Search `https://datasetsearch.research.google.com/` (accessed Dec. 8, 2024).

(*13*) Find Open Datasets and Machine Learning Projects | Kaggle `https://www.kaggle.com/datasets` (accessed Dec. 8, 2024).

(*14*) awesomedata/awesome-public-datasets `https://github.com/awesomedata/awesome-public-datasets` (accessed Dec. 8, 2024).

(*15*) Gao, L. et al. (2020). The Pile: An 800GB Dataset of Diverse Text for Language Modeling. DOI: `10.48550/arXiv.2101.00027`.

(*16*) The CERN Data Centre, CERN `https://home.cern/science/computing/data-centre` (accessed Dec. 8, 2024).

(*17*) Desai, K., Kaul, G., Aysola, Z., and Johnson, J. (2021). RedCaps: web-curated image-text data created by the people, for the people. DOI: `10.48550/arXiv.2111.11431`.

(*18*) Fang, H., Beaudoing, H. K., Rodell, m., Teng, W. L., and Vollmer, B. E. Global Land Data Assimilation System (GLDAS) Products, Services and Application from NASA Hydrology Data and Information Services Center (HDISC) `https://ntrs.nasa.gov/citations/20090005038` (accessed Dec. 8, 2024).

(*19*) McPherson, A., and Gavira, J. A. (2013). Introduction to protein crystallization. *Acta Crystallographica. Section F, Structural Biology Communications 70*, 2–20.

(*20*) Wlodawer, A., Minor, W., Dauter, Z., and Jaskolski, M. (2013). Protein crystallography for aspiring crystallographers or how to avoid pitfalls and traps in macromolecular structure determination. *The FEBS journal 280*, 5705–5736.

(*21*) MacDougall, A. et al. (2020). UniRule: a unified rule resource for automatic annotation in the UniProt Knowledgebase. *Bioinformatics 36*, 4643–4648.

(*22*) Valencia, A. (2005). Automatic annotation of protein function. *Current Opinion in Structural Biology 15*, 267–274.

(*23*) Rost, B., Liu, J., Nair, R., Wrzeszczynski, K. O., and Ofran, Y. (2003). Automatic prediction of protein function. *Cellular and molecular life sciences: CMLS 60*, 2637–2650.

(*24*) Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., and Bourne, P. E. (2000). The Protein Data Bank. *Nucleic Acids Research 28*, 235–242.

(*25*) The UniProt Consortium (2017). UniProt: the universal protein knowledgebase. *Nucleic Acids Research 45*, D158–D169.

(*26*) UniProtKB search `https://www.uniprot.org/uniprotkb?query=*` (accessed Dec. 8, 2024).

(*27*) Amaro, R. et al. (2024). The need to implement FAIR principles in biomolecular simulations. DOI: `10.48550/arXiv.2407.16584`.

(*28*) Gaulton, A. et al. (2016). The ChEMBL database in 2017. *Nucleic Acids Research 45*, D945–D954.

(*29*) Sterling, T., and Irwin, J. J. (2015). ZINC 15 – Ligand Discovery for Everyone. *Journal of Chemical Information and Modeling 55*, 2324–2337.

(*30*) Huggins, D. J., Sherman, W., and Tidor, B. (2012). Rational Approaches to Improving Selectivity in Drug Design. *Journal of Medicinal Chemistry 55*, 1424–1444.

(*31*) Adam, M. (2005). Integrating research and development: the emergence of rational drug design in the pharmaceutical industry. *Studies in History and Philosophy of Science Part C: Studies in History and Philosophy of Biological and Biomedical Sciences 36*, 513–537.

(*32*) Rognan, D. (2007). Chemogenomic approaches to rational drug design. *British journal of pharmacology 152*, 38–52.

(*33*) Noé, F., De Fabritiis, G., and Clementi, C. (2020). Machine learning for protein folding and dynamics. *Current Opinion in Structural Biology 60*, 77–84.

(*34*) Zitnik, M., Nguyen, F., Wang, B., Leskovec, J., Goldenberg, A., and Hoffman, M. M. (2019). Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities. *Information Fusion 50*, 71–91.

(*35*) Tarca, A. L., Carey, V. J., Chen, X.-w., Romero, R., and Drăghici, S. (2007). Machine learning and its applications to biology. *PLoS computational biology 3*, e116.

(*36*) Jumper, J. et al. (2021). Highly accurate protein structure prediction with alphafold. *Nature 596*, 583–589.

(*37*) Yang, J., Anishchenko, I., Park, H., Peng, Z., Ovchinnikov, S., and Baker, D. (2020). Improved protein structure prediction using predicted interresidue orientations. *Proceedings of the National Academy of Sciences 117*, 1496–1503.

(*38*) Fradkov, A. L. (2020). Early History of Machine Learning. *IFAC-PapersOnLine 53*, 1385–1390.

(*39*) Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks 61*, 85–117.

(*40*) Schmidhuber, J. (2022). Annotated History of Modern AI and Deep Learning. DOI: 10.48550/arXiv.2212.11279.

(*41*) Child, J. M. (1917). THE MANUSCRIPTS OF LEIBNIZ ON HIS DISCOVERY OF THE DIFFERENTIAL CALCULUS. PART II. *The Monist 27*, 411–454.

(*42*) Banks, D. L., and Fienberg, S. E. In *Encyclopedia of Physical Science and Technology (Third Edition)*, Meyers, R. A., Ed.; Academic Press: New York, 2003, pp 247–261.

(*43*) Agar, J. (2020). What is science for? The Lighthill report on artificial intelligence reinterpreted. *The British Journal for the History of Science 53*, 289–310.

(*44*) Benest, I. D. (1979). A review of computer graphics publications. *Computers & Graphics 4*, 95–136.

(*45*) Foley, J. D., *Introduction to Computer Graphics*; Addison-Wesley: 1994; 616 pp.

(*46*) Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., and Phillips, J. C. (2008). GPU Computing. *Proceedings of the IEEE 96*, 879–899.

(*47*) Nickolls, J., and Dally, W. J. (2010). The GPU Computing Era. *IEEE Micro 30*, 56–69.

(*48*) Deng, Y., Ni, Y., Li, Z., Mu, S., and Zhang, W. (2017). Toward Real-Time Ray Tracing: A Survey on Hardware Acceleration and Microarchitecture Techniques. *ACM Comput. Surv. 50*, 58:1–58:41.

(*49*) Wald, I., Mark, W. R., Günther, J., Boulos, S., Ize, T., Hunt, W., Parker, S. G., and Shirley, P. (2009). State of the Art in Ray Tracing Animated Scenes. *Computer Graphics Forum 28*, 1691–1722.

(*50*) Nickolls, J., Buck, I., Garland, M., and Skadron, K. In *ACM SIGGRAPH 2008 classes*, Association for Computing Machinery: New York, NY, USA, 2008, pp 1–14.

(*51*) Spencer, G. H., and Murty, M. V. R. K. (1962). General Ray-Tracing Procedure. *JOSA 52*, 672–678.

(*52*) Glassner, A. S., *An Introduction to Ray Tracing*; Morgan Kaufmann: 1989; 364 pp.

(*53*) Peter Shirley, S. H., Trevor David Black Ray Tracing in One Weekend `https://raytracing.github.io/books/RayTracingInOneWeekend.html`.

(*54*) Landau, H. (1967). Sampling, data transmission, and the Nyquist rate. *Proceedings of the IEEE 55*, 1701–1706.

(*55*) Vaidyanathan, P. (2001). Generalizations of the sampling theorem: Seven decades after Nyquist. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications 48*, 1094–1109.

(*56*) Mitchell, D. P. (1990). The antialiasing problem in ray tracing. *Advanced Topics in Ray Tracing, SIGGRAPH 1990 Course Notes*.

(*57*) Kwan, A., Dudley, J., and Lantz, E. (2002). Who really discovered Snell's law? *Physics World 15*, 64.

(*58*) Wald, I., Purcell, T. J., Schmittler, J., Benthin, C., and Slusallek, P. In *Eurographics (State of the Art Reports)*, 2003, pp 85–122.

(*59*) Singh, J. M., and Narayanan, P. J. (2010). Real-Time Ray Tracing of Implicit Surfaces on the GPU. *IEEE Transactions on Visualization and Computer Graphics 16*, 261–272.

(*60*) Santos, A. L. D., Lemos, D., Lindoso, J. E. F., and Teichrieb, V. In *2012 14th Symposium on Virtual and Augmented Reality*, 2012 14th Symposium on Virtual and Augmented Reality, 2012, pp 131–140.

(*61*) Ludvigsen, H., and Elster, A. C. In *Eurographics (Short Papers)*, 2010, pp 65–68.

(*62*) Li, C., Ding, C., and Shen, K. In *Proceedings of the 2007 workshop on Experimental computer science*, Association for Computing Machinery: New York, NY, USA, 2007, 2–es.

(*63*) Herlihy, M., and Shavit, N., *The Art of Multiprocessor Programming, Revised Reprint*, 1st; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2012; 536 pp.

(*64*) Lindholm, E., Nickolls, J., Oberman, S., and Montrym, J. (2008). NVIDIA Tesla: A Unified Graphics and Computing Architecture. *IEEE Micro 28*, 39–55.

(*65*) CUDA C++ Programming Guide `https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#c-language-extensions` (accessed Dec. 8, 2024).

(*66*) Harris, C. R. et al. (2020). Array programming with NumPy. *Nature 585*, 357–362.

(*67*) Virtanen, P. et al. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods 17*, 261–272.

(*68*) Okuta, R., Unno, Y., Nishino, D., Hido, S., and Loomis, C. In *CuPy: A NumPy-Compatible Library for NVIDIA GPU Calculations*, 2017.

(*69*) Lam, S. K., Pitrou, A., and Seibert, S. In *Numba: a LLVM-based Python JIT compiler*, Association for Computing Machinery: New York, NY, USA, 2015, pp 1–6.

(*70*) Huang, Z., Ma, N., Wang, S., and Peng, Y. (2019). GPU computing performance analysis on matrix multiplication. *The Journal of Engineering 2019*, 9043–9048.

(*71*) Sarker, I. H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science 2*, 160.

(*72*) Jordan, M. I., and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science 349*, 255–260.

(*73*) Carbonell, J. G., Michalski, R. S., and Mitchell, T. M. In *Machine Learning*, Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., Eds.; Morgan Kaufmann: San Francisco (CA), 1983, pp 3–23.

(*74*) Cid Vidal, X., Dieste Maroñas, L., and Dosil Suárez, Á. In *Machine Learning and Artificial Intelligence with Industrial Applications: From Big Data to Small Data*, Carou, D., Sartal, A., and Davim, J. P., Eds.; Springer International Publishing: Cham, 2022, pp 19–61.

(*75*) Carleo, G., Cirac, I., Cranmer, K., Daudet, L., Schuld, M., Tishby, N., Vogt-Maranto, L., and Zdeborová, L. (2019). Machine learning and the physical sciences. *Reviews of Modern Physics 91*, 045002.

(*76*) Steinkraus, D., Buck, I., and Simard, P. In *Using GPUs for machine learning algorithms*, Eighth International Conference on Document Analysis and Recognition (ICDAR'05), 2005, 1115–1120 Vol. 2.

(*77*) Mittal, S., and Vaishay, S. (2019). A survey of techniques for optimizing deep learning on GPUs. *Journal of Systems Architecture 99*, 101635.

(*78*) Wang, Y. E., Wei, G.-Y., and Brooks, D. (2019). Benchmarking TPU, GPU, and CPU Platforms for Deep Learning. DOI: 10.48550/arXiv.1907.10701.

(*79*) Pandey, M., Fernandez, M., Gentile, F., Isayev, O., Tropsha, A., Stern, A. C., and Cherkasov, A. (2022). The transformational role of GPU computing and deep learning in drug discovery. *Nature Machine Intelligence 4*, 211–221.

(*80*) Buber, E., and Diri, B. In *Performance Analysis and CPU vs GPU Comparison for Deep Learning*, 2018 6th International Conference on Control Engineering & Information Technology (CEIT), 2018, pp 1–6.

(*81*) Jeon, W., Ko, G., Lee, J., Lee, H., Ha, D., and Ro, W. W. In *Advances in Computers*, Kim, S., and Deka, G. C., Eds.; Hardware Accelerator Systems for Artificial Intelligence and Machine Learning, Vol. 122; Elsevier: 2021, pp 167–215.

(*82*) Rojas, R. In *Neural Networks: A Systematic Introduction*, Rojas, R., Ed.; Springer: Berlin, Heidelberg, 1996, pp 149–182.

(*83*) Widrow, B., and Lehr, M. (1990). 30 years of adaptive neural networks: perceptron, Madaline, and backpropagation. *Proceedings of the IEEE 78*, 1415–1442.

(*84*) Rosenblatt, F., *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*; Spartan Books: 1962; 648 pp.

(*85*) Griewank, A., and Walther, A., *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, Second; Society for Industrial and Applied Mathematics: USA, 2008.

(*86*) Skansi, S., *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*; Springer: 2018; 196 pp.

(*87*) Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2018). Automatic Differentiation in Machine Learning: a Survey. *Journal of Machine Learning Research 18*, 1–43.

(*88*) Fei-Fei, L., Fergus, R., and Perona, P. (2006). One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence 28*, 594–611.

(*89*) Dhal, P., and Azad, C. (2022). A comprehensive survey on feature selection in the various fields of machine learning. *Applied Intelligence 52*, 4543–4581.

(*90*) Hartshorne, R., *Algebraic Geometry*; Graduate Texts in Mathematics, Vol. 52; Springer: New York, NY, 1977.

(*91*) Lee, J. M. In *Introduction to Smooth Manifolds*, Lee, J. M., Ed.; Springer: New York, NY, 2012, pp 1–31.

(*92*) Bronstein, M. M., Bruna, J., Cohen, T., and Veliĉković, P. (2021). Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. DOI: `10.48550/arXiv.2104.13478`.

(*93*) Cao, W., Yan, Z., He, Z., and He, Z. (2020). A comprehensive survey on geometric deep learning. *IEEE Access 8*, 35929–35949.

(*94*) Spivak, M., *Calculus On Manifolds: A Modern Approach To Classical Theorems Of Advanced Calculus*; CRC Press: Boca Raton, 2018; 162 pp.

(*95*) Guillemin, V., and Pollack, A., *Differential Topology*; American Mathematical Soc.: 2010; 242 pp.

(*96*) Neuenhofen, M. (2018). Review of theory and implementation of hyper-dual numbers for first and second order automatic differentiation. DOI: `10.48550/arXiv.1801.03614`.

(*97*) Griewank, A. (2012). Who invented the reverse mode of differentiation. *Documenta Mathematica, Extra Volume ISMP 389400*.

(*98*) Paszke, A. et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. DOI: `10.48550/arXiv.1912.01703`.

(*99*) Martin Abadi et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems `https://www.tensorflow.org/`.

(*100*) Bradbury, J. et al. JAX: composable transformations of Python+NumPy programs version 0.3.13, `http://github.com/jax-ml/jax`.

(*101*) Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks 2*, 359–366.

(*102*) Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019). How Powerful are Graph Neural Networks? DOI: `10.48550/arXiv.1810.00826`.

(*103*) Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Sohl-Dickstein, J. In *international conference on machine learning*, PMLR: 2017, pp 2847–2854.

(*104*) Popescu, M.-C., Balas, V. E., Perescu-Popescu, L., and Mastorakis, N. (2009). Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems 8*, 579–588.

(*105*) Baum, E. B. (1988). On the capabilities of multilayer perceptrons. *Journal of complexity 4*, 193–215.

(*106*) Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 2278–2324.

(*107*) Fukushima, K., Miyake, S., and Ito, T. (1983). Neocognitron: A neural network model for a mechanism of visual pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics SMC-13*, 826–834.

(*108*) Hadji, I., and Wildes, R. P. (2018). What Do We Understand About Convolutional Networks? DOI: 10.48550/arXiv.1803.08834.

(*109*) Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems 25*.

(*110*) Gholamalinezhad, H., and Khosravi, H. (2020). Pooling Methods in Deep Neural Networks, a Review. DOI: 10.48550/arXiv.2009.07485.

(*111*) Ciregan, D., Meier, U., and Schmidhuber, J. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp 3642–3649.

(*112*) McGreivy, N., and Hakim, A. (2023). Convolutional layers are equivariant to discrete shifts but not continuous translations. DOI: 10.48550/arXiv.2206.04979.

(*113*) Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp 248–255.

(*114*) Beyer, L., Hénaff, O. J., Kolesnikov, A., Zhai, X., and Oord, A. v. d. (2020). Are we done with ImageNet? DOI: 10.48550/arXiv.2006.07159.

(*115*) Zhou, M., Duan, N., Liu, S., and Shum, H.-Y. (2020). Progress in Neural NLP: Modeling, Learning, and Reasoning. *Engineering 6*, 275–290.

(*116*) Van Houdt, G., Mosquera, C., and Nápoles, G. (2020). A review on the long short-term memory model. *Artificial Intelligence Review 53*, 5929–5955.

(*117*) Yu, Y., Si, X., Hu, C., and Zhang, J. (2019). A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Computation 31*, 1235–1270.

(*118*) Brauwers, G., and Frasincar, F. (2023). A General Survey on Attention Mechanisms in Deep Learning. *IEEE Transactions on Knowledge and Data Engineering 35*, 3279–3298.

(*119*) Vaswani, A. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.

(*120*) Lin, T., Wang, Y., Liu, X., and Qiu, X. (2022). A survey of transformers. *AI Open 3*, 111–132.

(*121*) Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., and Shah, M. (2022). Transformers in vision: A survey. *ACM computing surveys (CSUR) 54*, 1–41.

(*122*) Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., and Sun, L. (2023). Transformers in Time Series: A Survey. DOI: 10.48550/arXiv.2202.07125.

(*123*) Ke, G., He, D., and Liu, T.-Y. (2021). Rethinking Positional Encoding in Language Pre-training. DOI: 10.48550/arXiv.2006.15595.

(*124*) Dufter, P., Schmitt, M., and Schütze, H. (2022). Position Information in Transformers: An Overview. *Computational Linguistics 48*, 733–763.

(*125*) Jumper, J. et al. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature 596*, 583–589.

(*126*) Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open 1*, 57–81.

(*127*) Wu, M., Yan, M., Li, W., Ye, X., Fan, D., and Xie, Y. (2024). A Comprehensive Survey on GNN Characterization. DOI: 10.48550/arXiv.2408.01902.

(*128*) Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. (2021). A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems 32*, 4–24.

(*129*) Khemani, B., Patil, S., Kotecha, K., and Tanwar, S. (2024). A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions. *Journal of Big Data 11*, 18.

