LEVERAGING RELATIONAL STRUCTURE THROUGH MESSAGE PASSING FOR MODELLING NON-EUCLIDEAN DATA

PETER MELTZER

THESIS

Submitted for degree of PhD

First Supervisor Peter J. Bentley
Second Supervisor Mark Herbster

Computer Science Department
Engineering Faculty
University College London (UCL)

I, Peter Meltzer, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this had been indicated in the thesis.

[redacted]

ABSTRACT

Modelling non-Euclidean data is difficult since objects for comparison can be formed of different numbers of constituent parts with different numbers of relations between them, and traditional (Euclidean) methods are non-trivial to apply. Message passing enables such modelling by leveraging the structure of the relations within a (or between) given object(s) in order to represent and compare structure in a vectorized form of fixed dimensions.

In this work, we contribute novel message passing techniques that improve state of the art for non-Euclidean modelling in a set of specifically chosen domains. In particular, (1) we introduce an attention-based structure-aware global pooling operator for graph classification and demonstrate its effectiveness on a range of chemical property prediction benchmarks, we also show that our method outperforms state of the art graph classifiers in a graph isomorphism test, and demonstrate the interpretability of our method with respect to the learned attention coefficients. (2) We propose a style similarity measure for Boundary Representations (B-Reps) that leverages the style signals in the second order statistics of the activations in a pre-trained (unsupervised) 3D encoder, and learns their relative importance to an end-user through few-shot learning. Our approach differs from existing data-driven 3D style methods since it may be used in completely unsupervised settings. We show quantitatively that our proposed method with B-Reps is able to capture stronger style signals than alternative methods on meshes and point clouds despite its significantly greater computational efficiency. We also show it is able to generate meaningful style gradients with respect to the input shape. (3) We introduce a novel message passing-based model of computation and demonstrate its effectiveness in expressing the complex dependencies of biological systems necessary to model life-like systems and tracing cell lineage during cancerous tumour growth, and demonstrate the improvement over existing methods in terms of post-analysis.

IMPACT STATEMENT

This thesis has practical applications in three diverse yet important areas.

First, we address graph level representation learning in the context of chemical property prediction. The ability to reliably predict molecular properties such toxicity and mutagenicity is of vital significance in drug discovery and other pharmaceutical applications. Conventional high throughput screening techniques are expensive and time consuming, thus any computational methods able to reduce the number of drug candidates by ruling out those that are harmful will help to reduce costs, save time and ultimately deliver new drugs more efficiently.

Second, we propose an unsupervised method for 3D style metric learning for Boundary Representations (B-Reps). B-Reps are the industry standard representation for Computer Aided Design (CAD), and are heavily relied on in product design industries where style is of great importance. However, there is very little existing work applying style learning (or even any machine learning) methods to B-Reps. Our style metric may be used to provide feedback directly to designers by indicating parts of a design that do no match the target style, as well as by finding existing components that match intended style.

Finally, we propose a novel modelling technique for simulating complex dynamic biological systems, which we investigate in the context of cancerous tumour growth. Understanding how cells evolve during division, and thus how tumours arise and develop is paramount in determining which treatments may be effective, as well as aiding in the design and understanding of new treatments.

ACKNOWLEDGMENTS

First and foremost I would like to thank my supervisor, Peter J. Bentley, for always demanding the best from me, and supporting and challenging me (and sometimes my patience!) until I get there. It's been a long tough ride, but I have grown so much as a result. I'd like to thank my family, and in particular my brother Tom, for always believing in me and helping me to believe in myself, and all my UCL friends, whether from CS or Physics, for keeping me sane, letting me know I'm not alone, and just being there for me. I could not have done this without you.

Thank you to my ex-Braintree colleagues (Marcelo, Kasia, Kyohei, Laura, ...) for making my day-to-day in the office so much fun! In particular thanks to Arturo Araujo (my unofficial adopted second supervisor) for being such a great support and role model.

Special thanks to Amir Khasahmadi for sanity checking my Introduction and Literature Review chapters, to Victoria Clark for her outstanding knowledge of chemistry that helped me to understand and plot the molecules in Chapter 3, to Hooman Shayani and my other Autodesk collaborators for teaching me so much and being such a pleasure to work with, and to Braintree and UCL CS department for providing the funding for me to complete the work presented in this thesis.

Last, but by no means least, I would like to thank my loving partner, Nick Pelly-Fry, for putting up with my stress, my ranting, my anxieties, and all the rest, and supporting me to succeed. You are awesome, thank you!