(*130*) Gong, L., and Cheng, Q. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp 9211–9219.

(*131*) Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. In *Machine Learning Meets Quantum Physics*, Schütt, K. T., Chmiela, S., von Lilienfeld, O. A., Tkatchenko, A., Tsuda, K., and Müller, K.-R., Eds.; Springer International Publishing: Cham, 2020, pp 199–214.

(*132*) Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. In *International conference on machine learning*, 2017, pp 1263–1272.

(*133*) Kipf, T. N., and Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. DOI: 10.48550/arXiv.1609.02907.

(*134*) Hamilton, W. L., Ying, R., and Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. DOI: 10.48550/arXiv.1706.02216.

(*135*) Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph Attention Networks. DOI: 10.48550/arXiv.1710.10903.

(*136*) Brody, S., Alon, U., and Yahav, E. (2022). How Attentive are Graph Attention Networks? DOI: 10.48550/arXiv.2105.14491.

(*137*) Maglogiannis, I. G., *Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies*; IOS Press: 2007; 420 pp.

(*138*) *Unsupervised Learning Algorithms*; Celebi, M. E., and Aydin, K., Eds.; Springer International Publishing: Cham, 2016.

(*139*) Usama, M., Qadir, J., Raza, A., Arif, H., Yau, K.-l. A., Elkhatib, Y., Hussain, A., and Al-Fuqaha, A. (2019). Unsupervised Machine Learning for Networking: Techniques, Applications and Research Challenges. *IEEE Access 7*, 65579–65615.

(*140*) Bengio, Y., Courville, A. C., and Vincent, P. (2012). Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR, abs/1206.5538 1*, 2012.

(*141*) Bond-Taylor, S., Leach, A., Long, Y., and Willcocks, C. G. (2022). Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence 44*, 7327–7347.

(*142*) Gm, H., Gourisaria, M. K., Pandey, M., and Rautaray, S. S. (2020). A comprehensive survey and analysis of generative models in machine learning. *Computer Science Review 38*, 100285.

(*143*) Gui, J., Sun, Z., Wen, Y., Tao, D., and Ye, J. (2020). A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications. DOI: 10.48550/arXiv.2001.06937.

(*144*) Narkhede, M. V., Bartakke, P. P., and Sutaone, M. S. (2022). A review on weight initialization strategies for neural networks. *Artificial Intelligence Review 55*, 291–322.

(*145*) Skorski, M., Temperoni, A., and Theobald, M. (2020). Revisiting Initialization of Neural Networks. DOI: 10.48550/arXiv.2004.09506.

(*146*) Kiefer, J., and Wolfowitz, J. (1952). Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics 23*, 462–466.

(*147*) Kingma, D. P., and Ba, J. (2017). Adam: A Method for Stochastic Optimization. DOI: 10.48550/arXiv.1412.6980.

(*148*) Graves, A. (2014). Generating Sequences With Recurrent Neural Networks. DOI: 10.48550/arXiv.1308.0850.

(*149*) Briscoe, E., and Feldman, J. (2011). Conceptual complexity and the bias/variance tradeoff. *Cognition 118*, 2–16.

(*150*) Neal, B., Mittal, S., Baratin, A., Tantia, V., Scicluna, M., Lacoste-Julien, S., and Mitliagkas, I. (2019). A Modern Take on the Bias-Variance Tradeoff in Neural Networks. DOI: 10.48550/arXiv.1810.08591.

(*151*) He, F., Liu, T., and Tao, D. In *Advances in Neural Information Processing Systems*, Curran Associates, Inc.: 2019; Vol. 32.

(*152*) Radiuk, P. M. (2017). Impact of training set batch size on the performance of convolutional neural networks for diverse datasets. *Information Technology and Management Science*.

(*153*) Lin, R. In *2022 International Conference on Machine Learning and Knowledge Engineering (MLKE)*, 2022 International Conference on Machine Learning and Knowledge Engineering (MLKE), 2022, pp 106–109.

(*154*) Tan, J., Yang, J., Wu, S., Chen, G., and Zhao, J. (2021). A critical look at the current train/test split in machine learning. DOI: 10.48550/arXiv.2106.04525.

(*155*) James, G., Witten, D., Hastie, T., Tibshirani, R., and Taylor, J., *An Introduction to Statistical Learning: with Applications in Python*; Springer Texts in Statistics; Springer International Publishing: Cham, 2023.

(*156*) Yu, T., and Zhu, H. (2020). Hyper-Parameter Optimization: A Review of Algorithms and Applications. DOI: 10.48550/arXiv.2003.05689.

(*157*) Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. In *Advances in Neural Information Processing Systems*, Curran Associates, Inc.: 2011; Vol. 24.

(*158*) Yang, L., and Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing 415*, 295–316.

(*159*) Feurer, M., and Hutter, F. In *Automated Machine Learning: Methods, Systems, Challenges*, Hutter, F., Kotthoff, L., and Vanschoren, J., Eds.; Springer International Publishing: Cham, 2019, pp 3–33.

(*160*) Bischl, B. et al. (2023). Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *WIREs Data Mining and Knowledge Discovery 13*, e1484.

(*161*) Hollingsworth, S. A., and Dror, R. O. (2018). Molecular Dynamics Simulation for All. *Neuron 99*, 1129–1143.

(*162*) Karplus, M., and Petsko, G. A. (1990). Molecular Dynamics Simulations in Biology. *Nature 347*, 631–639.

(*163*) Leimkuhler, B., and Matthews, C., *Molecular Dynamics: With Deterministic and Stochastic Numerical Methods*; Interdisciplinary Applied Mathematics, Vol. 39; Springer International Publishing: Cham, 2015.

(*164*) Karplus, M., and Kuriyan, J. (2005). Molecular Dynamics and Protein Function. *Proceedings of the National Academy of Sciences 102*, 6679–6685.

(*165*) Durrant, J. D., and McCammon, J. A. (2011). Molecular Dynamics Simulations and Drug Discovery. *BMC Biology 9*, 71.

(*166*) González, M. A. (2011). Force Fields and Molecular Dynamics Simulations. *École thématique de la Société Française de la Neutronique 12*, 169–200.

(*167*) Monticelli, L., and Tieleman, D. P. In *Biomolecular Simulations: Methods and Protocols*, Monticelli, L., and Salonen, E., Eds.; Humana Press: Totowa, NJ, 2013, pp 197–213.

(*168*) Guvench, O., and MacKerell, A. D. In *Molecular Modeling of Proteins*, Kukol, A., Ed.; Humana Press: Totowa, NJ, 2008, pp 63–88.

(*169*) Saunders, M. G., and Voth, G. A. (2013). Coarse-Graining Methods for Computational Biology. *Annual Review of Biophysics 42*, 73–93.

(*170*) Huang, J., and MacKerell, A. D. (2013). CHARMM36 All-Atom Additive Protein Force Field: Validation Based on Comparison to NMR Data. *Journal of computational chemistry 34*, 2135–2145.

(*171*) Huang, J., Rauscher, S., Nawrocki, G., Ran, T., Feig, M., de Groot, B. L., Grubmüller, H., and MacKerell, A. D. (2017). CHARMM36m: An Improved Force Field for Folded and Intrinsically Disordered Proteins. *Nature methods 14*, 71–73.

(*172*) Salomon-Ferrer, R., Case, D. A., and Walker, R. C. (2013). An Overview of the Amber Biomolecular Simulation Package. *WIREs Computational Molecular Science 3*, 198–210.

(*173*) Hornak, V., Abel, R., Okur, A., Strockbine, B., Roitberg, A., and Simmerling, C. (2006). Comparison of Multiple Amber Force Fields and Development of Improved Protein Backbone Parameters. *Proteins: Structure, Function, and Bioinformatics 65*, 712–725.

(*174*) Filho, R. N. C., Alencar, G., Skagerstam, B.-S., and Andrade, J. S. (2013). Morse Potential Derived from First Principles. *Europhysics Letters 101*, 10009.

(*175*) Vanommeslaeghe, K., Guvench, O., and MacKerell, A. D. (2014). Molecular Mechanics. *Current pharmaceutical design 20*, 3281–3292.

(*176*) Allinger, N. L., Li, F., and Yan, L. (1990). Molecular Mechanics. The MM3 Force Field for Alkenes. *Journal of Computational Chemistry 11*, 848–867.

(*177*) Mackerell Jr., A. D., Feig, M., and Brooks III, C. L. (2004). Extending the Treatment of Backbone Energetics in Protein Force Fields: Limitations of Gas-Phase Quantum Mechanics in Reproducing Protein Conformational Distributions in Molecular Dynamics Simulations. *Journal of Computational Chemistry 25*, 1400–1415.

(*178*) Brooks, B. R. et al. (2009). CHARMM: The Biomolecular Simulation Program. *Journal of Computational Chemistry 30*, 1545–1614.

(*179*) Buck, M., Bouguet-Bonnet, S., Pastor, R. W., and MacKerell, A. D. (2006). Importance of the CMAP Correction to the CHARMM22 Protein Force Field: Dynamics of Hen Lysozyme. *Biophysical Journal 90*, L36–L38.

(*180*) Tian, C. et al. (2020). ff19SB: Amino-Acid-Specific Protein Backbone Parameters Trained against Quantum Mechanics Energy Surfaces in Solution. *Journal of Chemical Theory and Computation 16*, 528–552.

(*181*) Zhang, C., Lu, C., Jing, Z., Wu, C., Piquemal, J.-P., Ponder, J. W., and Ren, P. (2018). AMOEBA Polarizable Atomic Multipole Force Field for Nucleic Acids. *Journal of chemical theory and computation 14*, 2084–2108.

(*182*) Salsbury, A. M., and Lemkul, J. A. (2021). Recent Developments in Empirical Atomistic Force Fields for Nucleic Acids and Applications to Studies of Folding and Dynamics. *Current Opinion in Structural Biology 67*, 9–17.

(*183*) Halgren, T. A. (1992). The Representation of van Der Waals (vdW) Interactions in Molecular Mechanics Force Fields: Potential Form, Combination Rules, and vdW Parameters. *Journal of the American Chemical Society 114*, 7827–7843.

(*184*) Fischer, J., and Wendland, M. (2023). On the History of Key Empirical Intermolecular Potentials. *Fluid Phase Equilibria 573*, 113876.

(*185*) Nicolini, P., Guàrdia, E., and Masia, M. (2013). Shortcomings of the Standard Lennard–Jones Dispersion Term in Water Models, Studied with Force Matching. *The Journal of Chemical Physics 139*, 184111.

(*186*) Pirani, F., Brizi, S., Roncaratti, L. F., Casavecchia, P., Cappelletti, D., and Vecchiocattivi, F. (2008). Beyond the Lennard-Jones Model: A Simple and Accurate Potential Function Probed by High Resolution Scattering Data Useful for Molecular Dynamics Simulations. *Physical Chemistry Chemical Physics 10*, 5489–5503.

(*187*) Wang, X., Ramírez-Hinestrosa, S., Dobnikar, J., and Frenkel, D. (2020). The Lennard-Jones Potential: When (Not) to Use It. *Physical Chemistry Chemical Physics 22*, 10624–10633.

(*188*) Ercolessi, F., and Adams, J. B. (1994). Interatomic Potentials from First-Principles Calculations: The Force-Matching Method. *Europhysics Letters (EPL) 26*, 583–588.

(*189*) He, X., Walker, B., Man, V. H., Ren, P., and Wang, J. (2022). Recent Progress in General Force Fields of Small Molecules. *Current Opinion in Structural Biology 72*, 187–193.

(*190*) Polêto, M. D., and Lemkul, J. A. (2022). Integration of Experimental Data and Use of Automated Fitting Methods in Developing Protein Force Fields. *Communications Chemistry 5*, 1–10.

(*191*) Lopes, P. E., Guvench, O., and MacKerell, A. D. (2015). Current Status of Protein Force Fields for Molecular Dynamics. *Methods in molecular biology (Clifton, N.J.) 1215*, 47–71.

(*192*) Ouyang, J. F., and Bettens, R. P. A. (2015). Modelling Water: A Lifetime Enigma. *Chimia 69*, 104–111.

(*193*) Bellissent-Funel, M.-C., Hassanali, A., Havenith, M., Henchman, R., Pohl, P., Sterpone, F., van der Spoel, D., Xu, Y., and Garcia, A. E. (2016). Water Determines the Structure and Dynamics of Proteins. *Chemical Reviews 116*, 7673–7697.

(*194*) Huggins, D. J., and Tidor, B. (2011). Systematic Placement of Structural Water Molecules for Improved Scoring of Protein–Ligand Interactions. *Protein Engineering, Design and Selection 24*, 777–789.

(*195*) Ido, S., Kobayashi, K., Oyabu, N., Hirata, Y., Matsushige, K., and Yamada, H. (2022). Structured Water Molecules on Membrane Proteins Resolved by Atomic Force Microscopy. *Nano Letters 22*, 2391–2397.

(*196*) Onufriev, A. In *Annual Reports in Computational Chemistry*, Wheeler, R. A., and Spellmeyer, D. C., Eds.; Elsevier: 2008; Vol. 4, pp 125–137.

(*197*) Guillot, B. (2002). A Reappraisal of What We Have Learnt during Three Decades of Computer Simulations on Water. *Journal of Molecular Liquids 101*, 219–260.

(*198*) Jorgensen, W. L. (1981). Quantum and Statistical Mechanical Studies of Liquids. 10. Transferable Intermolecular Potential Functions for Water, Alcohols, and Ethers. Application to Liquid Water. *Journal of the American Chemical Society 103*, 335–340.

(*199*) Mao, Y., and Zhang, Y. (2012). Thermal Conductivity, Shear Viscosity and Specific Heat of Rigid Water Models. *Chemical Physics Letters 542*, 37–41.

(*200*) Berendsen, H. J. C., Postma, J. P. M., van Gunsteren, W. F., and Hermans, J. In *Intermolecular Forces: Proceedings of the Fourteenth Jerusalem Symposium on Quantum Chemistry and Biochemistry Held in Jerusalem, Israel, April 13–16, 1981*, Pullman, B., Ed.; Springer Netherlands: Dordrecht, 1981, pp 331–342.

(*201*) Bukowski, R., Szalewicz, K., Groenenboom, G. C., and van der Avoird, A. (2007). Predictions of the Properties of Water from First Principles. *Science 315*, 1249–1252.

(*202*) Gomes, A. M. M., Costa, P. J., and Machuqueiro, M. (2023). Recent Advances on Molecular Dynamics-Based Techniques to Address Drug Membrane Permeability with Atomistic Detail. *BBA Advances 4*, 100099.

(*203*) Ibragimova, G. T., and Wade, R. C. (1998). Importance of Explicit Salt Ions for Protein Stability in Molecular Dynamics Simulation. *Biophysical Journal 74*, 2906–2911.

(*204*) Schmit, J. D., Kariyawasam, N. L., Needham, V., and Smith, P. E. (2018). SLTCAP: A Simple Method for Calculating the Number of Ions Needed for MD Simulation. *Journal of Chemical Theory and Computation 14*, 1823–1827.

(*205*) Bou-Rabee, N. (2014). Time Integrators for Molecular Dynamics. *Entropy 16*, 138–162.

(*206*) Yu, T.-Q., Alejandre, J., López-Rendón, R., Martyna, G. J., and Tuckerman, M. E. (2010). Measure-Preserving Integrators for Molecular Dynamics in the Isothermal–Isobaric Ensemble Derived from the Liouville Operator. *Chemical Physics 370*, 294–305.

(*207*) Gray, S. K., Noid, D. W., and Sumpter, B. G. (1994). Symplectic Integrators for Large Scale Molecular Dynamics Simulations: A Comparison of Several Explicit Methods. *The Journal of Chemical Physics 101*, 4062–4072.

(*208*) Abraham, M. J., Murtola, T., Schulz, R., Páll, S., Smith, J. C., Hess, B., and Lindahl, E. (2015). GROMACS: High Performance Molecular Simulations through Multi-Level Parallelism from Laptops to Supercomputers. *SoftwareX 1–2*, 19–25.

(*209*) Case, D. A. et al. (2023). AmberTools. *Journal of Chemical Information and Modeling 63*, 6183–6191.

(*210*) Phillips, J. C. et al. (2020). Scalable Molecular Dynamics on CPU and GPU Architectures with NAMD. *The Journal of Chemical Physics 153*, 044130.

(*211*) Klepeis, J. L., Lindorff-Larsen, K., Dror, R. O., and Shaw, D. E. (2009). Long-Timescale Molecular Dynamics Simulations of Protein Structure and Function. *Current Opinion in Structural Biology 19*, 120–127.

(*212*) Zwier, M. C., and Chong, L. T. (2010). Reaching Biological Timescales with All-Atom Molecular Dynamics Simulations. *Current Opinion in Pharmacology 10*, 745–752.

(*213*) Shim, K. S., Greskamp, B., Towles, B., Edwards, B., Grossman, J. P., and Shaw, D. E. (2022). The Specialized High-Performance Network on Anton 3. DOI: `10.48550/arXiv.2201.08357`.

(*214*) Schlick, T., *Molecular Modeling and Simulation*; Springer: 2002.

(*215*) Bekker, H., Dijkstra, E. J., Renardus, M. K. R., and Berendsen, H. J. C. (1995). An Efficient, Box Shape Independent Non-Bonded Force and Virial Algorithm for Molecular Dynamics. *Molecular Simulation 14*, 137–151.

(*216*) Herce, H. D., Garcia, A. E., and Darden, T. (2007). The Electrostatic Surface Term: (I) Periodic Systems. *The Journal of Chemical Physics 126*, 124106.

(*217*) Borwein, D., Borwein, J. M., and Taylor, K. F. (1985). Convergence of Lattice Sums and Madelung's Constant. *Journal of Mathematical Physics 26*, 2999–3009.

(*218*) Sakamoto, Y. (1958). Madelung Constants of Simple Crystals Expressed in Terms of Born's Basic Potentials of 15 Figures. *The Journal of Chemical Physics 28*, 164–165.

(*219*) Stamm, B., Lagardère, L., Polack, É., Maday, Y., and Piquemal, J.-P. (2018). A Coherent Derivation of the Ewald Summation for Arbitrary Orders of Multipoles: The Self-Terms. *The Journal of Chemical Physics 149*, 124103.

(*220*) York, D. M., Wlodawer, A., Pedersen, L. G., and Darden, T. A. (1994). Atomic-Level Accuracy in Simulations of Large Protein Crystals. *Proceedings of the National Academy of Sciences of the United States of America 91*, 8715–8718.