PUBLICATIONS AND OPEN SOURCE CONTRIBUTIONS

PUBLICATIONS

Peer Reviewed:

- Peter Meltzer, Hooman Shayani, Amir Khasahmadi, Pradeep Kumar Jayaraman, Aditya Sanghi, and Joseph Lambourne. "UVStyle-Net: Unsupervised Few-Shot Learning of 3D Style Similarity Measure for B-Reps." In: *International Conference on Computer Vision* (*ICCV*) (2021)
- Joseph Lambourne, Karl D. D. Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. "BRepNet: A Topological Message Passing System for Solid Models." In: *Computer Vision and Pattern Recognition* (*CVPR*) (2021)
- Peter Meltzer, Marcelo Daniel Gutierrez Mallea, and Peter J. Bentley. "PiNet: Attention Pooling for Graph Classification." In: *Neural Information Processing Systems* (*NeurIPS*)*: Graph Representation Learning Workshop*. 2019
- Peter Meltzer and Peter J. Bentley. "Interacting Hierarchical Dynamic Networks." In: *The 2018 Conference on Artificial Life: A Hybrid of the European Conference on Artificial Life* (*ECAL*) *and the International Conference on the Synthesis and Simulation of Living Systems* (*ALIFE*) (2018), pp. 582–589. DOI: 10.1162/isal_a_00108

ArXiv:

- Peter Meltzer, M.D.G. Marcelo Daniel Gutierrez Mallea, and Peter J. Bentley. "PiNet: A Permutation Invariant Graph Neural Network for Graph Classification." In: *arXiv* (2019)
- Marcelo Daniel Gutierrez Mallea, Peter Meltzer, and Peter J. Bentley. "Capsule Neural Networks for Graph Classification Using Explicit Tensorial Graph Representations." In: *arXiv* (2019)

Principle Supervisor for MSc Computer Science Student:

- Federated Learning with Graph ML Algorithms — Akash Bhattacharya

OPEN SOURCE

- DeepGL: https://github.com/neo4j-graph-analytics/ml-models

- IHDNs: `https://github.com/meltzerpete/IHDN`
- PiNet: `https://github.com/meltzerpete/PiNet`
- UVStyle-Net: `https://github.com/AutodeskAILab/UVStyle-Net`

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

ABM	Agent-Based Model
API	Application Programming Interface
CAD	Computer Aided Design
B-Rep	Boundary Representation
CNN	Convolutional Neural Network
DGCNN	Dynamic Graph CNN
DGI	Deep Graph InfoMax
FLAG	Free Large-scale Adversarial Augmentation on Graphs
GAN	Generative Adversarial Network
GAT	Graph Attention Networks
GCN	Graph Convolutional Network
GG-NN	Gated Graph Neural Network
GIN	Graph Isomorphism Network
GNN	Graph Neural Network
GRU	Gated Recurrent Unit
IHDN	Interacting Hierarchical Dynamic Network
OGB	Open Graph Benchmark
MLP	Multi-Layer Perceptron
MPNN	Message Passing Neural Network
NLP	Natural Language Processing
PCA	Principal Component Analysis
PIC	Power Iteration Clustering
PNA	Principle Neighbourhood Aggregation
RNN	Recurrent Neural Network
ReLU	Rectified Linear Unit
WL	Weisfeiler-Lehman

1 INTRODUCTION

Message passing is a fundamental abstraction in Computer Science that has become ubiquitous in modern computing. While message passing has a long history in concurrency and distributed systems, even forming the basis of the Turing Complete π-calculus [7], its use in data-driven settings is just as significant.

We provide a formal definition of message passing in Section 1.3, however for the purposes of this introduction it is sufficient to consider message passing as the exchange and/or update of state between separate units connected according to some graph. Messages facilitate the exchange of state between units and may only be sent and received between connected units; and based on messages received, units may update their own state or otherwise. The vast uptake of message passing in concurrent and distributed systems is likely due to the expressive yet simple abstraction it affords, making it easy to reason about and formalise the behaviour of systems of separate computational units connected according to fixed, or even dynamic, physical or virtual networks.

The effectiveness and expressivity of message passing for Artificial Intelligence applications can be seen in Cellular Automata as far back as the 1950's [8], in which a dense grid of units perform simple update rules based on their local neighbourhoods, giving rise to emergent properties and effects. However, it is not until much later that this concept is applied in data-driven settings.

The efficiency advantages of message passing in leveraging sparsity are of particular significance in Belief Propagation methods, which have been a popular technique since their introduction in the 1980's [9]. In this context, sum-product message passing (messages are summed or multiplied) enables efficient inference on graphical models such as Bayesian networks and Markov Random Fields [10], where the graphs express the conditional dependence structure between random variables.

Further advantages to message passing can be seen in algorithms where the message passing graph is instead prescribed by the non-Euclidean (not defined in n-dimensional linear space) relational structure of the given input data. The Page-rank algorithm [11] in which vertices represent web pages with edges indicating hyperlinks between them is an early example of this, and showcases the trivial scalability and efficiency typically afforded by message passing algorithms.

Since the fusion of message passing with deep neural networks, *i.e.* Message Passing Neural Networks (MPNNs), there has been an explosion in techniques and applications, especially in the areas of computer vision and chemo/bio-informatics, with MPNNs obtaining state of the art performance on a wide range of tasks [12]. It is precisely these areas that this work contributes to.

The power in MPNNs is largely due to their relational inductive biases, which enable the networks to learn not only about isolated entities, but also their relations and composition [13]. For example, through local aggregations and pooling in pixel coordinate space, conventional Convolutional Neural Networks (CNNs) provide a strong inductive bias for the hierarchical composition of edges, textures and objects [14]. Similarly, for MPNNs, relations between entities provide strong inductive biases for the hierarchical composition of these related entities, for example atoms to molecular substructures/motifs, words to concepts, or people to social groups and subcultures.