(*221*) Darden, T., York, D., and Pedersen, L. (1993). Particle Mesh Ewald: An N·log(N) Method for Ewald Sums in Large Systems. *The Journal of Chemical Physics 98*, 10089–10092.

(*222*) Essmann, U., Perera, L., Berkowitz, M. L., Darden, T., Lee, H., and Pedersen, L. G. (1995). A Smooth Particle Mesh Ewald Method. *The Journal of Chemical Physics 103*, 8577–8593.

(*223*) Harvey, M. J., and De Fabritiis, G. (2009). An Implementation of the Smooth Particle Mesh Ewald Method on GPU Hardware. *Journal of Chemical Theory and Computation 5*, 2371–2377.

(*224*) Guo, X., Liu, X., Xu, P., Du, Z., and Chow, E. (2015). Efficient Particle-mesh Spreading on GPUs. *Procedia Computer Science 51*, 120–129.

(*225*) Ke, Q., Gong, X., Liao, S., Duan, C., and Li, L. (2022). Effects of Thermostats/Barostats on Physical Properties of Liquids by Molecular Dynamics Simulations. *Journal of Molecular Liquids 365*, 120116.

(*226*) Hünenberger, P. H. In *Advanced Computer Simulation: Approaches for Soft Matter Sciences I*, Dr. Holm, C., and Prof. Dr. Kremer, K., Eds.; Springer: Berlin, Heidelberg, 2005, pp 105–149.

(*227*) Berendsen, H. J. C., Postma, J. P. M., van Gunsteren, W. F., DiNola, A., and Haak, J. R. (1984). Molecular Dynamics with Coupling to an External Bath. *Journal of Chemical Physics 81*, 3684–3690.

(*228*) Hoover, W. G. (1985). Canonical Dynamics: Equilibrium Phase-Space Distributions. *Physical Review A 31*, 1695–1697.

(*229*) Paquet, E., and Viktor, H. L. (2015). Molecular Dynamics, Monte Carlo Simulations, and Langevin Dynamics: A Computational Review. *BioMed Research International 2015*, 183918.

(*230*) Pastor, R. W., Brooks, B. R., and Szabo, A. (1988). An Analysis of the Accuracy of Langevin and Molecular Dynamics Algorithms. *Molecular Physics 65*, 1409–1419.

(*231*) Han, K.-K., and Son, H. S. (2001). On the Isothermal-Isobaric Ensemble Partition Function. *The Journal of Chemical Physics 115*, 7793–7794.

(*232*) Jorgensen, W. L., and Tirado-Rives, J. (2005). Potential Energy Functions for Atomic-Level Simulations of Water and Organic and Biomolecular Systems. *Proceedings of the National Academy of Sciences 102*, 6665–6670.

(*233*) Bernetti, M., and Bussi, G. (2020). Pressure Control Using Stochastic Cell Rescaling. *The Journal of Chemical Physics 153*, 114107.

(*234*) Lin, Y., Pan, D., Li, J., Zhang, L., and Shao, X. (2017). Application of Berendsen Barostat in Dissipative Particle Dynamics for Nonequilibrium Dynamic Simulation. *The Journal of Chemical Physics 146*, 124108.

(*235*) Parrinello, M., and Rahman, A. (1980). Crystal Structure and Pair Potentials: A Molecular-Dynamics Study. *Physical Review Letters 45*, 1196–1199.

(*236*) Nosé, S., and Klein, M. (1983). Constant Pressure Molecular Dynamics for Molecular Systems. *Molecular Physics 50*, 1055–1076.

(*237*) Martyna, G. J., Tuckerman, M. E., Tobias, D. J., and Klein, M. L. (1996). Explicit Reversible Integrators for Extended Systems Dynamics. *Molecular Physics 87*, 1117–1157.

(*238*) Quigley, D., and Probert, M. I. J. (2004). Langevin Dynamics in Constant Pressure Extended Systems. *The Journal of Chemical Physics 120*, 11432–11441.

(*239*) Feller, S. E., Zhang, Y., Pastor, R. W., and Brooks, B. R. (1995). Constant Pressure Molecular Dynamics Simulation: The Langevin Piston Method. *The Journal of Chemical Physics 103*, 4613–4621.

(*240*) Abraham, M. et al. (2024). GROMACS 2024.4 Manual. DOI: `10.5281/zenodo.14016613`.

(*241*) Zhang, X., Wu, F., Yang, N., Zhan, X., Liao, J., Mai, S., and Huang, Z. (2022). In Silico Methods for Identification of Potential Therapeutic Targets. *Interdisciplinary Sciences: Computational Life Sciences 14*, 285–310.

(*242*) Zheng, C. J., Han, L. Y., Yap, C. W., Ji, Z. L., Cao, Z. W., and Chen, Y. Z. (2006). Therapeutic Targets: Progress of Their Exploration and Investigation of Their Characteristics. *Pharmacological Reviews 58*, 259–279.

(*243*) Nicolette, C. A., and Miller, G. A. (2003). The Identification of Clinically Relevant Markers and Therapeutic Targets. *Drug Discovery Today 8*, 31–38.

(*244*) Plenge, R. M., Scolnick, E. M., and Altshuler, D. (2013). Validating Therapeutic Targets through Human Genetics. *Nature Reviews Drug Discovery 12*, 581–594.

(*245*) Zheng, C., Han, L., Yap, C. W., Xie, B., and Chen, Y. (2006). Progress and Problems in the Exploration of Therapeutic Targets. *Drug Discovery Today 11*, 412–420.

(*246*) Dixon, S. J., and Stockwell, B. R. (2009). Identifying Druggable Disease-Modifying Gene Products. *Current opinion in chemical biology 13*, 549–555.

(*247*) Sayers, E. W., Cavanaugh, M., Clark, K., Ostell, J., Pruitt, K. D., and Karsch-Mizrachi, I. (2020). GenBank. *Nucleic Acids Research 48*, D84–D86.

(*248*) The UniProt Consortium (2024). UniProt: The Universal Protein Knowledgebase in 2025. *Nucleic Acids Research*, gkae1010.

(*249*) Hasegawa, H., and Holm, L. (2009). Advances and Pitfalls of Protein Structural Alignment. *Current Opinion in Structural Biology 19*, 341–348.

(*250*) Do, C. B., and Katoh, K. In *Functional Proteomics: Methods and Protocols*, Thompson, J. D., Schaeffer-Reiss, C., and Ueffing, M., Eds.; Humana Press: Totowa, NJ, 2008, pp 379–413.

(*251*) Taylor, W. R., and Orengo, C. A. (1989). Protein Structure Alignment. *Journal of Molecular Biology 208*, 1–22.

(*252*) Lipman, D. J., and Pearson, W. R. (1985). Rapid and Sensitive Protein Similarity Searches. *Science 227*, 1435–1441.

(*253*) Pearson, W. R., and Lipman, D. J. (1988). Improved Tools for Biological Sequence Comparison. *Proceedings of the National Academy of Sciences of the United States of America 85*, 2444–2448.

(*254*) Polyanovsky, V. O., Roytberg, M. A., and Tumanyan, V. G. (2011). Comparative Analysis of the Quality of a Global Algorithm and a Local Algorithm for Alignment of Two Sequences. *Algorithms for Molecular Biology : AMB 6*, 25.

(*255*) Needleman, S. B., and Wunsch, C. D. (1970). A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. *Journal of Molecular Biology 48*, 443–453.

(*256*) Smith, T. F., and Waterman, M. S. (1981). Identification of Common Molecular Subsequences. *Journal of Molecular Biology 147*, 195–197.

(*257*) Liu, Y., Huang, W., Johnson, J., and Vaidya, S. In *Computational Science–ICCS 2006: 6th International Conference, Reading, UK, May 28-31, 2006, Proceedings, Part IV 6*, Springer: 2006, pp 188–195.

(*258*) Manavski, S. A., and Valle, G. (2008). CUDA Compatible GPU Cards as Efficient Hardware Accelerators for Smith-Waterman Sequence Alignment. *BMC Bioinformatics 9*, S10.

(*259*) Dayhoff, M. O., *Atlas of Protein Sequence and Structure*; National Biomedical Research Foundation.: 1979.

(*260*) Wilbur, W. J. (1985). On the PAM Matrix Model of Protein Evolution. *Molecular Biology and Evolution 2*, 434–447.

(*261*) Moeckel, C., Mareboina, M., Konnaris, M. A., Chan, C. S. Y., Mouratidis, I., Montgomery, A., Chantzi, N., Pavlopoulos, G. A., and Georgakopoulos-Soares, I. (2024). A Survey of K-Mer Methods and Applications in Bioinformatics. *Computational and Structural Biotechnology Journal 23*, 2289–2303.

(*262*) Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic Local Alignment Search Tool. *Journal of Molecular Biology 215*, 403–410.

(*263*) Johnson, M., Zaretskaya, I., Raytselis, Y., Merezhuk, Y., McGinnis, S., and Madden, T. L. (2008). NCBI BLAST: A Better Web Interface. *Nucleic Acids Research 36*, W5–W9.

(*264*) Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K., and Madden, T. L. (2009). BLAST+: Architecture and Applications. *BMC Bioinformatics 10*, 421.

(*265*) Fox, N. K., Brenner, S. E., and Chandonia, J.-M. (2014). SCOPe: Structural Classification of Proteins—Extended, Integrating SCOP and ASTRAL Data and Classification of New Structures. *Nucleic Acids Research 42*, D304–D309.

(*266*) Orengo, C. A., Michie, A. D., Jones, S., Jones, D. T., Swindells, M. B., and Thornton, J. M. (1997). CATH – a Hierarchic Classification of Protein Domain Structures. *Structure 5*, 1093–1109.

(*267*) Holm, L., and Sander, C. (1994). The FSSP Database of Structurally Aligned Protein Fold Families. *Nucleic Acids Research 22*, 3600–3609.

(*268*) Orengo, C. A., and Thornton, J. M. (2005). PROTEIN FAMILIES AND THEIR EVOLUTION—A STRUCTURAL PERSPECTIVE. *Annual Review of Biochemistry 74*, 867–900.

(*269*) Vendruscolo, M., Kussell, E., and Domany, E. (1997). Recovery of Protein Structure from Contact Maps. *Folding and Design 2*, 295–306.

(*270*) Holm, L., Kääriäinen, S., Wilton, C., and Plewczynski, D. (2006). Using Dali for Structural Comparison of Proteins. *Current Protocols in Bioinformatics 14*, 5.5.1–5.5.24.

(*271*) Wohlers, I., Andonov, R., and Klau, G. W. (2012). Optimal DALI Protein Structure Alignment. *IEEE/ACM Trans Comput Biol Bioinform 20*.

(*272*) Wohlers, I., Andonov, R., and Klau, G. W. (2013). DALIX: Optimal DALI Protein Structure Alignment. *IEEE/ACM Transactions on Computational Biology and Bioinformatics 10*, 26–36.

(*273*) Kabsch, W. (1976). A Solution for the Best Rotation to Relate Two Sets of Vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography 32*, 922–923.

(*274*) Voight, J., *Quaternion Algebras*; Graduate Texts in Mathematics, Vol. 288; Springer International Publishing: Cham, 2021.

(*275*) Zhang, F. (1997). Quaternions and Matrices of Quaternions. *Linear Algebra and its Applications 251*, 21–57.

(*276*) Helmich, B., and Sierka, M. (2012). Similarity Recognition of Molecular Structures by Optimal Atomic Matching and Rotational Superposition. *Journal of Computational Chemistry 33*, 134–140.

(*277*) Hanson, A. J., and Thakur, S. (2012). Quaternion Maps of Global Protein Structure. *Journal of Molecular Graphics and Modelling 38*, 256–278.

(*278*) Coutsias, E. A., Seok, C., and Dill, K. A. (2004). Using Quaternions to Calculate RMSD. *Journal of Computational Chemistry 25*, 1849–1857.

(*279*) Hung, L.-H., Guerquin, M., and Samudrala, R. (2011). GPU-Q-J, a Fast Method for Calculating Root Mean Square Deviation (RMSD) after Optimal Superposition. *BMC Research Notes 4*, 97.

(*280*) Konagurthu, A. S., Whisstock, J. C., Stuckey, P. J., and Lesk, A. M. (2006). MUSTANG: A Multiple Structural Alignment Algorithm. *Proteins 64*, 559–574.

(*281*) Richards, F. M., and Kundrot, C. E. (1988). Identification of Structural Motifs from Protein Coordinate Data: Secondary Structure and First-Level Supersecondary Structure. *Proteins: Structure, Function, and Bioinformatics 3*, 71–84.

(*282*) Lustig, R. (1994). Statistical Thermodynamics in the Classical Molecular Dynamics Ensemble. I. Fundamentals. *The Journal of Chemical Physics 100*, 3048–3059.

(*283*) Laidler, K. J., and King, M. C. (1983). The Development of Transition-State Theory. *J. phys. Chem 87*, 2657–2664.

(*284*) Yang, Y. I., Shao, Q., Zhang, J., Yang, L., and Gao, Y. Q. (2019). Enhanced Sampling in Molecular Dynamics. *The Journal of Chemical Physics 151*, 070902.

(*285*) Bernardi, R. C., Melo, M. C. R., and Schulten, K. (2015). Enhanced Sampling Techniques in Molecular Dynamics Simulations of Biological Systems. *Biochimica et Biophysica Acta (BBA) - General Subjects 1850*, 872–877.

(*286*) Lazim, R., Suh, D., and Choi, S. (2020). Advances in Molecular Dynamics Simulations and Enhanced Sampling Methods for the Study of Protein Systems. *International Journal of Molecular Sciences 21*, 6339.

(*287*) Tiwary, P., and van de Walle, A. In *Multiscale Materials Modeling for Nanomechanics*, Weinberger, C. R., and Tucker, G. J., Eds.; Springer International Publishing: Cham, 2016, pp 195–221.

(*288*) Wang, A.-h., Zhang, Z.-c., and Li, G.-h. (2019). Advances in Enhanced Sampling Molecular Dynamics Simulations for Biomolecules. *Chinese Journal of Chemical Physics 32*, 277–286.

(*289*) Li, W., and Ma, A. (2014). Recent Developments in Methods for Identifying Reaction Coordinates. *Molecular Simulation 40*, 784–793.

(*290*) Peters, B. (2016). Reaction Coordinates and Mechanistic Hypothesis Tests. *Annual Review of Physical Chemistry 67*, 669–690.

(*291*) McGibbon, R. T., Husic, B. E., and Pande, V. S. (2017). Identification of Simple Reaction Coordinates from Complex Dynamics. *The Journal of Chemical Physics 146*, 044109.

(*292*) Hayward, S., and Go, N. (1995). Collective Variable Description of Native Protein Dynamics. *Annual Review of Physical Chemistry 46*, 223–250.

(*293*) Cohen, I., Huang, Y., Chen, J., Benesty, J., Benesty, J., Chen, J., Huang, Y., and Cohen, I. (2009). Pearson Correlation Coefficient. *Noise reduction in speech processing*, 1–4.

(*294*) Loğoğlu, K. B., and Ateş, T. K. In *2010 IEEE 18th Signal Processing and Communications Applications Conference*, 2010, pp 840–843.

(*295*) McCammon, J. (1984). Protein dynamics. *Reports on Progress in Physics 47*, 1.

(*296*) Ichiye, T., and Karplus, M. (1991). Collective motions in proteins: a covariance analysis of atomic fluctuations in molecular dynamics and normal mode simulations. *Proteins: Structure, Function, and Bioinformatics 11*, 205–217.

(*297*) Yu, H., and Dalby, P. A. In *Methods in Enzymology*, Tawfik, D. S., Ed.; Enzyme Engineering and Evolution: General Methods, Vol. 643; Academic Press: 2020, pp 15–49.

(*298*) Stein, S. A. M., Loccisano, A. E., Firestine, S. M., and Evanseck, J. D. In *Annual Reports in Computational Chemistry*, Spellmeyer, D. C., Ed.; Elsevier: 2006; Vol. 2, pp 233–261.

(*299*) Palma, J., and Pierdominici-Sottile, G. (2023). On the Uses of PCA to Characterise Molecular Dynamics Simulations of Biological Macromolecules: Basics and Tips for an Effective Use. *ChemPhysChem 24*, e202200491.

(*300*) Jolliffe, I. T., and Cadima, J. (2016). Principal Component Analysis: A Review and Recent Developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 374*, 20150202.

(*301*) Abdi, H., and Williams, L. J. (2010). Principal Component Analysis. *WIREs Computational Statistics 2*, 433–459.

(*302*) David, C. C., and Jacobs, D. J. (2014). Principal Component Analysis: A Method for Determining the Essential Dynamics of Proteins. *Methods in molecular biology (Clifton, N.J.) 1084*, 193–226.

(*303*) Frassek, M., Arjun, A., and Bolhuis, P. G. (2021). An Extended Autoencoder Model for Reaction Coordinate Discovery in Rare Event Molecular Dynamics Datasets. *The Journal of Chemical Physics 155*, 064103.

(*304*) Bandyopadhyay, S., and Mondal, J. (2021). A Deep Autoencoder Framework for Discovery of Metastable Ensembles in Biomacromolecules. *The Journal of Chemical Physics 155*, 114106.

(*305*) Tian, H., Jiang, X., Trozzi, F., Xiao, S., Larson, E. C., and Tao, P. (2021). Explore Protein Conformational Space With Variational Autoencoder. *Frontiers in Molecular Biosciences 8*, DOI: `10.3389/fmolb.2021.781635`.

(*306*) Goodfellow, I., Bengio, Y., and Courville, A., *Deep Learning*; The MIT Press: 2016.

(*307*) Cacciarelli, D., and Kulahci, M. (2023). Hidden Dimensions of the Data: PCA vs Autoencoders. *Quality Engineering 35*, 741–750.

(*308*) Kingma, D. P., and Welling, M. (2022). Auto-Encoding Variational Bayes. DOI: 10.48550/arXiv.1312.6114.

(*309*) Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association 112*, 859–877.

(*310*) Nguyen, D. An in Depth Introduction to Variational Bayes Note, SSRN Scholarly Paper, Rochester, NY, 2023.

(*311*) Ribeiro, J. M. L., Bravo, P., Wang, Y., and Tiwary, P. (2018). Reweighted Autoencoded Variational Bayes for Enhanced Sampling (RAVE). *The Journal of Chemical Physics 149*, 072301.

(*312*) Tomczak, J., and Welling, M. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, PMLR: 2018, pp 1214–1223.

(*313*) Tsuchiya, Y., Taneishi, K., and Yonezawa, Y. (2019). Autoencoder-Based Detection of Dynamic Allostery Triggered by Ligand Binding Based on Molecular Dynamics. *Journal of Chemical Information and Modeling 59*, 4043–4051.