Note, the terms MPNN, Graph Neural Network (GNN), and Graph Convolutional Network (GCN) are used inconsistently in recent works. For example, Zou *et al.* [15] draw a distinction between GCNs and self-attention methods, whereas others such as Zhang *et al.* [16] include attention methods under the GCN category. The term GCN is also frequently used to indicate the specific instance of Kipf *et al.* [17], *i.e.* [18], and other times as a general collection of convolution methods [16].

For clarity, in this work GNN is used to describe any neural network architecture that operates on graphs, GCN for any convolutional architecture for graphs, and MPNN to describe any neural network that passes messages according to the local relational structure of its input(s). Thus, all MPNNs and GCNs are GNNs; however, the converse is not the case (see Figure 1.1). For referring to the specific instance of the GCN from [17] we provide the reference each time, *i.e.* "*The GCN* [17]...".

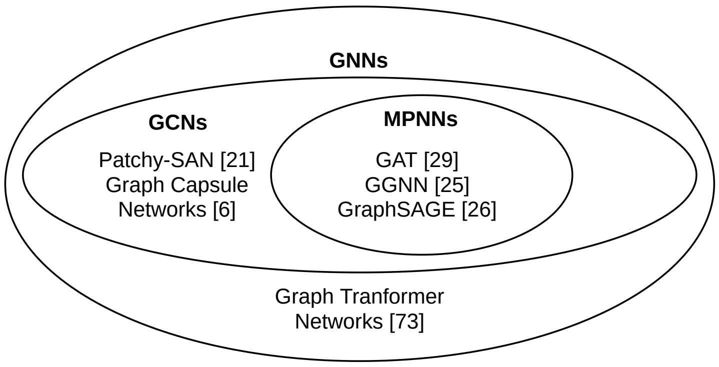


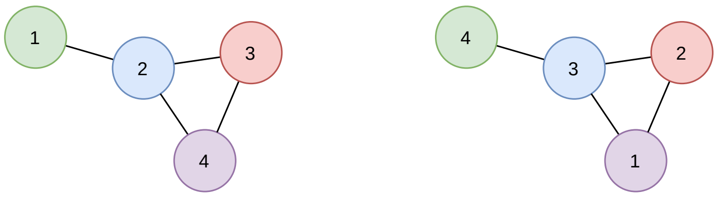
Figure 1.1: Differentiating GNNs, GCNs, and MPNNs, with examples of each.

1.1 WHY MESSAGE PASSING?

While the message passing abstraction has obvious advantages for reasoning about and implementing concurrent and distributed systems, message passing algorithms offer further advantages for handling non-Euclidean structured/relational data. In addition to their relational inductive biases discussed above, message passing algorithms are *permutation equivariant*.

In this work, a function f is *permutation equivariant* iff $f(P_\pi \cdot X) = P_\pi \cdot f(X)$, and *permutation invariant* iff $f(P_\pi \cdot X) = f(X)$, where $P_\pi \in \mathbb{R}^{N \times N}$ is any permutation matrix and $X \in \mathbb{R}^{N \times d}$.

By definition, the vertices of a graph form a set meaning that there is no explicit ordering to them, yet to represent them in computer memory and thus to process them requires applying some order. For example, the most common formats for storing and processing graphs include adjacency matrices, edge lists and linked lists, all of which assume an ordering on the vertices.



	1	2	3	4
1	0	1	0	0
2	1	0	1	1
3	0	1	0	1
4	0	1	1	0

	1	2	3	4
1	0	1	1	0
2	1	0	1	0
3	1	1	0	1
4	0	0	1	0

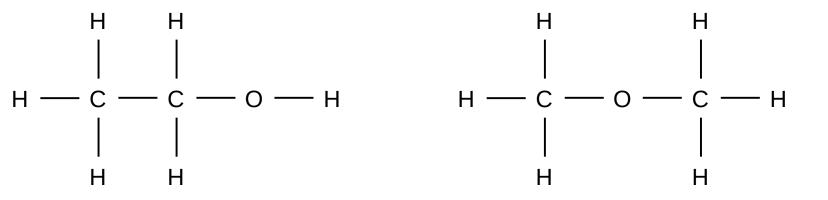
Figure 1.2: The two graphs shown are obviously isomorphic (colours indicate vertex correspondence), yet their adjacency matrices are not so clearly equivalent. While it is possible to permute the rows and columns of one matrix to exactly match the other, identifying exactly how is non-trivial and for arbitrary input graphs there exist no known polynomial time solutions [19].

As shown in Figure 1.2, isomorphic graphs can have completely different permutations of vertices. Thus, in order to recognise equivalent graphs and process them equivalently, there are two possibilities: apply a canonical labelling procedure as a pre-processing step to sort the vertices in a consistent way, or use permutation invariant or equivariant functions.

Due to the complexity of the graph isomorphism and sub-graph isomorphism problems (the sub-graph isomorphism problem is known to be NP-Complete [20]) and problems encountered with symmetries, canonical vertex labelling algorithms are either approximate or expensive to compute. Thus using message passing, which does not require such labellings, is clearly advantageous.

1.2 ILLUSTRATED EXAMPLE

Consider the molecules in Figure 1.3. Treating each molecule as a set of atoms is not enough to distinguish the two as the atoms in each are the same, thus the relational structure (*i.e.* which atoms are connected to which) is clearly of significance.



(a) Ethanol — boils at 78.4°*C* (b) Dimethyl Ether — boils at −24°*C*

Figure 1.3: Both molecules are formed of exactly the same number of carbon, hydrogen and oxygen atoms; yet the molecules have different properties due to the difference in their arrangement, *i.e.* their *relational structure*.

To illustrate the utility of message passing algorithms in capturing this relational information, we will consider the task of chemical property prediction. Suppose we have four graphs *A*, *B*, *C*, and *D* (see Figure 1.4) with arbitrarily assigned vertex ids.

In order to process the graphs with a conventional neural network we must first find a way to order the vertices canonically, not only to ensure that isomorphic graphs appear in exactly the same order, but also so that similar graphs appear in similar/sensible orders. Failure

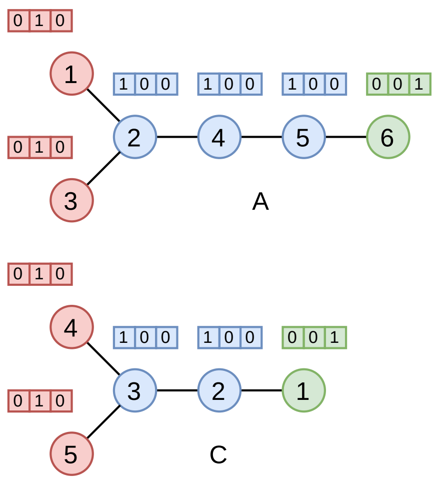
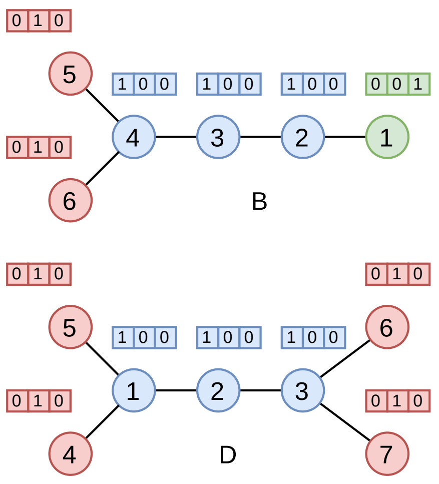



Figure 1.4: Four graphs representing chemical compounds. One-hot vertex vectors indicate type of atom.

to do so would be equivalent to feeding images to a 2D CNN with all of the pixels randomly shuffled.

We must then find a way to express each graph as a vector in a single vector space, *i.e.* a fixed number of dimensions. Common non-message passing techniques for this would be to either sample parts of larger graphs, zero pad smaller graphs, or to use some combination of both, *i.e.* [21]. However, for message passing we require neither sampling or padding, nor canonical ordering.

As a very trivial demonstration, suppose each vertex sends a message to its neighbours containing its associated state (the one-hot vector). If each vertex updates its new state to be the sum of its previous state (Figure 1.4) and the mean of all incoming messages, it would result in the vertex states shown in Figure 1.5.

To aid this discussion, let $G_v^{(t)}$ represent vertex v of graph G at time step t, *i.e.* $A_6^{(0)}$ represents the green vertex of graph A at $t = 0$ (before message passing), and $A_6^{(1)}$ the same vertex at $t = 1$ (after one round of message passing).

Already, even with such a trivial algorithm, a powerful property emerges. Namely, we observe that vertex roles with respect to local structure become apparent. For example, while all blue vertices at

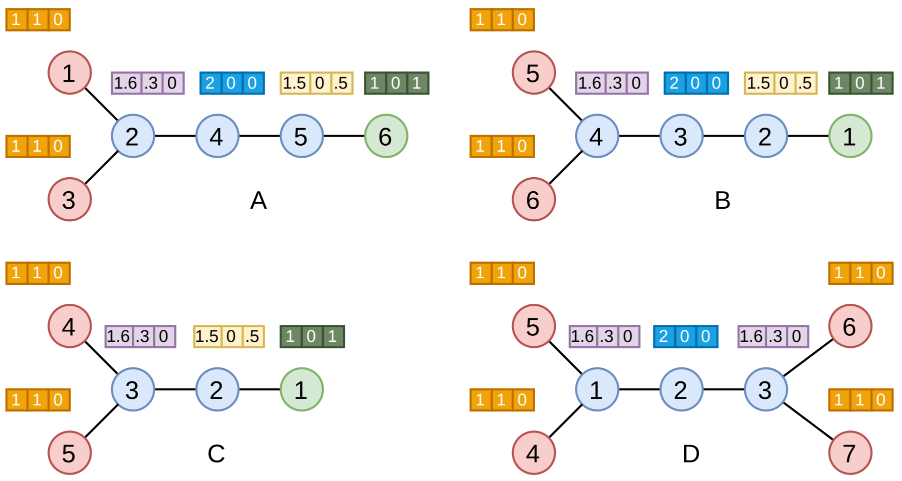


Figure 1.5: Each vertex updates its state to be the sum of its previous state with the mean of the incoming messages.