(*314*) Zhang, W., and Liu, H. T. (2002). MAPK signal pathways in the regulation of cell proliferation in mammalian cells. *Cell Research 12*, 9–18.

(*315*) Shaw, A. S., and Filbert, E. L. (2009). Scaffold proteins and immune-cell signalling. *Nature Reviews Immunology 9*, 47–56.

(*316*) Plotnikov, A., Zehorai, E., Procaccia, S., and Seger, R. (2011). The MAPK cascades: Signaling components, nuclear roles and mechanisms of nuclear translocation. *Biochimica et Biophysica Acta (BBA) - Molecular Cell Research 1813*, 1619–1633.

(*317*) Guo, Y.-J., Pan, W.-W., Liu, S.-B., Shen, Z.-F., Xu, Y., and Hu, L.-L. (2020). ERK/MAPK signalling pathway and tumorigenesis (Review). *Exp Ther Med 19*, 1997–2007.

(*318*) Ardito, F., Giuliani, M., Perrone, D., Troiano, G., and Muzio, L. L. (2017). The crucial role of protein phosphorylation in cell signaling and its use as targeted therapy (review). *International Journal of Molecular Medicine 40*, 271–280.

(*319*) Guo, Y.-J., Pan, W.-W., Liu, S.-B., Shen, Z.-F., Xu, Y., and Hu, L.-L. (2020). ERK/MAPK signalling pathway and tumorigenesis (Review). *Experimental and Therapeutic Medicine 19*, 1997–2007.

(*320*) Li, L., Zhao, G.-D., Shi, Z., Qi, L.-L., Zhou, L.-Y., and Fu, Z.-X. (2016). The Ras/Raf/MEK/ERK signaling pathway and its role in the occurrence and development of HCC (Review). *Oncology Letters 12*, 3045–3050.

(*321*) Kim, E. K., and Choi, E.-J. (2010). Pathological roles of MAPK signaling pathways in human diseases. *Biochimica et Biophysica Acta (BBA) - Molecular Basis of Disease 1802*, 396–405.

(*322*) Sun, Y., Liu, W.-Z., Liu, T., Feng, X., Yang, N., and Zhou, H.-F. (2015). Signaling pathway of MAPK/ERK in cell proliferation, differentiation, migration, senescence and apoptosis. *Journal of Receptors and Signal Transduction 35*, 600–604.

(*323*) Molina, J. R., and Adjei, A. A. (2006). The Ras/Raf/MAPK Pathway. *Journal of Thoracic Oncology 1*, 7–9.

(*324*) Matallanas, D., Birtwistle, M., Romano, D., Zebisch, A., Rauch, J., von Kriegsheim, A., and Kolch, W. (2011). Raf family kinases: Old dogs have learned new tricks. *Genes and Cancer 2*, 232–260.

(*325*) Roskoski Jr, R. (2010). RAF protein-serine/threonine kinases: structure and regulation. *Biochemical and biophysical research communications 399*, 313–317.

(*326*) Hagemann, C., and Rapp, U. R. (1999). Isotype-specific functions of Raf kinases. *Experimental Cell Research 253*, 34–46.

(*327*) Cook, F. A., and Cook, S. J. (2021). Inhibition of RAF dimers: it takes two to tango. *Biochemical Society Transactions 49*, 237–251.

(*328*) Davies, H. et al. (2002). Mutations of the BRAF gene in human cancer. *Nature 417*, 949–954.

(*329*) Leicht, D. T., Balan, V., Kaplun, A., Singh-Gupta, V., Kaplun, L., Dobson, M., and Tzivion, G. (2007). Raf Kinases: Function, Regulation and Role in Human Cancer. *Biochimica et biophysica acta 1773*, 1196–1212.

(*330*) Terrell, E. M., and Morrison, D. K. (2019). Ras-Mediated Activation of the Raf Family Kinases. *Cold Spring Harbor Perspectives in Medicine 9*, a033746.

(*331*) Roskoski, R. (2010). RAF protein-serine/threonine kinases: Structure and regulation. *Biochemical and Biophysical Research Communications 399*, 313–317.

(*332*) Tzivion, G., Luo, Z., and Avruch, J. (1998). A dimeric 14-3-3 protein is an essential cofactor for Raf kinase activity. *Nature 394*, 88–92.

(*333*) Shaw, A. S., Kornev, A. P., Hu, J., Ahuja, L. G., and Taylor, S. S. (2014). Kinases and Pseudokinases: Lessons from RAF. *Molecular and Cellular Biology 34*, 1538–1546.

(*334*) Savoia, P., Fava, P., Casoni, F., and Cremona, O. (2019). Targeting the ERK Signaling Pathway in Melanoma. *International Journal of Molecular Sciences 20*, 1483.

(*335*) Kondo, Y., Ognjenović, J., Banerjee, S., Karandur, D., Merk, A., Kulhanek, K., Wong, K., Roose, J. P., Subramaniam, S., and Kuriyan, J. (2019). Cryo-EM structure of a dimeric B-Raf:14-3-3 complex reveals asymmetry in the active sites of B-Raf kinases. *Science 366*, 109–115.

(*336*) Davies, H., Bignell, G. R., Cox, C., Stephens, P., Edkins, S., Clegg, S., Teague, J., Woffendin, H., Garnett, M. J., Bottomley, W., et al. (2002). Mutations of the BRAF gene in human cancer. *Nature 417*, 949–954.

(*337*) Wellbrock, C., Karasarides, M., and Marais, R. (2004). The RAF proteins take centre stage. *Nature reviews Molecular cell biology 5*, 875–885.

(*338*) Hu, J., Stites, E. C., Yu, H., Germino, E. A., Meharena, H. S., Stork, P. J. S., Kornev, A. P., Taylor, S. S., and Shaw, A. S. (2013). Allosteric activation of functionally asymmetric RAF kinase dimers. *Cell 154*, 1036–1046.

(*339*) Kim, A., and Cohen, M. S. (2016). The discovery of Vemurafenib for the treatment of BRAF-mutated metastatic melanoma. *Expert Opinion on Drug Discovery 11*, 907–916.

(*340*) Chapman, P. B. et al. (2011). Improved Survival with Vemurafenib in Melanoma with BRAF V600E Mutation. *New England Journal of Medicine 364*, 2507–2516.

(*341*) Jambrina, P. G., Rauch, N., Pilkington, R., Rybakova, K., Nguyen, L. K., Kholodenko, B. N., Buchete, N.-V., Kolch, W., and Rosta, E. (2016). Phosphorylation of RAF Kinase Dimers Drives Conformational Changes that Facilitate Transactivation. *Angewandte Chemie (International Ed. in English) 55*, 983–986.

(*342*) Maloney, R. C., Zhang, M., Jang, H., and Nussinov, R. (2021). The mechanism of activation of monomeric B-Raf V600E. *Computational and Structural Biotechnology Journal 19*, 3349–3363.

(*343*) Rukhlenko, O. S. et al. (2018). Dissecting RAF Inhibitor Resistance by Structure-based Modeling Reveals Ways to Overcome Oncogenic RAS Signaling. *Cell Systems 7*, 161–179.e14.

(*344*) Jambrina, P. G., Bohuszewicz, O., Buchete, N.-V., Kolch, W., and Rosta, E. (2014). Molecular mechanisms of asymmetric RAF dimer activation. *Biochemical Society Transactions 42*, 784–790.

(*345*) Steinegger, M., and Söding, J. (2017). MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature Biotechnology 35*, 1026–1028.

(*346*) BLAST: Basic local alignment search tool `https://blast.ncbi.nlm.nih.gov/Blast.cgi` (accessed Dec. 20, 2024).

(*347*) BLAST® Command Line Applications User Manual `https://www.ncbi.nlm.nih.gov/books/NBK279684/` (accessed Dec. 19, 2024).

(*348*) Phillips, J. C., Braun, R., Wang, W., Gumbart, J., Tajkhorshid, E., Villa, E., Chipot, C., Skeel, R. D., Kalé, L., and Schulten, K. (2005). Scalable molecular dynamics with NAMD. *Journal of Computational Chemistry 26*, 1781–1802.

(*349*) Schneider, M. et al. (2022). Structure-Based and Knowledge-Informed Design of B-Raf Inhibitors Devoid of Deleterious PXR Binding. *Journal of Medicinal Chemistry 65*, 1552–1566.

(*350*) Cotto-Rios, X. M., Agianian, B., Gitego, N., Zacharioudakis, E., Giricz, O., Wu, Y., Zou, Y., Verma, A., Poulikakos, P. I., and Gavathiotis, E. (2020). Inhibitors of BRAF dimers using an allosteric site. *Nature Communications 11*, 4370.

(*351*) Zhang, C. et al. (2015). RAF inhibitors that evade paradoxical MAPK pathway activation. *Nature 526*, 583–586.

(*352*) Park, E., Rawson, S., Li, K., Kim, B.-W., Ficarro, S. B., Pino, G. G.-D., Sharif, H., Marto, J. A., Jeon, H., and Eck, M. J. (2019). Architecture of autoinhibited and active BRAF–MEK1–14-3-3 complexes. *Nature 575*, 545–550.

(*353*) Webb, B., and Sali, A. (2016). Comparative Protein Structure Modeling Using MODELLER. *Current Protocols in Bioinformatics 54*, 5.6.1–5.6.37.

(*354*) Hu, G., Katuwawala, A., Wang, K., Wu, Z., Ghadermarzi, S., Gao, J., and Kurgan, L. (2021). flDPnn: Accurate intrinsic disorder prediction with putative propensities of disorder functions. *Nature communications 12*, 1–8.

(*355*) Woo, P. C. Y. et al. (2012). Discovery of Seven Novel Mammalian and Avian Coronaviruses in the Genus Deltacoronavirus Supports Bat Coronaviruses as the Gene Source of Alphacoronavirus and Betacoronavirus and Avian Coronaviruses as the Gene Source of Gammacoronavirus and Deltacoronavirus. *Journal of Virology 86*, 3995–4008.

(*356*) Hilgenfeld, R., and Peiris, M. (2013). From SARS to MERS: 10 Years of Research on Highly Pathogenic Human Coronaviruses. *Antiviral Research 100*, 286–295.

(*357*) Gorbalenya, A. E. et al. (2020). The Species Severe Acute Respiratory Syndrome-Related Coronavirus: Classifying 2019-nCoV and Naming It SARS-CoV-2. *Nature Microbiology 5*, 536–544.

(*358*) Koonin, E. V., Krupovic, M., and Agol, V. I. (2021). The Baltimore Classification of Viruses 50 Years Later: How Does It Stand in the Light of Virus Evolution? *Microbiology and Molecular Biology Reviews 85*, e00053–21.

(*359*) Modrow, S., Falke, D., Truyen, U., and Schätzl, H. (2013). Viruses with single-stranded, positive-sense RNA genomes. *Molecular Virology*, 185–349.

(*360*) Yadav, R., Chaudhary, J. K., Jain, N., Chaudhary, P. K., Khanra, S., Dhamija, P., Sharma, A., Kumar, A., and Handu, S. (2021). Role of Structural and Non-Structural Proteins and Therapeutic Targets of SARS-CoV-2 for COVID-19. *Cells 10*, 821.

(*361*) Justo Arevalo, S., Castillo-Chávez, A., Uribe Calampa, C. S., Zapata Sifuentes, D., Huallpa, C. J., Landa Bianchi, G., Garavito-Salini Casas, R., Quiñones Aguilar, M., and Pineda Chavarría, R. (2023). What Do We Know about the Function of SARS-CoV-2 Proteins? *Frontiers in Immunology 14*, DOI: 10.3389/fimmu.2023.1249607.

(*362*) Jin, Z. et al. (2020). Structure of Mpro from SARS-CoV-2 and Discovery of Its Inhibitors. *Nature 582*, 289–293.

(*363*) Pang, X., Xu, W., Liu, Y., Li, H., and Chen, L. (2023). The Research Progress of SARS-CoV-2 Main Protease Inhibitors from 2020 to 2022. *European Journal of Medicinal Chemistry 257*, 115491.

(*364*) Chen, J. et al. (2020). Structural Basis for Helicase-Polymerase Coupling in the SARS-CoV-2 Replication-Transcription Complex. *Cell 182*, 1560–1573.e13.

(*365*) Kirchdoerfer, R. N., and Ward, A. B. (2019). Structure of the SARS-CoV Nsp12 Polymerase Bound to Nsp7 and Nsp8 Co-Factors. *Nature Communications 10*, 2342.

(*366*) Wu, J., and Gong, P. (2018). Visualizing the Nucleotide Addition Cycle of Viral RNA-Dependent RNA Polymerase. *Viruses 10*, 24.

(*367*) Yin, W. et al. (2020). Structural Basis for Inhibition of the RNA-dependent RNA Polymerase from SARS-CoV-2 by Remdesivir. *Science 368*, 1499–1504.

(*368*) Kokic, G., Hillen, H. S., Tegunov, D., Dienemann, C., Seitz, F., Schmitzova, J., Farnung, L., Siewert, A., Höbartner, C., and Cramer, P. (2021). Mechanism of SARS-CoV-2 Polymerase Stalling by Remdesivir. *Nature Communications 12*, 279.

(*369*) Malone, B. et al. (2021). Structural Basis for Backtracking by the SARS-CoV-2 Replication–Transcription Complex. *Proceedings of the National Academy of Sciences 118*, e2102516118.

(*370*) Jia, Z. et al. (2019). Delicate Structural Coordination of the Severe Acute Respiratory Syndrome Coronavirus Nsp13 upon ATP Hydrolysis. *Nucleic Acids Research 47*, 6538–6550.

(*371*) Tanner, J. A., Watt, R. M., Chai, Y.-B., Lu, L.-Y., Lin, M. C., Peiris, J. S. M., Poon, L. L. M., Kung, H.-F., and Huang, J.-D. (2003). The Severe Acute Respiratory Syndrome (SARS) Coronavirus NTPase/Helicase Belongs to a Distinct Class of 5' to 3' Viral Helicases. *The Journal of Biological Chemistry 278*, 39578–39582.

(*372*) Kwong, A. D., Rao, B. G., and Jeang, K.-T. (2005). Viral and Cellular RNA Helicases as Antiviral Targets. *Nature Reviews Drug Discovery 4*, 845–853.

(*373*) Tanner, J. A., Watt, R. M., Chai, Y.-B., Lu, L.-Y., Lin, M. C., Peiris, J., Poon, L. L., Kung, H.-F., and Huang, J.-D. (2003). The severe acute respiratory syndrome (SARS) coronavirus NTPase/helicase belongs to a distinct class of 5' to 3' viral helicases. *Journal of Biological Chemistry 278*, 39578–39582.

(*374*) Ivanov, K. A., Thiel, V., Dobbe, J. C., van der Meer, Y., Snijder, E. J., and Ziebuhr, J. (2004). Multiple enzymatic activities associated with severe acute respiratory syndrome coronavirus helicase. *Journal of Virology 78*, 5619–5632.

(*375*) Gorbalenya, A. E., and Koonin, E. V. (1993). Helicases: amino acid sequence comparisons and structure-function relationships. *Current Opinion in Structural Biology 3*, 419–429.

(*376*) Dittrich, M., and Schulten, K. (2006). PcrA Helicase, a Prototype ATP-Driven Molecular Motor. *Structure 14*, 1345–1353.

(*377*) Makurath, M. A., Whitley, K. D., Nguyen, B., Lohman, T. M., and Chemla, Y. R. (2019). Regulation of rep helicase unwinding by an auto-inhibitory subdomain. *Nucleic Acids Research 47*, 2523–2532.

(*378*) Toseland, C. P., and Webb, M. R. (2013). ATPase mechanism of the 5'-3' DNA helicase, RECD2. *Journal of Biological Chemistry 288*, 25183–25193.

(*379*) Raney, K. D., Byrd, A. K., and Aarattuthodiyil, S. In *DNA Helicases and DNA Motor Proteins*, Spies, M., Ed.; Springer: New York, NY, 2013, pp 17–46.

(*380*) Chen, J. et al. (2022). Ensemble cryo-em reveals conformational states of the nsp13 helicase in the SARS-COV-2 helicase replication–transcription complex. *Nature Structural and Molecular Biology 29*, 250–260.

(*381*) Malone, B. et al. (2021). Structural basis for backtracking by the SARS-CoV-2 replication and transcription complex. *Proceedings of the National Academy of Sciences 118*, e2102516118.

(*382*) Sonntag, K.-C., and Darai, G. (1995). Evolution of viral DNA-dependent RNA polymerases. *Virus Genes 11*, 271–284.

(*383*) Chen, J. et al. (2020). Structural basis for helicase-polymerase coupling in the SARS-COV-2 replication-transcription complex. *Cell 182*, DOI: `10.1016/j.cell.2020.07.033`.

(*384*) Brosh, R. M., Jr (2014). UvrD Helicase: The little engine that could. *Cell Cycle 13*, 1213–1215.

(*385*) White, M. A., Lin, W., and Cheng, X. (2020). Discovery of covid-19 inhibitors targeting the SARS-COV-2 NSP13 helicase. *The Journal of Physical Chemistry Letters 11*, 9144–9151.

(*386*) Hall, T. M. T. (2005). Multiple Modes of RNA Recognition by Zinc Finger Proteins. *Current Opinion in Structural Biology 15*, 367–373.

(*387*) Yu, J., Ha, T., and Schulten, K. (2006). Structure-based model of the stepping motor of PcrA Helicase. *Biophysical Journal 91*, 2097–2114.

(*388*) Marx, S. K. et al. (2023). Observing Inhibition of the SARS-CoV-2 Helicase at Single-Nucleotide Resolution. *Nucleic Acids Research 51*, 9266–9278.

(*389*) Singleton, M. R., Dillingham, M. S., and Wigley, D. B. (2007). Structure and Mechanism of Helicases and Nucleic Acid Translocases. *Annual Review of Biochemistry 76*, 23–50.

(*390*) Weber, R., and McCullagh, M. (2021). Role of ATP in the RNA Translocation Mechanism of SARS-CoV-2 NSP13 Helicase. *The Journal of Physical Chemistry B 125*, 8787–8796.

(*391*) Walker, J., Saraste, M., Runswick, M., and Gay, N. (1982). Distantly related sequences in the alpha- and beta-subunits of ATP synthase, myosin, kinases and other ATP-requiring enzymes and a common nucleotide binding fold. *The EMBO Journal 1*, 945–951.

(*392*) Saikrishnan, K., Powell, B., Cook, N. J., Webb, M. R., and Wigley, D. B. (2009). Mechanistic basis of 5'-3' translocation in SF1B helicases. *Cell 137*, 849–859.

(*393*) Newman, J. A. et al. (2021). Structure, mechanism and crystallographic fragment screening of the SARS-COV-2 NSP13 helicase. *Nature Communications 12*, DOI: 10.1038/s41467-021-25166-6.

(*394*) Mickolajczyk, K. J., Shelton, P. M., Grasso, M., Cao, X., Warrington, S. E., Aher, A., Liu, S., and Kapoor, T. M. (2021). Force-dependent stimulation of RNA unwinding by SARS-COV-2 and NSP13 helicase. *Biophysical Journal 120*, 1020–1030.

(*395*) Adedeji, A. O., Marchand, B., te Velthuis, A. J. W., Snijder, E. J., Weiss, S., Eoff, R. L., Singh, K., and Sarafianos, S. G. (2012). Mechanism of Nucleic Acid Unwinding by SARS-CoV Helicase. *PLOS ONE 7*, 1–11.

(*396*) Yue, K., Yao, B., Shi, Y., Yang, Y., Qian, Z., Ci, Y., and Shi, L. (2022). The stalk domain of SARS-COV-2 NSP13 is essential for its helicase activity. *Biochemical and Biophysical Research Communications 601*, 129–136.

(*397*) Huang, J., and MacKerell Jr, A. D. (2013). CHARMM36 all-atom additive protein force field: Validation based on comparison to NMR data. *Journal of computational chemistry 34*, 2135–2145.

(*398*) Case, D. A., Babin, V., Berryman, J., Betz, R., Cai, Q., Cerutti, D., Cheatham Iii, T., Darden, T., Duke, R., Gohlke, H. e., et al. (2014). Amber 14.

(*399*) Yu, Z., Li, P., and Merz, K. M. J. (2018). Extended Zinc AMBER Force Field (EZAFF). *Journal of Chemical Theory and Computation 14*, 242–254.

(*400*) Zgarbová, M., Otyepka, M., Šponer, J., Mládek, A., Banáš, P., Cheatham, T. E. I., and Jurečka, P. (2011). Refinement of the Cornell et al. Nucleic Acids Force Field Based on Reference Quantum Chemical Calculations of Glycosidic Torsion Profiles. *Journal of Chemical Theory and Computation 7*, 2886–2902.

(*401*) Berta, D., Badaoui, M., Martino, S. A., Buigues, P. J., Pisliakov, A. V., Elghobashi-Meinhardt, N., Wells, G., Harris, S. A., Frezza, E., and Rosta, E. (2021). Modelling the Active SARS-CoV-2 Helicase Complex as a Basis for Structure-Based Inhibitor Design. *Chemical Science 12*, 13492–13505.

(*402*) Pedregosa, F. et al. (2012). Scikit-learn: Machine Learning in Python. DOI: `10.48550/arXiv.1201.0490`.

(*403*) Halko, N., Martinsson, P.-G., and Tropp, J. A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions arXiv [math. NA], 2009.

(*404*) Smith, R. H. B., Dar, A. C., and Schlessinger, A. (2019). PyVOL: A PyMOL Plugin for Visualization, Comparison, and Volume Calculation of Drug-Binding Sites. 816702.

(*405*) Phillips, J. C., Braun, R., Wang, W., Gumbart, J., Tajkhorshid, E., Villa, E., Chipot, C., Skeel, R. D., Kale, L., and Schulten, K. (2005). Scalable molecular dynamics with NAMD. *Journal of computational chemistry 26*, 1781–1802.

(*406*) Fiorin, G., Klein, M. L., and Hénin, J. (2013). Using collective variables to drive molecular dynamics simulations. *Molecular Physics 111*, 3345–3362.

(*407*) Peters, M. B., Yang, Y., Wang, B., Fusti-Molnar, L., Weaver, M. N., and Merz Jr, K. M. (2010). Structural survey of zinc-containing proteins and development of the zinc AMBER force field (ZAFF). *Journal of chemical theory and computation 6*, 2935–2947.

(*408*) Newman, J. A., Douangamath, A., Yadzani, S., Yosaatmadja, Y., Aimon, A., Brandão-Neto, J., Dunnett, L., Gorrie-Stone, T., Skyner, R., Fearon, D., et al. (2021). Structure, mechanism and crystallographic fragment screening of the SARS-CoV-2 NSP13 helicase. *Nature communications 12*, 1–11.

(*409*) Jia, Z. et al. (2019). Delicate structural coordination of the Severe Acute Respiratory Syndrome coronavirus Nsp13 upon ATP hydrolysis. *Nucleic Acids Research 47*, 6538–6550.

(*410*) Okamoto, H., and Qiu, X. (2022). Detecting hierarchical organization of pervasive communities by modular decomposition of Markov chain. *Scientific Reports 12*, 20211.

(*411*) Von Hilgers, P., and Langville, A. N. In *Proceedings of the Markov Anniversary meeting*, 2006, pp 155–158.

(*412*) Sanderson, C., and Guenter, S. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, 2006, pp 482–491.

(*413*) Schbath, S. (2000). An overview on the distribution of word counts in Markov chains. *Journal of Computational Biology 7*, 193–201.

(*414*) Bianchini, M., Gori, M., and Scarselli, F. (2005). Inside pagerank. *ACM Transactions on Internet Technology (TOIT) 5*, 92–128.

(*415*) Berthiaux, H., and Mizonov, V. (2004). Applications of Markov chains in particulate process engineering: a review. *The Canadian journal of chemical engineering 82*, 1143–1168.

(*416*) Shukla, D., Hernández, C. X., Weber, J. K., and Pande, V. S. (2015). Markov State Models Provide Insights into Dynamic Modulation of Protein Function. *Accounts of Chemical Research 48*, 414–422.

(*417*) Swope, W. C., Pitera, J. W., and Suits, F. (2004). Describing Protein Folding Kinetics by Molecular Dynamics Simulations. 1. Theory. *The Journal of Physical Chemistry B 108*, 6571–6581.

(*418*) Singhal, N., Snow, C. D., and Pande, V. S. (2004). Using path sampling to build better Markovian state models: Predicting the folding rate and mechanism of a tryptophan zipper beta hairpin. *The Journal of Chemical Physics 121*, 415–425.

(*419*) Pan, A. C., and Roux, B. (2008). Building Markov state models along pathways to determine free energies and rates of transitions. *The Journal of chemical physics 129*.

(*420*) Martini, L., Kells, A., Covino, R., Hummer, G., Buchete, N.-V., and Rosta, E. (2017). Variational Identification of Markovian Transition States. *Phys. Rev. X 7*, 031060.

(*421*) Lane, T. J., Bowman, G. R., Beauchamp, K., Voelz, V. A., and Pande, V. S. (2011). Markov state model reveals folding and functional dynamics in ultra-long MD trajectories. *Journal of the American Chemical Society 133*, 18413–18419.

(*422*) Bowman, G. R., Bolin, E. R., Hart, K. M., Maguire, B. C., and Marqusee, S. (2015). Discovery of multiple hidden allosteric sites by combining Markov state models and experiments. *Proceedings of the National Academy of Sciences 112*, 2734–2739.

(*423*) Mardt, A., Pasquali, L., Wu, H., and Noé, F. (2018). VAMPnets for deep learning of molecular kinetics. *Nature Communications 9*, 4443.

(*424*) Jo, S., Suh, D., He, Z., Chipot, C., and Roux, B. (2016). Leveraging the information from Markov state models to improve the convergence of umbrella sampling simulations. *The journal of physical chemistry B 120*, 8733–8742.

(*425*) Husic, B. E., and Pande, V. S. (2018). Markov State Models: From an Art to a Science. *Journal of the American Chemical Society 140*, 2386–2396.

(*426*) Deuflhard, P., and Weber, M. (2005). Robust Perron cluster analysis in conformation dynamics. *Linear algebra and its applications 398*, 161–184.

(*427*) Hummer, G., and Szabo, A. (2015). Optimal dimensionality reduction of multistate kinetic and Markov-state models. *The Journal of Physical Chemistry B 119*, 9029–9037.

(*428*) Buchete, N.-V., and Hummer, G. (2008). Coarse master equations for peptide folding dynamics. *The Journal of Physical Chemistry B 112*, 6057–6069.

(*429*) Reuter, B., Fackeldey, K., and Weber, M. (2019). Generalized Markov modeling of nonreversible molecular kinetics. *The Journal of Chemical Physics 150*, 174103.

(*430*) Li, Y., and Dong, Z. (2016). Effect of clustering algorithm on establishing Markov state model for molecular dynamics simulations. *Journal of chemical information and modeling 56*, 1205–1215.

(*431*) Nagel, D., Sartore, S., and Stock, G. (2023). Toward a Benchmark for Markov State Models: The Folding of HP35. *The Journal of Physical Chemistry Letters 14*, 6956–6967.

(*432*) Röblitz, S., and Weber, M. (2013). Fuzzy spectral clustering by PCCA+: application to Markov state models and data classification. *Advances in Data Analysis and Classification 7*, 147–179.

(*433*) Barbu, and Zhu In *Proceedings Ninth IEEE International Conference on Computer Vision*, 2003, pp 320–327.

(*434*) Anderson, D. F., and Higham, D. J. (2012). Multilevel Monte Carlo for continuous time Markov chains, with applications in biochemical kinetics. *Multiscale Modeling & Simulation 10*, 146–179.

(*435*) Luo, Z. T., Sang, H., and Mallick, B. (2021). A Bayesian contiguous partitioning method for learning clustered latent variables. *The Journal of Machine Learning Research 22*, 1748–1799.

(*436*) Koskin, V., Kells, A., Clayton, J., Hartmann, A. K., Annibale, A., and Rosta, E. (2023). Variational kinetic clustering of complex networks. *The Journal of Chemical Physics 158*, 104112.

(*437*) Shaham, U., Stanton, K., Li, H., Nadler, B., Basri, R., and Kluger, Y. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.

(*438*) Kawamoto, T., Tsubaki, M., and Obuchi, T. (2018). Mean-field theory of graph neural networks in graph partitioning. *Advances in Neural Information Processing Systems 31*.

(*439*) Liao, R., Brockschmidt, M., Tarlow, D., Gaunt, A. L., Urtasun, R., and Zemel, R. (2018). Graph Partition Neural Networks for Semi-Supervised Classification. DOI: `10.48550/arXiv.1803.06272`.

(*440*) Nazi, A., Hang, W., Goldie, A., Ravi, S., and Mirhoseini, A. (2019). GAP: Generalizable Approximate Graph Partitioning Framework. DOI: `10.48550/arXiv.1903.00614`.

(*441*) Tsitsulin, A., Palowitch, J., Perozzi, B., and Müller, E. (2023). Graph clustering with graph neural networks. *Journal of Machine Learning Research 24*, 1–21.

(*442*) Singhal, N., and Pande, V. S. (2005). Error analysis and efficient sampling in Markovian state models for molecular dynamics. *The Journal of Chemical Physics 123*, 204909.

(*443*) Kube, S., and Weber, M. (2007). A coarse graining method for the identification of transition rates between molecular conformations. *The Journal of chemical physics 126*.

(*444*) Mor, B., Garhwal, S., and Kumar, A. (2021). A Systematic Review of Hidden Markov Models and Their Applications. *Archives of Computational Methods in Engineering 28*, 1429–1448.

(*445*) Eddy, S. R. (1998). Profile hidden Markov models. *Bioinformatics (Oxford, England) 14*, 755–763.

(*446*) Ullah, M., and Wolkenhauer, O. (2007). Family tree of Markov models in systems biology. *IET systems biology 1*, 247–254.

(*447*) Eddy, S. R. (1996). Hidden Markov models. *Current Opinion in Structural Biology 6*, 361–365.

(*448*) Kells, A., Koskin, V., Rosta, E., and Annibale, A. (2020). Correlation functions, mean first passage times, and the Kemeny constant. *The Journal of Chemical Physics 152*, 104108.

(*449*) Çınlar, E. In *Probability and Stochastics*, Çinlar, E., Ed.; Springer: New York, NY, 2011, pp 443–542.

(*450*) Newman, M. E. (2006). Modularity and community structure in networks. *Proceedings of the national academy of sciences 103*, 8577–8582.

(*451*) Guimera, R., Sales-Pardo, M., and Amaral, L. A. N. (2004). Modularity from fluctuations in random graphs and complex networks. *Physical Review E 70*, 025101.

(*452*) Good, B. H., De Montjoye, Y.-A., and Clauset, A. (2010). Performance of modularity maximization in practical contexts. *Physical review E 81*, 046106.

(*453*) Agarwal, G., and Kempe, D. (2008). Modularity-maximizing graph communities via mathematical programming. *The European Physical Journal B 66*, 409–418.

(*454*) Leighton, T., and Rao, S. (1999). Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM (JACM) 46*, 787–832.

(*455*) Ding, C. H., He, X., Zha, H., Gu, M., and Simon, H. D. In *Proceedings 2001 IEEE international conference on data mining*, 2001, pp 107–114.

(*456*) Hagen, L., and Kahng, A. In *1991 IEEE international conference on computer-aided design digest of technical papers*, 1991, pp 10–11.

(*457*) Even, G., Naor, J., Rao, S., and Schieber, B. (1999). Fast approximate graph partitioning algorithms. *SIAM Journal on Computing 28*, 2187–2214.

(*458*) Andreev, K., and Räcke, H. In *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, 2004, pp 120–124.

(*459*) Krauthgamer, R., Naor, J., and Schwartz, R. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, 2009, pp 942–949.

(*460*) Davies, D. L., and Bouldin, D. W. (1979). A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-1*, 224–227.

(*461*) Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research 12*, 2825–2830.

(*462*) PCA, scikit-learn `https://scikit-learn/stable/modules/generated/sklearn.decomposition.PCA.html` (accessed Dec. 11, 2024).

(*463*) Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems 26*.

(*464*) Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems 32*.

(*465*) He, K., Zhang, X., Ren, S., and Sun, J. In *Proceedings of the IEEE international conference on computer vision*, 2015, pp 1026–1034.

(*466*) Glorot, X., and Bengio, Y. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp 249–256.

(*467*) Abbe, E. (2018). Community detection and stochastic block models: recent developments. *Journal of Machine Learning Research 18*, 1–86.

(*468*) Scherer, M. K., Trendelkamp-Schroer, B., Paul, F., Pérez-Hernández, G., Hoffmann, M., Plattner, N., Wehmeyer, C., Prinz, J.-H., and Noé, F. (2015). PyEMMA 2: A software package for estimation, validation, and analysis of Markov models. *Journal of chemical theory and computation 11*, 5525–5542.

(*469*) Schwantes, C. R., and Pande, V. S. (2013). Improvements in Markov state model construction reveal many non-native interactions in the folding of NTL9. *Journal of chemical theory and computation 9*, 2000–2009.

(*470*) Pérez-Hernández, G., Paul, F., Giorgino, T., De Fabritiis, G., and Noé, F. (2013). Identification of slow molecular order parameters for Markov model construction. *The Journal of chemical physics 139*.

(*471*) Brodersen, P. J. N. (2023). Netgraph: Publication-quality Network Visualisations in Python. *Journal of Open Source Software 8*, 5372.

(*472*) Muratov, E. N. et al. (2020). QSAR without Borders. *Chemical Society Reviews 49*, 3525–3564.

(*473*) Benet, L. Z., Hosey, C. M., Ursu, O., and Oprea, T. I. (2016). BDDCS, the Rule of 5 and Drugability. *Advanced drug delivery reviews 101*, 89–98.

(*474*) Kralj, S., Jukič, M., and Bren, U. (2023). Molecular Filters in Medicinal Chemistry. *Encyclopedia 3*, 501–511.

(*475*) Kirkpatrick, P., and Ellis, C. (2004). Chemical Space. *Nature 432*, 823–823.

(*476*) Weininger, D. (1988). SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules. *Journal of Chemical Information and Computer Sciences 28*, 31–36.

(*477*) SMILES | Chemaxon Docs `https://docs.chemaxon.com/display/docs/formats_smiles.md` (accessed Nov. 20, 2024).

(*478*) Daylight Theory: SMARTS - A Language for Describing Molecular Patterns `https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html` (accessed Nov. 20, 2024).

(*479*) Yang, J. J., Ursu, O., Lipinski, C. A., Sklar, L. A., Oprea, T. I., and Bologa, C. G. (2016). Badapple: Promiscuity Patterns from Noisy Evidence. *Journal of Cheminformatics 8*, 29.

(*480*) Baell, J. B., and Holloway, G. A. (2010). New Substructure Filters for Removal of Pan Assay Interference Compounds (PAINS) from Screening Libraries and for Their Exclusion in Bioassays. *Journal of Medicinal Chemistry 53*, 2719–2740.

(*481*) Wigh, D. S., Goodman, J. M., and Lapkin, A. A. (2022). A Review of Molecular Representation in the Age of Machine Learning. *WIREs Computational Molecular Science 12*, e1603.

(*482*) Boldini, D., Ballabio, D., Consonni, V., Todeschini, R., Grisoni, F., and Sieber, S. A. (2024). Effectiveness of Molecular Fingerprints for Exploring the Chemical Space of Natural Products. *Journal of Cheminformatics 16*, 35.

(*483*) Cereto-Massagué, A., Ojeda, M. J., Valls, C., Mulero, M., Garcia-Vallvé, S., and Pujadas, G. (2015). Molecular Fingerprint Similarity Search in Virtual Screening. *Methods 71*, 58–63.

(*484*) MACCS Key 44 `http://www.dalkescientific.com/writings/diary/archive/2014/10/17/maccs%5C_key%5C_44.html` (accessed Nov. 20, 2024).

(*485*) Durant, J. L., Leland, B. A., Henry, D. R., and Nourse, J. G. (2002). Reoptimization of MDL Keys for Use in Drug Discovery. *Journal of Chemical Information and Computer Sciences 42*, 1273–1280.

(*486*) Carhart, R. E., Smith, D. H., and Venkataraghavan, R. (1985). Atom Pairs as Molecular Features in Structure-Activity Studies: Definition and Applications. *Journal of Chemical Information and Computer Sciences 25*, 64–73.

(*487*) Kahn, I., Lomaka, A., and Karelson, M. (2014). Topological Fingerprints as an Aid in Finding Structural Patterns for LRRK2 Inhibition. *Molecular Informatics 33*, 269–275.

(*488*) Fingerprint Generation Toolkits in Python `https://docs.eyesopen.com/toolkits/python/graphsimtk/fingerprint.html%5C#section-fingerprint-path` (accessed Nov. 20, 2024).

(*489*) Morgan, H. L. (1965). The Generation of a Unique Machine Description for Chemical Structures-A Technique Developed at Chemical Abstracts Service. *Journal of Chemical Documentation 5*, 107–113.