$t = 0$ (Figure 1.4) share the state $(1, 0, 0)$, at $t = 1$ (Figure 1.5) we observe that vertices with shared local structure now share their representations, *i.e.* $A_2^{(1)} = B_4^{(1)} = C_3^{(1)} = D_1^{(1)} = D_3^{(1)}$, but differ from vertices with different local structure, *i.e.* $A_2^{(1)} \neq A_4^{(1)}$ etc. This is significant, since it demonstrates that through passing messages according to the relations between entities (in this case atoms), we have leveraged the relational structure to implicitly capture structural information.

To consider each graph as a single vector, we now only require an aggregation function that is invariant to the ordering of the vertices, for example summation. Summing the resulting vertex representations in each graph, *i.e.* $\sum_v G_v^{(1)}$, produces the vectors shown in Figure 1.6.

First, we observe that the isomorphic graphs A and B receive exactly the same vectors. This property is in fact guaranteed by this algorithm due to the permutation equivariance of the message passing combined with the permutation invariance of the sum aggregation. Second, due to the fact that they differ by only one vertex, we observe that A and B are closer in Euclidean distance to C than D — a result that would also be obtained if we were to consider the graph edit distance [22]. Third, we see that C differs from A and B in only the x dimension; however,

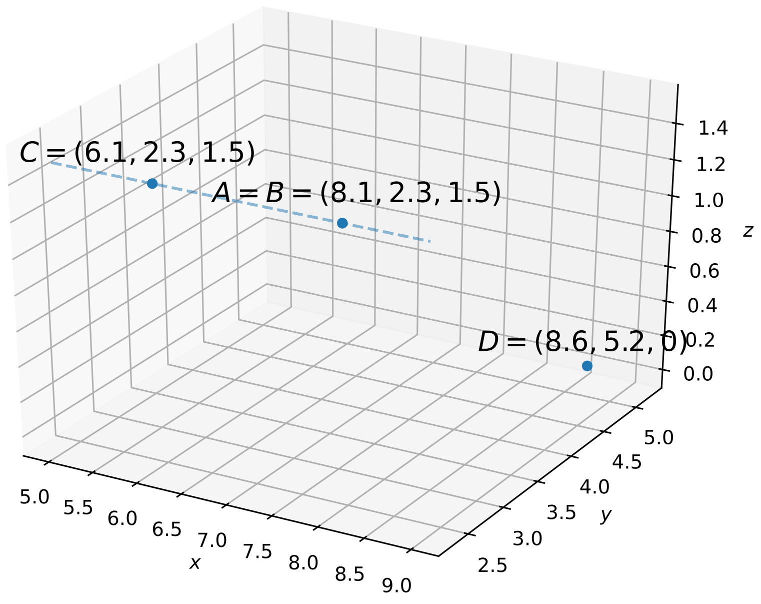


Figure 1.6: Resulting graph representation vectors — summation of all vertex states at $t = 1$: We see that $A = B$, $\|A - C\|_2 < \|A - D\|_2$ and that C differs from A and B only in the x dimension.

in this dimension, due to the shared chain of three blue vertices, A and B are in fact closer to D than C.

It is clear that despite the simplicity of this algorithm, the resulting vector representations capture the similarities and differences between these examples, *i.e.* they would be useful representations upon which a neural network could learn the relative importance of each dimension in discriminating the target property. Of course, the use of more complex, learnable message and update functions, along with more rounds of message passing and a greater number of dimensions would increase the expressivity and discriminative power significantly. For an example of this, Kipf *et al.* [17] demonstrate that a GCN with random weights (without training) is able to produce meaningful embeddings, which are then improved significantly with training.

1.3 AIM

Following the discussion above, the aim of this work is

> to develop novel message passing techniques that leverage relational structure in order to improve on existing state of the art methods for modelling non-Euclidean data,

according to the definitions below.

Given a graph $G = (V, E)$, with vertices $V = \{1, \dots, |V|\}$ and edges $E \subseteq V^2$, we adapt the MPNN framework [23] to define *message passing* to be the update of vertex states h_v^t, $\forall v \in V$ at time step t, according to messages m_v^{t+1}, such that

$$m_v^{t+1} = \bigotimes_t(\{M_t(h_v^t, h_u^t, e_{uv}),\ u \in \mathcal{N}(v)\}) \tag{1.1}$$

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1}) \tag{1.2}$$

where M_t and U_t may be arbitrary message and update functions respectively, $\bigotimes_t$ is a set aggregation function, e_{uv} is the state associated with edge (u, v), and $\mathcal{N}(v) = \{u, \forall (v, u) \in E\}$, see Figure 1.7.

Following Wang *et al.* [24], we deviate from the definition given in [23], by generalizing $\bigotimes_t$ to arbitrary aggregation functions instead of only summation. However, contrary to existing works, we also relax the requirement that M_t and U_t be differentiable to allow any arbitrary functions. While differentiability is desirable, or even essential in many deep learning/optimization situations, it is not a necessity for message passing in general and is not required in the modelling approach explored in Chapter 5. We provide justification for this choice of framework and further detail on its universality in Section 1.4.

We define *relational structure* as the edges E which represent the relations between entities and thus prescribe the connections by which messages may be sent/received.