(*490*) Rogers, D., and Hahn, M. (2010). Extended-Connectivity Fingerprints. *Journal of Chemical Information and Modeling 50*, 742–754.

(*491*) Bero, S. A., Muda, A. K., Choo, Y. H., Muda, N. A., and Pratama, S. F. (2017). Similarity Measure for Molecular Structure: A Brief Review. *Journal of Physics: Conference Series 892*, 012015.

(*492*) Willett, P. (2009). Similarity Methods in Chemoinformatics. *Annual Review of Information Science and Technology 43*, 3–71.

(*493*) Maeda, M. H. (2015). Current Challenges in Development of a Database of Three-Dimensional Chemical Structures. *Frontiers in Bioengineering and Biotechnology 3*, 66.

(*494*) Maeda, M. H., and Kondo, K. (2013). Three-Dimensional Structure Database of Natural Metabolites (3DMET): A Novel Database of Curated 3D Structures. *Journal of Chemical Information and Modeling 53*, 527–533.

(*495*) Chiarparin, E., Packer, M. J., and Wilson, D. M. (2019). Experimental Free Ligand Conformations: A Missing Link in Structure-Based Drug Discovery. *Future Medicinal Chemistry 11*, 79–82.

(*496*) McNutt, A. T., Bisiriyu, F., Song, S., Vyas, A., Hutchison, G. R., and Koes, D. R. (2023). Conformer Generation for Structure-Based Drug Design: How Many and How Good? *Journal of Chemical Information and Modeling 63*, 6598–6607.

(*497*) Jain, A. N., Brueckner, A. C., Cleves, A. E., Reibarkh, M., and Sherer, E. C. (2023). A Distributional Model of Bound Ligand Conformational Strain: From Small Molecules up to Large Peptidic Macrocycles. *Journal of Medicinal Chemistry 66*, 1955–1971.

(*498*) Maeda, M. H., Yonezawa, T., and Komaba, T. (2018). Chemical Curation to Improve Data Accuracy: Recent Development of the 3DMET Database. *Biophysics and Physicobiology 15*, 87–93.

(*499*) Foloppe, N., and Chen, I.-J. (2009). Conformational Sampling and Energetics of Drug-like Molecules. *Current Medicinal Chemistry 16*, 3381–3413.

(*500*) Sadowski, J., and Gasteiger, J. (1993). From Atoms and Bonds to Three-Dimensional Atomic Coordinates: Automatic Model Builders. *Chemical Reviews 93*, 2567–2581.

(*501*) Gasteiger, J., Rudolph, C., and Sadowski, J. (1990). Automatic Generation of 3D-atomic Coordinates for Organic Molecules. *Tetrahedron Computer Methodology 3*, 537–547.

(*502*) Rusinko, A. I., Sheridan, R. P., Nilakantan, R., Haraki, K. S., Bauman, N., and Venkataraghavan, R. (1989). Using CONCORD to Construct a Large Database of Three-Dimensional Coordinates from Connection Tables. *Journal of Chemical Information and Computer Sciences 29*, 251–255.

(*503*) Groom, C. R., Bruno, I. J., Lightfoot, M. P., and Ward, S. C. (2016). The Cambridge Structural Database. *Acta Crystallographica Section B, Structural Science, Crystal Engineering and Materials 72*, 171–179.

(*504*) Chen, J. H., Linstead, E., Swamidass, S. J., Wang, D., and Baldi, P. (2007). ChemDB Update—Full-Text Search and Virtual Chemical Space. *Bioinformatics 23*, 2348–2351.

(*505*) Gražulis, S., Daškevič, A., Merkys, A., Chateigner, D., Lutterotti, L., Quirós, M., Serebryanaya, N. R., Moeck, P., Downs, R. T., and Le Bail, A. (2012). Crystallography Open Database (COD): An Open-Access Collection of Crystal Structures and Platform for World-Wide Collaboration. *Nucleic Acids Research 40*, D420–427.

(*506*) Yoshikawa, N., and Hutchison, G. R. (2019). Fast, Efficient Fragment-Based Coordinate Generation for Open Babel. *Journal of Cheminformatics 11*, 49.

(*507*) Kothiwale, S., Mendenhall, J. L., and Meiler, J. (2015). BCL::Conf: Small Molecule Conformational Sampling Using a Knowledge Based Rotamer Library. *Journal of Cheminformatics 7*, 47.

(*508*) Hawkins, P. C. D., Skillman, A. G., Warren, G. L., Ellingson, B. A., and Stahl, M. T. (2010). Conformer Generation with OMEGA: Algorithm and Validation Using High Quality Structures from the Protein Databank and Cambridge Structural Database. *Journal of Chemical Information and Modeling 50*, 572–584.

(*509*) Leach, A. R., Prout, K., and Dolata, D. P. (1988). An Investigation into the Construction of Molecular Models by the Template Joining Method. *Journal of Computer-Aided Molecular Design 2*, 107–123.

(*510*) Havel, T. F., Kuntz, I. D., and Crippen, G. M. (1983). The Theory and Practice of Distance Geometry. *Bulletin of Mathematical Biology 45*, 665–720.

(*511*) Habgood, M., James, T., and Heifetz, A. In *Quantum Mechanics in Drug Discovery*, Heifetz, A., Ed.; Springer US: New York, NY, 2020, pp 207–229.

(*512*) Axelrod, S., and Gómez-Bombarelli, R. (2022). GEOM, Energy-Annotated Molecular Conformations for Property Prediction and Molecular Generation. *Scientific Data 9*, 185.

(*513*) Pracht, P., Bohle, F., and Grimme, S. (2020). Automated Exploration of the Low-Energy Chemical Space with Fast Quantum Chemical Methods. *Physical Chemistry Chemical Physics 22*, 7169–7192.

(*514*) Lam, C. C., and Goodman, J. M. (2023). CONFPASS: Fast DFT Re-Optimizations of Structures from Conformation Searches. *Journal of Chemical Information and Modeling 63*, 4364–4375.

(*515*) Żurański, A. M., Wang, J. Y., Shields, B. J., and Doyle, A. G. (2022). Auto-QChem: An Automated Workflow for the Generation and Storage of DFT Calculations for Organic Molecules. *Reaction Chemistry & Engineering 7*, 1276–1284.

(*516*) Lewis-Atwell, T., Townsend, P. A., and Grayson, M. N. (2021). Comparisons of Different Force Fields in Conformational Analysis and Searching of Organic Molecules: A Review. *Tetrahedron 79*, 131865.

(*517*) Allinger, N. L., Chen, K., and Lii, J.-H. (1996). An Improved Force Field (MM4) for Saturated Hydrocarbons. *Journal of Computational Chemistry 17*, 642–668.

(*518*) Allinger, N. L. (1977). Conformational Analysis. 130. MM2. A Hydrocarbon Force Field Utilizing V1 and V2 Torsional Terms. *Journal of the American Chemical Society 99*, 8127–8134.

(*519*) Tronchet, J. M. J., and Komaromi, I. (1993). Molecular Mechanical Parameters (MM2 Force Field) for the N(Sp3)-O(Sp3) Bond. *International Journal of Biological Macromolecules 15*, 69–72.

(*520*) Halgren, T. A. (1996). Merck Molecular Force Field. I. Basis, Form, Scope, Parameterization, and Performance of MMFF94. *Journal of Computational Chemistry 17*, 490–519.

(*521*) Jorgensen, W. L., Maxwell, D. S., and Tirado-Rives, J. (1996). Development and Testing of the OPLS All-Atom Force Field on Conformational Energetics and Properties of Organic Liquids. *Journal of the American Chemical Society 118*, 11225–11236.

(*522*) Jász, Á., Rák, Á., Ladjánszki, I., and Cserey, G. (2019). Optimized GPU Implementation of Merck Molecular Force Field and Universal Force Field. *Journal of Molecular Structure 1188*, 227–233.

(*523*) Hopkins, E., Sanvictores, T., and Sharma, S. In *StatPearls*; StatPearls Publishing: Treasure Island (FL), 2024.

(*524*) Bochevarov, A. D., Watson, M. A., Greenwood, J. R., and Philipp, D. M. (2016). Multiconformation, Density Functional Theory-Based pKa Prediction in Application to Large, Flexible Organic Molecules with Diverse Functional Groups. *Journal of Chemical Theory and Computation 12*, 6001–6019.

(*525*) Klicić, J. J., Friesner, R. A., Liu, S.-Y., and Guida, W. C. (2002). Accurate Prediction of Acidity Constants in Aqueous Solution via Density Functional Theory and Self-Consistent Reaction Field Methods. *The Journal of Physical Chemistry A 106*, 1327–1335.

(*526*) Perrin, D. D., Dempsey, B., and Serjeant, E. P. In *pKa Prediction for Organic Acids and Bases*, Perrin, D. D., Dempsey, B., and Serjeant, E. P., Eds.; Springer Netherlands: Dordrecht, 1981, pp 21–26.

(*527*) Shelley, J. C., Cholleti, A., Frye, L. L., Greenwood, J. R., Timlin, M. R., and Uchimaya, M. (2007). Epik: A Software Program for pKaprediction and Protonation State Generation for Drug-like Molecules. *Journal of Computer-Aided Molecular Design 21*, 681–691.

(*528*) Watts, K. S., Dalal, P., Murphy, R. B., Sherman, W., Friesner, R. A., and Shelley, J. C. (2010). ConfGen: A Conformational Search Method for Efficient Generation of Bioactive Conformers. *Journal of Chemical Information and Modeling 50*, 534–546.

(*529*) Johnston, R. C. et al. (2023). Epik: pKa and Protonation State Prediction through Machine Learning. *Journal of Chemical Theory and Computation 19*, 2380–2388.

(*530*) Luo, Y., Yan, K., and Ji, S. (2021). GraphDF: A Discrete Flow Model for Molecular Graph Generation. DOI: `10.48550/arXiv.2102.01189`.

(*531*) Olivecrona, M., Blaschke, T., Engkvist, O., and Chen, H. (2017). Molecular De Novo Design through Deep Reinforcement Learning. DOI: `10.48550/arXiv.1704.07555`.

(*532*) Podda, M., Bacciu, D., and Micheli, A. In *International Conference on Artificial Intelligence and Statistics*, PMLR: 2020, pp 2240–2250.

(*533*) Powers, A. S., Yu, H. H., Suriana, P., Koodli, R. V., Lu, T., Paggi, J. M., and Dror, R. O. (2023). Geometric Deep Learning for Structure-Based Ligand Design. *ACS Central Science 9*, 2257–2267.

(*534*) Alakhdar, A., Poczos, B., and Washburn, N. (2024). Diffusion Models in De Novo Drug Design. *Journal of Chemical Information and Modeling 64*, 7238–7256.

(*535*) Li, M., Zhang, H., Chen, B., Wu, Y., and Guan, L. (2018). Prediction of pKa Values for Neutral and Basic Drugs Based on Hybrid Artificial Intelligence Methods. *Scientific Reports 8*, 3991.

(*536*) Pan, X., Wang, H., Li, C., Zhang, J. Z. H., and Ji, C. (2021). MolGpka: A Web Server for Small Molecule pKa Prediction Using a Graph-Convolutional Neural Network. *Journal of Chemical Information and Modeling 61*, 3159–3165.

(*537*) Unke, O. T., Chmiela, S., Sauceda, H. E., Gastegger, M., Poltavsky, I., Schütt, K. T., Tkatchenko, A., and Müller, K.-R. (2021). Machine Learning Force Fields. *Chemical Reviews 121*, 10142–10186.

(*538*) Brooijmans, N., and Kuntz, I. D. (2003). Molecular Recognition and Docking Algorithms. *Annual Review of Biophysics 32*, 335–373.

(*539*) Wodak, S. J., De Crombrugghe, M., and Janin, J. (1987). Computer Studies of Interactions between Macromolecules. *Progress in Biophysics and Molecular Biology 49*, 29–63.

(*540*) Menchaca, T. M., Juárez-Portilla, C., Zepeda, R. C., Menchaca, T. M., Juárez-Portilla, C., and Zepeda, R. C. In *Drug Discovery and Development - New Advances*; IntechOpen: 2020.

(*541*) Paggi, J. M., Pandit, A., and Dror, R. O. (2024). The Art and Science of Molecular Docking. *Annual Review of Biochemistry 93*, 389–410.

(*542*) Torres, P. H. M., Sodero, A. C. R., Jofily, P., and Silva-Jr, F. P. (2019). Key Topics in Molecular Docking for Drug Design. *International Journal of Molecular Sciences 20*, 4574.

(*543*) Zev, S., Raz, K., Schwartz, R., Tarabeh, R., Gupta, P. K., and Major, D. T. (2021). Benchmarking the Ability of Common Docking Programs to Correctly Reproduce and Score Binding Modes in SARS-CoV-2 Protease Mpro. *Journal of Chemical Information and Modeling 61*, 2957–2966.

(*544*) Huang, N., Shoichet, B. K., and Irwin, J. J. (2006). Benchmarking Sets for Molecular Docking. *Journal of Medicinal Chemistry 49*, 6789–6801.

(*545*) Cross, J. B., Thompson, D. C., Rai, B. K., Baber, J. C., Fan, K. Y., Hu, Y., and Humblet, C. (2009). Comparison of Several Molecular Docking Programs: Pose Prediction and Virtual Screening Accuracy. *Journal of Chemical Information and Modeling 49*, 1455–1474.

(*546*) Wang, Z., Sun, H., Yao, X., Li, D., Xu, L., Li, Y., Tian, S., and Hou, T. (2016). Comprehensive Evaluation of Ten Docking Programs on a Diverse Set of Protein–Ligand Complexes: The Prediction Accuracy of Sampling Power and Scoring Power. *Physical Chemistry Chemical Physics 18*, 12964–12975.

(*547*) Cole, J. C., Murray, C. W., Nissink, J. W. M., Taylor, R. D., and Taylor, R. (2005). Comparing Protein–Ligand Docking Programs Is Difficult. *Proteins: Structure, Function, and Bioinformatics 60*, 325–332.

(*548*) Poli, G., Martinelli, A., and Tuccinardi, T. (2016). Reliability Analysis and Optimization of the Consensus Docking Approach for the Development of Virtual Screening Studies. *Journal of Enzyme Inhibition and Medicinal Chemistry 31*, 167–173.

(*549*) Tuccinardi, T., Poli, G., Romboli, V., Giordano, A., and Martinelli, A. (2014). Extensive Consensus Docking Evaluation for Ligand Pose Prediction and Virtual Screening Studies. *Journal of Chemical Information and Modeling 54*, 2980–2986.

(*550*) Kuntz, I. D., Blaney, J. M., Oatley, S. J., Langridge, R., and Ferrin, T. E. (1982). A Geometric Approach to Macromolecule-Ligand Interactions. *Journal of Molecular Biology 161*, 269–288.

(*551*) Ferro, D. R., and Hermans, J. (1977). A Different Best Rigid-Body Molecular Fit Routine. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography 33*, 345–347.

(*552*) Totrov, M., and Abagyan, R. (1994). Detailed Ab Initio Prediction of Lysozyme–Antibody Complex with 1.6 Å Accuracy. *Nature Structural Biology 1*, 259–263.

(*553*) Jorgensen, W. L. (1991). Rusting of the Lock and Key Model for Protein-Ligand Binding. *Science 254*, 954–955.

(*554*) Miteva, M. A., Lee, W. H., Montes, M. O., and Villoutreix, B. O. (2005). Fast Structure-Based Virtual Ligand Screening Combining FRED, DOCK, and Surflex. *Journal of Medicinal Chemistry 48*, 6012–6022.

(*555*) Sauton, N., Lagorce, D., Villoutreix, B. O., and Miteva, M. A. (2008). MS-DOCK: Accurate Multiple Conformation Generator and Rigid Docking Protocol for Multi-Step Virtual Ligand Screening. *BMC Bioinformatics 9*, 184.

(*556*) Koshland Jr., D. E. (1995). The Key–Lock Theory and the Induced Fit Theory. *Angewandte Chemie International Edition in English 33*, 2375–2378.

(*557*) Rosenfeld, R., Vajda, S., and DeLisi, C. (1995). Flexible Docking and Design. *Annual Review of Biophysics 24*, 677–700.

(*558*) Teodoro, M. L., and Kavraki, L. E. Conformational Flexibility Models for the Receptor in Structure Based Drug Design. *Current Pharmaceutical Design 9*, 1635–1648.

(*559*) Sousa, S. F., Ribeiro, A. J. M., Coimbra, J. T. S., Neves, R. P. P., Martins, S. A., Moorthy, N. S. H. N., Fernandes, P. A., and Ramos, M. J. Protein-Ligand Docking in the New Millennium – A Retrospective of 10 Years in the Field. *Current Medicinal Chemistry 20*, 2296–2314.

(*560*) Wong, C. F. (2015). Flexible Receptor Docking for Drug Discovery. *Expert Opinion on Drug Discovery 10*, 1189–1200.

(*561*) Barril, X., and Morley, S. D. (2005). Unveiling the Full Potential of Flexible Receptor Docking Using Multiple Crystallographic Structures. *Journal of Medicinal Chemistry 48*, 4432–4443.

(*562*) Liu, M., and Wang, S. (1999). MCDOCK: A Monte Carlo Simulation Approach to the Molecular Docking Problem. *Journal of Computer-Aided Molecular Design 13*, 435–451.

(*563*) Zhang, W., Bell, E. W., Yin, M., and Zhang, Y. (2020). EDock: Blind Protein–Ligand Docking by Replica-Exchange Monte Carlo Simulation. *Journal of Cheminformatics 12*, 37.

(*564*) Jones, G., Willett, P., Glen, R. C., Leach, A. R., and Taylor, R. (1997). Development and Validation of a Genetic Algorithm for Flexible Docking1. *Journal of Molecular Biology 267*, 727–748.

(*565*) Flachsenberg, F., Meyder, A., Sommer, K., Penner, P., and Rarey, M. (2020). A Consistent Scheme for Gradient-Based Optimization of Protein-Ligand Poses. *Journal of Chemical Information and Modeling 60*, 6502–6522.

(*566*) Vanommeslaeghe, K. et al. (2010). CHARMM General Force Field: A Force Field for Drug-like Molecules Compatible with the CHARMM All-Atom Additive Biological Force Fields. *Journal of Computational Chemistry 31*, 671–690.

(*567*) Ewing, T. J., Makino, S., Skillman, A. G., and Kuntz, I. D. (2001). DOCK 4.0: Search Strategies for Automated Molecular Docking of Flexible Molecule Databases. *Journal of Computer-Aided Molecular Design 15*, 411–428.

(*568*) Park, H., Zhou, G., Baek, M., Baker, D., and DiMaio, F. (2021). Force Field Optimization Guided by Small Molecule Crystal Lattice Data Enables Consistent Sub-Angstrom Protein–Ligand Docking. *Journal of Chemical Theory and Computation 17*, 2000–2010.

(*569*) Jakalian, A., Jack, D. B., and Bayly, C. I. (2002). Fast, efficient generation of high-quality atomic charges. AM1-BCC model: II. Parameterization and validation. *Journal of Computational Chemistry 23*, 1623–1641.

(*570*) Gasteiger, J., and Marsili, M. (1978). A New Model for Calculating Atomic Charges in Molecules. *Tetrahedron Letters 19*, 3181–3184.

(*571*) Gasteiger, J., and Marsili, M. (1980). Iterative Partial Equalization of Orbital Electronegativity—a Rapid Access to Atomic Charges. *Tetrahedron 36*, 3219–3228.

(*572*) Dewar, M. J. S., Zoebisch, E. G., Healy, E. F., and Stewart, J. J. P. (1985). Development and Use of Quantum Mechanical Molecular Models. 76. AM1: A New General Purpose Quantum Mechanical Molecular Model. *Journal of the American Chemical Society 107*, 3902–3909.

(*573*) Hou, X., Du, J., Zhang, J., Du, L., Fang, H., and Li, M. (2013). How to Improve Docking Accuracy of AutoDock4.2: A Case Study Using Different Electrostatic Potentials. *Journal of Chemical Information and Modeling 53*, 188–200.

(*574*) Sippl, M. J. (1990). Calculation of Conformational Ensembles from Potentials of Mean Force. An Approach to the Knowledge-Based Prediction of Local Structures in Globular Proteins. *Journal of Molecular Biology 213*, 859–883.

(*575*) Thomas, P. D., and Dill, K. A. (1996). Statistical Potentials Extracted From Protein Structures: How Accurate Are They? *Journal of Molecular Biology 257*, 457–469.

(*576*) Sippl, M. J. (1995). Knowledge-Based Potentials for Proteins. *Current Opinion in Structural Biology 5*, 229–235.

(*577*) Huang, S.-Y., and Zou, X. (2010). Inclusion of Solvation and Entropy in the Knowledge-Based Scoring Function for Protein-Ligand Interactions. *Journal of Chemical Information and Modeling 50*, 262–273.

(*578*) Muegge, I., and Martin, Y. C. (1999). A General and Fast Scoring Function for Protein-Ligand Interactions: A Simplified Potential Approach. *Journal of Medicinal Chemistry 42*, 791–804.

(*579*) Kramer, C., and Gedeck, P. (2010). Leave-Cluster-out Cross-Validation Is Appropriate for Scoring Functions Derived from Diverse Protein Data Sets. *Journal of Chemical Information and Modeling 50*, 1961–1969.

(*580*) Huang, S.-Y., and Zou, X. (2006). An iterative knowledge-based scoring function to predict protein–ligand interactions: II. Validation of the scoring function. *Journal of Computational Chemistry 27*, 1876–1882.

(*581*) Cheng, T., Li, X., Li, Y., Liu, Z., and Wang, R. (2009). Comparative Assessment of Scoring Functions on a Diverse Test Set. *Journal of Chemical Information and Modeling 49*, 1079–1093.

(*582*) Muegge, I. (2006). PMF Scoring Revisited. *Journal of Medicinal Chemistry 49*, 5895–5902.

(*583*) Guedes, I. A., Pereira, F. S. S., and Dardenne, L. E. (2018). Empirical Scoring Functions for Structure-Based Virtual Screening: Applications, Critical Aspects, and Challenges. *Frontiers in Pharmacology 9*, 1089.

(*584*) Hu, L., Benson, M. L., Smith, R. D., Lerner, M. G., and Carlson, H. A. (2005). Binding MOAD (Mother Of All Databases). *Proteins 60*, 333–340.

(*585*) Wang, R., Fang, X., Lu, Y., and Wang, S. (2004). The PDBbind Database: Collection of Binding Affinities for Protein-Ligand Complexes with Known Three-Dimensional Structures. *Journal of Medicinal Chemistry 47*, 2977–2980.

(*586*) Pason, L. P., and Sotriffer, C. A. (2016). Empirical Scoring Functions for Affinity Prediction of Protein-ligand Complexes. *Molecular Informatics 35*, 541–548.

(*587*) Morris, G. M., Huey, R., Lindstrom, W., Sanner, M. F., Belew, R. K., Goodsell, D. S., and Olson, A. J. (2009). AutoDock4 and AutoDockTools4: Automated Docking with Selective Receptor Flexibility. *Journal of computational chemistry 30*, 2785–2791.

(*588*) Huey, R., Morris, G. M., Olson, A. J., and Goodsell, D. S. (2007). A Semiempirical Free Energy Force Field with Charge-Based Desolvation. *Journal of Computational Chemistry 28*, 1145–1152.

(*589*) Weiner, S. J., Kollman, P. A., Case, D. A., Singh, U. C., Ghio, C., Alagona, G., Profeta, S., and Weiner, P. (1984). A New Force Field for Molecular Mechanical Simulation of Nucleic Acids and Proteins. *Journal of the American Chemical Society 106*, 765–784.

(*590*) Mehler, E., and Solmajer, T. (1991). Electrostatic Effects in Proteins: Comparison of Dielectric and Charge Models. *Protein Engineering, Design and Selection 4*, 903–910.

(*591*) Wade, R. C., Clark, K. J., and Goodford, P. J. (1993). Further Development of Hydrogen Bond Functions for Use in Determining Energetically Favorable Binding Sites on Molecules of Known Structure. 1. Ligand Probe Groups with the Ability to Form Two Hydrogen Bonds. *Journal of Medicinal Chemistry 36*, 140–147.

(*592*) Wesson, L., and Eisenberg, D. (1992). Atomic Solvation Parameters Applied to Molecular Dynamics of Proteins in Solution. *Protein Science: A Publication of the Protein Society 1*, 227–235.

(*593*) AutoDock4/AD4_parameters.Dat at Master · Ccsb-Scripps/AutoDock4 `https://github.com/ccsb-scripps/AutoDock4/blob/master/AD4%5C_parameters.dat` (accessed Nov. 20, 2024).

(*594*) Solis, F. J., and Wets, R. J.-B. (1981). Minimization by Random Search Techniques. *Mathematics of Operations Research 6*, 19–30.

(*595*) Santos-Martins, D., Solis-Vasquez, L., Tillack, A. F., Sanner, M. F., Koch, A., and Forli, S. (2021). Accelerating AutoDock4 with GPUs and Gradient-Based Local Search. *Journal of Chemical Theory and Computation 17*, 1060–1073.

(*596*) Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method. DOI: `10.48550/arXiv.1212.5701`.

(*597*) Eberhardt, J., Santos-Martins, D., Tillack, A. F., and Forli, S. (2021). AutoDock Vina 1.2.0: New Docking Methods, Expanded Force Field, and Python Bindings. *Journal of Chemical Information and Modeling 61*, 3891–3898.

(*598*) Trott, O., and Olson, A. J. (2010). AutoDock Vina: Improving the Speed and Accuracy of Docking with a New Scoring Function, Efficient Optimization and Multithreading. *Journal of computational chemistry 31*, 455–461.

(*599*) Abagyan, R., Totrov, M., and Kuznetsov, D. (1994). ICM—A New Method for Protein Modeling and Design: Applications to Docking and Structure Prediction from the Distorted Native Conformation. *Journal of Computational Chemistry 15*, 488–506.

(*600*) *Numerical Optimization*; Springer Series in Operations Research and Financial Engineering; Springer New York: 2006.

(*601*) Nguyen, N. T., Nguyen, T. H., Pham, T. N. H., Huy, N. T., Bay, M. V., Pham, M. Q., Nam, P. C., Vu, V. V., and Ngo, S. T. (2020). Autodock Vina Adopts More Accurate Binding Poses but Autodock4 Forms Better Binding Affinity. *Journal of Chemical Information and Modeling 60*, 204–211.

(*602*) Chen, Y.-C. (2015). Beware of Docking! *Trends in Pharmacological Sciences 36*, 78–95.

(*603*) Li, H., Peng, J., Sidorov, P., Leung, Y., Leung, K.-S., Wong, M.-H., Lu, G., and Ballester, P. J. (2019). Classical Scoring Functions for Docking Are Unable to Exploit Large Volumes of Structural and Interaction Data. *Bioinformatics (Oxford, England) 35*, 3989–3995.

(*604*) Chodera, J. D., Mobley, D. L., Shirts, M. R., Dixon, R. W., Branson, K., and Pande, V. S. (2011). Alchemical Free Energy Methods for Drug Discovery: Progress and Challenges. *Current opinion in structural biology 21*, 150–160.

(*605*) York, D. M. (2023). Modern Alchemical Free Energy Methods for Drug Discovery Explained. *ACS Physical Chemistry Au 3*, 478–491.

(*606*) Zwanzig, R. W. (1954). High-Temperature Equation of State by a Perturbation Method. I. Nonpolar Gases. *The Journal of Chemical Physics 22*, 1420–1426.

(*607*) Kirkwood, J. G. (1935). Statistical Mechanics of Fluid Mixtures. *The Journal of Chemical Physics 3*, 300–313.

(*608*) Straatsma, T. P., and Berendsen, H. J. C. (1988). Free Energy of Ionic Hydration: Analysis of a Thermodynamic Integration Technique to Evaluate Free Energy Differences by Molecular Dynamics Simulations. *The Journal of Chemical Physics 89*, 5876–5886.

(*609*) Jarzynski, C. (1997). Nonequilibrium Equality for Free Energy Differences. *Physical Review Letters 78*, 2690–2693.

(*610*) Lee, T.-S. et al. (2020). Alchemical Binding Free Energy Calculations in AMBER20: Advances and Best Practices for Drug Discovery. *Journal of Chemical Information and Modeling 60*, 5595–5623.

(*611*) FEP+ Performance / Benchmark Data for Supported Cards `https://support.schrodinger.com/s/article/185572` (accessed Nov. 20, 2024).

(*612*) Genheden, S., and Ryde, U. (2015). The MM/PBSA and MM/GBSA Methods to Estimate Ligand-Binding Affinities. *Expert Opinion on Drug Discovery 10*, 449–461.

(*613*) Srinivasan, J., Cheatham, T. E., Cieplak, P., Kollman, P. A., and Case, D. A. (1998). Continuum Solvent Studies of the Stability of DNA, RNA, and Phosphoramidate-DNA Helices. *Journal of the American Chemical Society 120*, 9401–9409.

(*614*) Mikulskis, P., Genheden, S., and Ryde, U. (2014). Effect of Explicit Water Molecules on Ligand-Binding Affinities Calculated with the MM/GBSA Approach. *Journal of Molecular Modeling 20*, 2273.

(*615*) Swanson, J. M. J., Henchman, R. H., and McCammon, J. A. (2004). Revisiting Free Energy Calculations: A Theoretical Connection to MM/PBSA and Direct Calculation of the Association Free Energy. *Biophysical Journal 86*, 67–74.

(*616*) Wang, C., Greene, D., Xiao, L., Qi, R., and Luo, R. (2018). Recent Developments and Applications of the MMPBSA Method. *Frontiers in Molecular Biosciences 4*, 87.

(*617*) Xie, D. (2014). New Solution Decomposition and Minimization Schemes for Poisson–Boltzmann Equation in Calculation of Biomolecular Electrostatics. *Journal of Computational Physics 275*, 294–309.

(*618*) Onufriev, A., Case, D. A., and Bashford, D. (2002). Effective Born Radii in the Generalized Born Approximation: The Importance of Being Perfect. *Journal of Computational Chemistry 23*, 1297–1304.

(*619*) Schaefer, M., and Froemmel, C. (1990). A Precise Analytical Method for Calculating the Electrostatic Energy of Macromolecules in Aqueous Solution. *Journal of Molecular Biology 216*, 1045–1066.

(*620*) Colmenares, J., Ortiz, J., and Rocchia, W. (2014). GPU Linear and Non-Linear Poisson–Boltzmann Solver Module for DelPhi. *Bioinformatics 30*, 569–570.

(*621*) Qi, R., Botello-Smith, W. M., and Luo, R. (2017). Acceleration of Linear Finite-Difference Poisson–Boltzmann Methods on Graphics Processing Units. *Journal of Chemical Theory and Computation 13*, 3378–3387.

(*622*) Kaukonen, M., Söderhjelm, P., Heimdal, J., and Ryde, U. (2008). QM/MM-PBSA Method To Estimate Free Energies for Reactions in Proteins. *The Journal of Physical Chemistry B 112*, 12537–12548.

(*623*) Su, P.-C., Tsai, C.-C., Mehboob, S., Hevener, K. E., and Johnson, M. E. (2015). Comparison of Radii Sets, Entropy, QM Methods, and Sampling on MM-PBSA, MM-GBSA, and QM/MM-GBSA Ligand Binding Energies of F. Tularensis Enoyl-ACP Reductase (FabI). *Journal of Computational Chemistry 36*, 1859–1873.

(*624*) Chen, J., Wang, J., Zhang, Q., Chen, K., and Zhu, W. (2015). A Comparative Study of Trypsin Specificity Based on QM/MM Molecular Dynamics Simulation and QM/MM GBSA Calculation. *Journal of Biomolecular Structure and Dynamics 33*, 2606–2618.

(*625*) Qian, G. et al. (2023). Outcomes with and without Outpatient SARS-CoV-2 Treatment for Patients with COVID-19 and Systemic Autoimmune Rheumatic Diseases: A Retrospective Cohort Study. *The Lancet Rheumatology 5*, e139–e150.

(*626*) Tahir, M. (2021). Coronavirus Genomic nsp14-ExoN, Structure, Role, Mechanism, and Potential Application as a Drug Target. *Journal of Medical Virology 93*, 4258–4264.

(*627*) Painter, W. P., Holman, W., Bush, J. A., Almazedi, F., Malik, H., Eraut, N. C. J. E., Morin, M. J., Szewczyk, L. J., and Painter, G. R. (2021). Human Safety, Tolerability, and Pharmacokinetics of Molnupiravir, a Novel Broad-Spectrum Oral Antiviral Agent with Activity against SARS-CoV-2. *Antimicrobial Agents and Chemotherapy 65*, e02428–20.

(*628*) Amara, A., Penchala, S. D., Else, L., Hale, C., FitzGerald, R., Walker, L., Lyons, R., Fletcher, T., and Khoo, S. (2021). The Development and Validation of a Novel LC-MS/MS Method for the Simultaneous Quantification of Molnupiravir and Its Metabolite ß-d-N4-hydroxycytidine in Human Plasma and Saliva. *Journal of Pharmaceutical and Biomedical Analysis 206*, 114356.

(*629*) Malone, B., and Campbell, E. A. (2021). Molnupiravir: Coding for Catastrophe. *Nature Structural & Molecular Biology 28*, 706–708.

(*630*) Gordon, C. J., Tchesnokov, E. P., Woolner, E., Perry, J. K., Feng, J. Y., Porter, D. P., and Götte, M. (2020). Remdesivir Is a Direct-Acting Antiviral That Inhibits RNA-dependent RNA Polymerase from Severe Acute Respiratory Syndrome Coronavirus 2 with High Potency. *The Journal of Biological Chemistry 295*, 6785–6797.

(*631*) Sanderson, T., Hisner, R., Donovan-Banfield, I., Hartman, H., Løchen, A., Peacock, T. P., and Ruis, C. (2023). A Molnupiravir-Associated Mutational Signature in Global SARS-CoV-2 Genomes. *Nature 623*, 594–600.

(*632*) Tchesnokov, E. P., Feng, J. Y., Porter, D. P., and Götte, M. (2019). Mechanism of Inhibition of Ebola Virus RNA-Dependent RNA Polymerase by Remdesivir. *Viruses 11*, 326.

(*633*) Akinosoglou, K., Schinas, G., and Gogos, C. (2022). Oral Antiviral Treatment for COVID-19: A Comprehensive Review on Nirmatrelvir/Ritonavir. *Viruses 14*, 2540.

(*634*) Cooper, C. L., van Heeswijk, R. P. G., Gallicano, K., and Cameron, D. W. (2003). A Review of Low-Dose Ritonavir in Protease Inhibitor Combination Therapy. *Clinical Infectious Diseases: An Official Publication of the Infectious Diseases Society of America 36*, 1585–1592.

(*635*) Joyce, R. P., Hu, V. W., and Wang, J. (2022). The History, Mechanism, and Perspectives of Nirmatrelvir (PF-07321332): An Orally Bioavailable Main Protease Inhibitor Used in Combination with Ritonavir to Reduce COVID-19-related Hospitalizations. *Medicinal Chemistry Research 31*, 1637–1646.

(*636*) Fountain-Jones, N. M., Vanhaeften, R., Williamson, J., Maskell, J., Chua, I.-L. J., Charleston, M., and Cooley, L. (2024). Effect of Molnupiravir on SARS-CoV-2 Evolution in Immunocompromised Patients: A Retrospective Observational Study. *The Lancet Microbe 5*, e452–e458.

(*637*) Yan, V. C., and Muller, F. L. Why Remdesivir Failed: Preclinical Assumptions Overestimate the Clinical Efficacy of Remdesivir for COVID-19 and Ebola. *Antimicrobial Agents and Chemotherapy 65*, e01117–21.

(*638*) Wang, Y. et al. (2020). Remdesivir in Adults with Severe COVID-19: A Randomised, Double-Blind, Placebo-Controlled, Multicentre Trial. *The Lancet 395*, 1569–1578.

(*639*) Grundeis, F. et al. (2023). Remdesivir for the Treatment of COVID-19. *The Cochrane Database of Systematic Reviews 1*, CD014962.

(*640*) Godwin, P. O., Polsonetti, B., Caron, M. F., and Oppelt, T. F. (2024). Remdesivir for the Treatment of COVID-19: A Narrative Review. *Infectious Diseases and Therapy 13*, 1–19.

(*641*) Wang, Y., Chen, X., Xiao, W., Zhao, D., and Feng, L. (2022). Rapid COVID-19 Rebound in a Severe COVID-19 Patient during 20-Day Course of Paxlovid. *Journal of Infection 85*, e134–e136.

(*642*) Anderson, A. S., Caubel, P., and Rusnak, J. M. (2022). Nirmatrelvir–Ritonavir and Viral Load Rebound in Covid-19. *The New England Journal of Medicine*, NEJMc2205944.