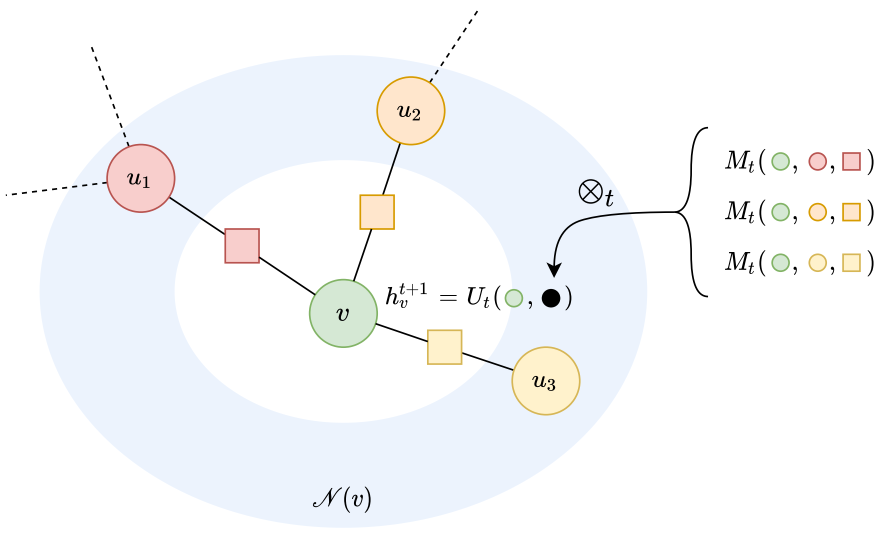


Figure 1.7: Message Passing: For incoming messages to vertex v, first the message function M_t is applied to each vertex and induced edge in the neighbourhood of v. The results are aggregated by $\bigotimes_t$, then the vertex state is updated according to U_t. This process is applied to all vertices at each time step $t \in 1, \ldots, T$.

Adapted from [25], we define *modelling* as the translation of problems from an application area into tractable mathematical or computation formulations, whereby analysis or execution of the models provides insights, answers or guidance useful for the originating application.

Finally, we define *non-Euclidean data* as data whose underlying structure is not Euclidean (not defined in n-dimensional linear space). For example graphs and manifolds.

Since the field of MPNNs has progressed so quickly since this work was started, we consider *state of the art methods* at the time each piece of research was conducted, and while we may review more current methods in Chapter 2, we do not perform comparisons against methods published after the creation of this work.

1.4 JUSTIFICATION FOR MPNN FRAMEWORK

In the past few years there have been an abundance of GNN architectures proposed with various convolution operators, *i.e.* [17, 18, 26–31]. The variety of notations and frameworks makes it difficult to compare and contrast these methods and to understand their nuances despite their commonalities. To solve this problem, [23] proposed the MPNN framework and reformulated a substantial number of existing methods to follow their framework. Recognising the potential for a unified language for all GNN architectures, many recent works have already adopted this framework [32–34].

An important property when choosing a message passing framework is of course its generality, *i.e.* how general is the framework and what can it (not) describe. The MPNN framework, under certain conditions, has been shown to be Turing universal [35]. This means that a MPNN can compute any function on its input that is computable by a Turing machine, providing the following conditions hold:

- Each layer must be sufficiently powerful, *i.e.* the message and update functions should be general vector to vector functions such as Multi-Layer Perceptrons (MLPs)
- The network must have sufficient depth (at least as many layers as the graph diameter) and width (units per layer is unbounded)
- Each vertex should have access to discriminative attributes that identify it uniquely.

In practise the width and depth requirements for universality may not always be tractable; however, we are adopting this framework for its universality in describing existing algorithms, thus theoretical universality is sufficient.

While we make some generalisations to the MPNN framework (detailed in Section 1.3), we note that these are not necessary for universality, but purely for simplicity and ease of notation, and allow us to include non-differentiable functions beyond neural networks. We also note that extension to directed graphs and those with temporal edge state are trivial [23], and asynchronous message passing is possible to express, simply with the insertion of additional time steps.

1.5 METHODOLOGY

To realise the aim outlined in Section 1.3, we consider precisely three practical use-cases: First, we consider how to utilize message passing to perform graph level representation learning in the context of chemical property prediction for drug discovery. We then investigate the use of message passing in performing geometric style representation learning for industrial and artistic Computer Aided Design (CAD) tasks. Finally, given the successes of message passing in data-driven settings, we consider a novel message passing based approach to modelling dynamic biological systems.

These specific use cases are chosen in order to cover a range of non-Euclidean data. First we consider molecules, which while existing in 3-dimensional Euclidean space everywhere in nature, require arbitrary ordering to represent in computer memory, thus presenting the challenges of permutation invariance discussed in Section 1.1. Second, we consider Boundary Representations (B-Reps) which contain a mixture of adjacency information and non-manifold boundary information such as parametric surfaces. Finally, we consider the case of a dynamic domain, specifically a dynamic gene regulatory network, which evolves over time as a function of its own stochastic configuration.

For validation, we focus on empirical evaluation of our methods, adopting a range of quantitative and qualitative metrics as appropriate in each use case. Further details for each use case are given below, and further justification for our choice of validation metrics are provided in each corresponding chapter (Chapters 3–5).