(*643*) Charness, M. E., Gupta, K., Stack, G., Strymish, J., Adams, E., Lindy, D. C., Mohri, H., and Ho, D. D. (2022). Rebound of SARS-CoV-2 Infection after Nirmatrelvir-Ritonavir Treatment. *The New England journal of medicine 387*, 1045–1047.

(*644*) Zheng, Q., Ma, P., Wang, M., Cheng, Y., Zhou, M., Ye, L., Feng, Z., and Zhang, C. (2023). Efficacy and Safety of Paxlovid for COVID-19:A Meta-Analysis. *Journal of Infection 86*, 66–117.

(*645*) Shadrick, W. R., Ndjomou, J., Kolli, R., Mukherjee, S., Hanson, A. M., and Frick, D. N. (2013). Discovering New Medicines Targeting Helicases: Challenges and Recent Progress. *Journal of Biomolecular Screening 18*, 761–781.

(*646*) Kleymann, G. (2004). Helicase Primase: Targeting the Achilles Heel of Herpes Simplex Viruses. *Antiviral Chemistry & Chemotherapy 15*, 135–140.

(*647*) Uhlig, N., Donner, A.-K., Gege, C., Lange, F., Kleymann, G., and Grunwald, T. (2021). Helicase Primase Inhibitors (HPIs) Are Efficacious for Therapy of Human Herpes Simplex Virus (HSV) Disease in an Infection Mouse Model. *Antiviral Research 195*, 105190.

(*648*) Field, H. J., and Biswas, S. (2011). Antiviral Drug Resistance and Helicase–Primase Inhibitors of Herpes Simplex Virus. *Drug Resistance Updates 14*, 45–51.

(*649*) Weller, S. K., and Kuchta, R. D. (2013). The DNA Helicase–Primase Complex as a Target for Herpes Viral Infection. *Expert opinion on therapeutic targets 17*, 1119–1132.

(*650*) Bermek, O., and Williams, R. S. The Three-Component Helicase/Primase Complex of Herpes Simplex Virus-1. *Open Biology 11*, 210011.

(*651*) Kawashima, M., Nemoto, O., Honda, M., Watanabe, D., Nakayama, J., Imafuku, S., Kato, T., and Katsuramaki, T. (2017). Amenamevir, a Novel Helicase–Primase Inhibitor, for Treatment of Herpes Zoster: A Randomized, Double-blind, Valaciclovir-controlled Phase 3 Study. *The Journal of Dermatology 44*, 1219–1227.

(*652*) Belon, C. A., and Frick, D. N. (2009). Helicase Inhibitors as Specifically Targeted Antiviral Therapy for Hepatitis C. *Future virology 4*, 277–293.

(*653*) Frick, D. N. (2007). The Hepatitis C Virus NS3 Protein: A Model RNA Helicase and Potential Drug Target. *Current Issues in Molecular Biology 9*, 1–20.

(*654*) Salam, K. A., and Akimitsu, N. (2013). Hepatitis C Virus NS3 Inhibitors: Current and Future Perspectives. *BioMed Research International 2013*, 467869.

(*655*) Mukherjee, S. et al. (2012). Identification and Analysis of Hepatitis C Virus NS3 Helicase Inhibitors Using Nucleic Acid Binding Assays. *Nucleic Acids Research 40*, 8607–8621.

(*656*) Jang, H., Ryoo, S.-R., Kim, Y.-K., Yoon, S., Kim, H., Han, S. W., Choi, B.-S., Kim, D.-E., and Min, D.-H. (2013). Discovery of Hepatitis C Virus NS3 Helicase Inhibitors by a Multiplexed, High-Throughput Helicase Activity Assay Based on Graphene Oxide. *Angewandte Chemie International Edition 52*, 2340–2344.

(*657*) Yazdani, S., De Maio, N., Ding, Y., Shahani, V., Goldman, N., and Schapira, M. (2021). Genetic Variability of the SARS-CoV-2 Pocketome. *Journal of Proteome Research 20*, 4212–4215.

(*658*) Wu, P. (2020). Inhibition of RNA-binding Proteins with Small Molecules. *Nature Reviews Chemistry 4*, 441–458.

(*659*) Neumann, T., Junker, H.-D., Schmidt, K., and Sekul, R. (2007). SPR-based Fragment Screening: Advantages and Applications. *Current Topics in Medicinal Chemistry 7*, 1630–1642.

(*660*) Lameirinhas, R. A. M., Torres, J. P. N., Baptista, A., and Martins, M. J. M. (2022). A New Method to Analyse the Role of Surface Plasmon Polaritons on Dielectric-Metal Interfaces. *IEEE Photonics Journal 14*, 1–9.

(*661*) Sparks, R. P., Jenkins, J. L., and Fratti, R. In *SNAREs: Methods and Protocols*, Fratti, R., Ed.; Springer: New York, NY, 2019, pp 199–210.

(*662*) Rule, C. S., Patrick, M., and Sandkvist, M. (2016). Measuring In Vitro ATPase Activity for Enzymatic Characterization. *Journal of Visualized Experiments : JoVE*, 54305.

(*663*) Newman, J. A. et al. (2021). Structure, Mechanism and Crystallographic Fragment Screening of the SARS-CoV-2 NSP13 Helicase. *Nature Communications 12*, 4848.

(*664*) REAL Database - Enamine `https://enamine.net/compound-collections/real-compounds/real-database` (accessed Nov. 21, 2024).

(*665*) Capecchi, A., Probst, D., and Reymond, J.-L. (2020). One Molecular Fingerprint to Rule Them All: Drugs, Biomolecules, and the Metabolome. *Journal of Cheminformatics 12*, 43.

(*666*) Riniker, S., and Landrum, G. A. (2013). Open-Source Platform to Benchmark Fingerprints for Ligand-Based Virtual Screening. *Journal of Cheminformatics 5*, 26.

(*667*) Landrum, G. et al. rdkit/rdkit: 2025_03_3 (Q1 2025) Release, version Release_2025_03_3, 2025.

(*668*) GitHub/Schrodinger/Gpusimilarity `https://github.com/schrodinger/gpusimilarity` (accessed Nov. 21, 2024).

(*669*) Schubert, E., and Rousseeuw, P. J. (2021). Fast and eager k-medoids clustering: $O(k)$ runtime improvement of the PAM, CLARA, and CLARANS algorithms. *Information Systems 101*, 101804.

(*670*) Schubert, E., and Lenssen, L. (2022). Fast K-Medoids Clustering in Rust and Python. *Journal of Open Source Software 7*, 4183.

(*671*) Valdés-Tresanco, M. S., Valdés-Tresanco, M. E., Valiente, P. A., and Moreno, E. (2021). gmx_MMPBSA: A New Tool to Perform End-State Free Energy Calculations with GROMACS. *Journal of Chemical Theory and Computation 17*, 6281–6291.

(*672*) Wang, J., Wang, W., Kollman, P. A., and Case, D. A. (2006). Automatic Atom Type and Bond Type Perception in Molecular Mechanical Calculations. *Journal of Molecular Graphics and Modelling 25*, 247–260.

(*673*) Friesner, R. A. et al. (2004). Glide: A New Approach for Rapid, Accurate Docking and Scoring. 1. Method and Assessment of Docking Accuracy. *Journal of Medicinal Chemistry 47*, 1739–1749.

(*674*) Ekberg, V., and Ryde, U. (2021). On the Use of Interaction Entropy and Related Methods to Estimate Binding Entropies. *Journal of Chemical Theory and Computation 17*, 5379–5391.

(*675*) Wever, M. J. A. et al. (2024). Structure-Based Discovery of First Inhibitors Targeting the Helicase Activity of Human PIF1. *Nucleic Acids Research 52*, 12616–12632.

(*676*) Paul, S. M., Mytelka, D. S., Dunwiddie, C. T., Persinger, C. C., Munos, B. H., Lindborg, S. R., and Schacht, A. L. (2010). How to improve R&D productivity: the pharmaceutical industry's grand challenge. *Nature reviews Drug discovery 9*, 203–214.

(*677*) Gimeno, A., Ojeda-Montes, M. J., Tomás-Hernández, S., Cereto-Massagué, A., Beltrán-Debón, R., Mulero, M., Pujadas, G., and Garcia-Vallvé, S. (2019). The Light and Dark Sides of Virtual Screening: What Is There to Know? *International Journal of Molecular Sciences 20*, 1375.

(*678*) T, S., A, B., G, T., Jl, M.-F., K, M.-M., T, L., K, C.-C., and Dk, A. (2012). Recognizing Pitfalls in Virtual Screening: A Critical Review. *Journal of chemical information and modeling 52*, DOI: 10.1021/ci200528d.

(*679*) Kolb, P., and Irwin, J. J. (2009). Docking Screens: Right for the Right Reasons? *Current Topics in Medicinal Chemistry 9*, 755.

(*680*) Groenhof, G. (2013). Introduction to QM/MM Simulations. *Methods in Molecular Biology (Clifton, N.J.) 924*, 43–66.

(*681*) Mehmood, R., and Kulik, H. J. (2022). Quantum-Mechanical/Molecular-Mechanical (QM/MM) Simulations for Understanding Enzyme Dynamics. *Methods in Molecular Biology (Clifton, N.J.) 2397*, 227–248.

(*682*) Ainsley, J., Lodola, A., Mulholland, A. J., Christov, C. Z., and Karabencheva-Christova, T. G. In *Advances in Protein Chemistry and Structural Biology*, Karabencheva-Christova, T. G., and Christov, C. Z., Eds.; Computational Molecular Modelling in Structural Biology, Vol. 113; Academic Press: 2018, pp 1–32.

(*683*) Walters, W. P., and Barzilay, R. (2021). Applications of Deep Learning in Molecule Generation and Molecular Property Prediction. *Accounts of Chemical Research 54*, 263–270.

(*684*) Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Gupta, B. B., Chen, X., and Wang, X. (2021). A Survey of Deep Active Learning. *ACM Comput. Surv. 54*, 180:1–180:40.

(*685*) Tharwat, A., and Schenck, W. (2023). A Survey on Active Learning: State-of-the-Art, Practical Challenges and Research Directions. *Mathematics 11*, 820.

(*686*) Cox, A. D., Fesik, S. W., Kimmelman, A. C., Luo, J., and Der, C. J. (2014). Drugging the Undruggable RAS: Mission Possible? *Nature Reviews. Drug Discovery 13*, 828–851.

(*687*) P, G., J, J., M, F., A, L., R, C., Je, S., and A, W. (1992). Mutational and Kinetic Analyses of the GTPase-activating Protein (GAP)-P21 Interaction: The C-terminal Domain of GAP Is Not Sufficient for Full Activity. *Molecular and cellular biology 12*, DOI: 10.1128/mcb.12.5.2050-2056.1992.

(*688*) Ak, M., and Dg, L. (2016). Invited Review: Small GTPases and Their GAPs. *Biopolymers 105*, DOI: 10.1002/bip.22833.

(*689*) Parker, M. I., Meyer, J. E., Golemis, E. A., and Dunbrack, J., Roland L. (2022). Delineating the RAS Conformational Landscape. *Cancer Research 82*, 2485–2498.

(*690*) Hobbs, G. A., Der, C. J., and Rossman, K. L. (2016). RAS Isoforms and Mutations in Cancer at a Glance. *Journal of Cell Science 129*, 1287–1292.

(*691*) Moore, A. R., Rosenberg, S. C., McCormick, F., and Malek, S. (2020). RAS-targeted Therapies: Is the Undruggable Drugged? *Nature Reviews Drug Discovery 19*, 533–552.

(*692*) Papke, B. et al. (2021). Silencing of Oncogenic KRAS by Mutant-Selective Small Interfering RNA. *ACS Pharmacology & Translational Science 4*, 703–712.

(*693*) Canon, J. et al. (2019). The clinical KRAS(G12C) inhibitor AMG 510 drives anti-tumour immunity. *Nature 575*, 217–223.

(*694*) Canon, J. et al. (2019). The Clinical KRAS(G12C) Inhibitor AMG 510 Drives Anti-Tumour Immunity. *Nature 575*, 217–223.

(*695*) Patricelli, M. P. et al. (2016). Selective Inhibition of Oncogenic KRAS Output with Small Molecules Targeting the Inactive State. *Cancer Discovery 6*, 316–329.

(*696*) Fell, J. B. et al. (2020). Identification of the Clinical Development Candidate MRTX849, a Covalent KRASG12C Inhibitor for the Treatment of Cancer. *Journal of Medicinal Chemistry 63*, 6679–6693.

(*697*) Wei, D., Wang, L., Zuo, X., Maitra, A., and Bresalier, R. S. (2024). A Small Molecule with Big Impact: MRTX1133 Targets the KRASG12D Mutation in Pancreatic Cancer. *Clinical Cancer Research: An Official Journal of the American Association for Cancer Research 30*, 655–662.

(*698*) Vatansever, S., Erman, B., and Gümüş, Z. H. (2019). Oncogenic G12D Mutation Alters Local Conformations and Dynamics of K-Ras. *Scientific Reports 9*, 11730.

(*699*) Kessler, D. et al. (2019). Drugging an Undruggable Pocket on KRAS. *Proceedings of the National Academy of Sciences 116*, 15823–15829.

(*700*) Li, C., Vides, A., Kim, D., Xue, J. Y., Zhao, Y., and Lito, P. (2021). The G protein signaling regulator RGS3 enhances the GTPase activity of KRAS. *Science 374*, 197–201.

(*701*) Cacciarelli, D., and Kulahci, M. (2024). Active Learning for Data Streams: A Survey. *Machine Learning 113*, 185–239.

(*702*) Cohn, D. A., Ghahramani, Z., and Jordan, M. I. (1996). Active Learning with Statistical Models. *Journal of Artificial Intelligence Research 4*, 129–145.

(*703*) Fontenla-Romero, Ó., Guijarro-Berdiñas, B., Martinez-Rego, D., Pérez-Sánchez, B., and Peteiro-Barral, D. In *Efficiency and Scalability Methods for Computational Intellect*; IGI Global Scientific Publishing: 2013, pp 27–54.

(*704*) Portugal, I., Alencar, P., and Cowan, D. (2018). The Use of Machine Learning Algorithms in Recommender Systems: A Systematic Review. *Expert Systems with Applications 97*, 205–227.

(*705*) Mohammed, A., and Kora, R. (2023). A Comprehensive Review on Ensemble Deep Learning: Opportunities and Challenges. *Journal of King Saud University - Computer and Information Sciences 35*, 757–774.

(*706*) Gastegger, M., Behler, J., and Marquetand, P. (2017). Machine Learning Molecular Dynamics for the Simulation of Infrared Spectra. *Chemical Science 8*, 6924–6935.

(*707*) Lubbers, N., Smith, J. S., and Barros, K. (2018). Hierarchical Modeling of Molecular Energies Using a Deep Neural Network. *The Journal of Chemical Physics 148*, 241715.

(*708*) Seung, H. S., Opper, M., and Sompolinsky, H. In *Proceedings of the fifth annual workshop on Computational learning theory*, 1992, pp 287–294.

(*709*) Freund, Y., Seung, H. S., Shamir, E., and Tishby, N. (1997). Selective Sampling Using the Query by Committee Algorithm. *Machine Learning 28*, 133–168.

(*710*) Smith, J. S., Nebgen, B., Lubbers, N., Isayev, O., and Roitberg, A. E. (2018). Less Is More: Sampling Chemical Space with Active Learning. *The Journal of Chemical Physics 148*, 241733.

(*711*) Khalak, Y., Tresadern, G., Hahn, D. F., de Groot, B. L., and Gapsys, V. (2022). Chemical Space Exploration with Active Learning and Alchemical Free Energies. *Journal of Chemical Theory and Computation 18*, 6259–6270.

(*712*) Yang, Y., Yao, K., Repasky, M. P., Leswing, K., Abel, R., Shoichet, B. K., and Jerome, S. V. (2021). Efficient Exploration of Chemical Space with Docking and Deep Learning. *Journal of Chemical Theory and Computation 17*, 7106–7119.

(*713*) Gentile, F., Agrawal, V., Hsing, M., Ton, A.-T., Ban, F., Norinder, U., Gleave, M. E., and Cherkasov, A. (2020). Deep Docking: A Deep Learning Platform for Augmentation of Structure Based Drug Discovery. *ACS Central Science 6*, 939–949.

(*714*) Thompson, J., Walters, W. P., Feng, J. A., Pabon, N. A., Xu, H., Maser, M., Goldman, B. B., Moustakas, D., Schmidt, M., and York, F. (2022). Optimizing Active Learning for Free Energy Calculations. *Artificial Intelligence in the Life Sciences 2*, 100050.

(*715*) Gasteiger, J., Groß, J., and Günnemann, S. (2022). Directional Message Passing for Molecular Graphs. DOI: 10.48550/arXiv.2003.03123.

(*716*) Muegge, I., and Mukherjee, P. (2016). An overview of molecular fingerprint similarity search in virtual screening. *Expert opinion on drug discovery 11*, 137–148.

(*717*) Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems 28*.

(*718*) Rogers, D., and Hahn, M. (2010). Extended-connectivity fingerprints. *Journal of chemical information and modeling 50*, 742–754.

(*719*) Cereto-Massagué, A., Ojeda, M. J., Valls, C., Mulero, M., Garcia-Vallvé, S., and Pujadas, G. (2015). Molecular fingerprint similarity search in virtual screening. *Methods 71*, 58–63.

(*720*) Bemis, G. W., and Murcko, M. A. (1996). The Properties of Known Drugs. 1. Molecular Frameworks. *Journal of Medicinal Chemistry 39*, 2887–2893.

(*721*) Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. (2018). MoleculeNet: A Benchmark for Molecular Machine Learning. *Chemical Science 9*, 513–530.

(*722*) Guo, Q., Hernandez-Hernandez, S., and Ballester, P. J. (2024). Scaffold Splits Overestimate Virtual Screening Performance. DOI: 10.48550/arXiv.2406.00873.

(*723*) Berta, D., Gehrke, S., Nyíri, K., Vértessy, B. G., and Rosta, E. (2023). Mechanism-Based Redesign of GAP to Activate Oncogenic Ras. *J. Am. Chem. Soc. 145*, 20302–20310.

(*724*) Heid, E., Greenman, K. P., Chung, Y., Li, S.-C., Graff, D. E., Vermeire, F. H., Wu, H., Green, W. H., and McGill, C. J. (2024). Chemprop: A Machine Learning Package for Chemical Property Prediction. *Journal of Chemical Information and Modeling 64*, 9–17.

(*725*) Ioffe, S., and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. DOI: 10.48550/arXiv.1502.03167.

(*726*) Loshchilov, I., and Hutter, F. (2017). SGDR: Stochastic Gradient Descent with Warm Restarts. DOI: 10.48550/arXiv.1608.03983.