1.5.1 *Chemical Property Prediction for Drug Discovery*

We consider the task of supervised learning for the prediction of molecular properties. Drug discovery is typically a very expensive process, in which thousands of possible drug candidates need to be screened for toxicity and other factors that may deem them unsuitable for medical use. Conventional processes for drug candidate screening involve many expensive wet lab procedures. Thus, efficient methods for determining the properties of a given chemical compound are desirable, both in terms of reducing costs and speeding up the process.

The field of GNNs and MPNNs has progressed very quickly over the last few years. While there are now an abundance of vertex and graph level network architectures achieving state of the art results on a variety of tasks, the earliest networks were typically focussed on vertex level tasks, and extended to graph level tasks using global uniform aggregations of vertex representations such as summation [36], or mean pooling [17].

In this work, we investigate the use of message passing in order to perform more sophisticated global aggregations, whereby the relative importance of each vertex in the final representation can be learned.

In this context, the relational structure is prescribed by the bonds between atoms, and we focus on discriminative graph classification. We consider a range of common chemo-informatic benchmarks and perform evaluation based on the standard metric of classification accuracy. For baseline we compare against a range of state of the art methods.

1.5.2 *Geometric Style Similarity*

Quantifying and learning representations for geometric style is of relevance to many CAD tasks. For example, a car manufacturer may wish to ensure their new design is in-keeping with their existing range. They may wish to identify specific parts that do not match, and adjust them accordingly, or they may wish to search a database of exterior parts that best match the style of parts already included. In this use case, we investigate the use of MPNNs in defining a data-driven style similarity metric that may be used for these purposes.

We consider B-Reps (the industry standard for CAD), as well as mesh and point cloud representations. In the context of B-Reps, relational structure is defined by face adjacency. For meshes, the relational structure is the edges between vertex samples, *i.e.* it is dependant on the sampling strategy of the particular meshing algorithm used. For point clouds, any relational structure must be inferred. As such, we hypothesise that the semantic significance of the face adjacency information in B-Reps makes them more suitable for style metric learning.

To evaluate our method quantitatively we use linear probing and precision@k on a dataset of extruded characters from different fonts, and binary classification weighted-F1 scores on manually labelled CAD models. For qualitative evaluation we perform top-k queries on both datasets, and visualize the gradients of our style distance metric on pairs of extruded letters.

1.5.3 *Biological Modelling for Healthcare*

In many healthcare and medical applications, due to a lack of data for training machine learning models, alternative methods are often sought, *i.e.* models that emulate complex biological systems which allow us to ask 'What if?' questions. These types of models, such as Agent-Based Models (ABMs), are typically based on behavioural observations of constituent parts of a larger system, which are then executed to allow the observation of emergent behaviours of the larger system and draw useful conclusions.

In this work we propose a novel message passing-based model of computation, that facilitates the modelling of complex dynamic systems. While our method is general for many applications, we consider in particular the application of modelling gene regulatory networks. In this context, relational structure is prescribed by the hierarchical composition of living tissue, cells, chromosomes, and genes. Note that in this work, as a necessary condition, we consider the case that the message passing graphs are dynamic, *i.e.* vertices and edges are created/removed during execution non-deterministically.

We validate this work according to the methodology of [37], as well as comparison against an existing biological model of cancerous tumour growth [38].

1.6 CONTRIBUTIONS

Here we briefly outline the contributions made by this work for the message passing and broader research communities:

- We introduce an attention-based structure-aware global pooling operator for graph classification — Chapter 3/[3]

- We demonstrate the effectiveness of our proposed approach on a range of chemo-informatic benchmarks — Chapter 3/[3]
- We show that our method outperforms state of the art graph classifiers on an isomorphism test — Chapter 3/[3]
- We create a challenging synthetic dataset for graph isomorphism testing — Chapter 3/[3]
- We demonstrate the interpretability of our proposed approach on a sample of chemical compounds from a widely used mutagenicity dataset — Chapter 3/[3]
- We show that the second order statistics (Gram matrix) approach used in 2D image style literature can be generalized to 3D shapes (B-Rep/mesh) using MPNNs — Chapter 4/[1]
- We adapt an existing dataset of 3D extruded fonts to strengthen the associated style labels and provide a useful tool for the design, debugging, and evaluation of 3D geometric style learning techniques — Chapter 4/[1]
- We demonstrate the advantages of using B-Reps for style representation learning over mesh/point cloud alternatives as a consequence of additional relational information — Chapter 4/[1]
- We introduce a novel normalization technique, which leverages relational structure (face groupings of samples) to improve style representation learning on B-Reps — Chapter 4/[1]
- We introduce a few-shot learning method for capturing a subjective end-user's definition of 3D style and demonstrate its effectiveness on B-Reps — Chapter 4/[1]
- We demonstrate the effectiveness of our proposed method quantitatively and qualitatively when no style or content labels are available for encoder pre-training — Chapter 4/[1]

- We introduce a novel message passing-based model of computation — Chapter 5/[4]
- We demonstrate the effectiveness of our approach in expressing the complex dependencies of biological systems necessary to model life-like systems — Chapter 5/[4]
- We show the benefit of our approach over a conventional method with respect to simplified post-analysis, by maintaining a non-Euclidean representation throughout simulation — Chapter 5/[4].

1.7 SUMMARY

Motivated by the history of success of message passing algorithms in Computer Science, and their advantageous property of permutation equivariance, the aim of this work is to improve on state of the art methods for modelling non-Euclidean data through leveraging relational information with novel message passing techniques.

We have formally defined message passing according to a relaxation of the MPNN framework [23], and justified the use of this framework through its Turing universality and community adoption.

We have focussed this work to three important non-Euclidean modelling use cases and outlined the key contributions we make in each: chemical property prediction for drug discovery (Chapter 3) in which we propose a new structure-aware global pooling operator for graph classification, geometric style similarity for CAD (Chapter 4) in which we propose a few-shot learning method for a 3D style metric using MPNNs, and biological modelling for healthcare (Chapter 5) in which we introduce a novel message passing-based model of computation able to capture the complex dynamic dependencies of biological systems and trace the evolution of emergent gene distributions.

2 LITERATURE REVIEW

One of the most significant challenges to reasoning with relational structure is that the relations found in nature are non-Euclidean. The relations in any physical, biological, or even conceptual system may be expressed as a graph, and as such there is no global information such as ordinality or a common coordinate system [39]. For example, in B-Reps 3D solids are modelled by the adjacencies of non-manifold partial entities whereby partial edges and faces are parametrized according to their own local coordinate systems [40], and for molecules there is no global information to indicate the top or bottom, or to provide a canonical order for the atoms.

Message passing algorithms offer one approach to working with non-Euclidean data and thus reasoning about relational structure, however they are not the only possibility. In this section, we first consider the challenges to relational modelling for both single- and multi-domain settings, we then consider how these challenges are tackled by both message passing and non-message passing methods. Finally, we discuss some limitations to message passing algorithms, and provide a brief overview of common non-Euclidean applications.

2.1 CHALLENGES IN MODELLING NON-EUCLIDEAN DATA

Non-Euclidean modelling problems can be generally split into two categories according to whether they are concerned with a single (generally fixed) domain, or multiple domains. The single domain case is usually concerned with predicting vertex or edge signals on a fixed graph, for example identifying fraudulent behaviour in financial networks [41], predicting unknown links in a protein-protein interaction network [42], and question answering using a knowledge graph [43].

For the case of multiple domains, the most typical task is to predict graph level signals, for example the mutagenicity of a molecule [44], or to label a computer program as malware [45]; however, there are also many cases where vertex and/or edge level signals may be predicted when working with multiple domains. Examples of this include finding shape correspondence [46] and protein interface prediction [47].

2.1.1 *Single Domain*

As discussed in Section 1.1, permutation equivariance is a critical property for relational reasoning. When making predictions for vertex or edge signals on a domain with relations that carry no natural ordering, the predictions should not depend on the order in which the known relations are presented. For example, in a social network we may wish to identify bots. The predictions made should of course not be affected by the order in which the users are considered, nor should they depend on the order in which each user's friends or posts/activities are considered.

Permutation of entities and relations is however not the only important consideration, it can often be necessary to consider other pairwise relations. For example, in a given set of objects, each object may be affected by the relations between other pairs of objects in the set [48]. In our social network bot detection example for instance, we may wish to consider the connections between a user's friends to inform our decisions about that user, *i.e.* "do any of this user's friends know each other or are they all disconnected?" Of course, this concept may be extended to even higher-order relations such as larger rings of connected people for the detection of terrorist cells [49] etc.

Furthermore, it is important to consider that connectivity may vary greatly across the domain. In our social network example, this could mean that some users will have very many friends, while others may have few. Thus, it is important that we can handle and compare sets of different sizes with different numbers of relations within them.

2.1.2 *Multiple Domains*

In addition to the challenges of single domain modelling, in order to predict graph level signals requires the ability to handle inputs of different sizes, *i.e.* we require methods that can compare graphs with varying numbers of vertices and/or relations, whether through explicitly mapping them to some fixed size representation [21], or by reasoning implicitly through Kernel methods [50], without loss of the important, task-dependent information present in their relations.

It is clear that permutation invariance is critical to enable generalization across different domains, but again it is not the only consideration.

In Euclidean settings such as 2D image classification, it is trivial to scale, crop, or pad images to match in size [51]. However, for non-Euclidean data such as graphs, the analogies are not so simple. For example, up-scaling can be understood as representing an image on a large bitmap from original data sampled on a smaller grid while preserving the relevant image features [52]. In such a case, it is known a priori that the new image will be a 2D grid, and the task is to find the pixel values that preserve the image features. However, for a graph, the up-scaled structure is not known. The key difference being that 2D images are discrete representations sampled from underlying continuous data, whereas graphs are discrete structures which represent underlying discrete data.

One of the greatest strengths of conventional 2D CNNs is their ability to compose learned features hierarchically in order to learn higher-order features at increasing scales moving from lines and corners to complete objects [53]. This hierarchical prior is built into their architecture by pooling (or down-sampling) layers, in which an input is projected into a coarser domain [54]. In the Euclidean setting of 2D images, the global properties such as boundaries and pixel order make it trivial to perform such operations, however coarsening graphs is much more challenging, and there are as of yet no standard methods for doing so [39].

2.2 ALTERNATIVES TO MESSAGE PASSING

2.2.1 *Set Processing*

In [55], Zaheer *et al.* proposed that a permutation invariant function f on the set $\mathbb{X}$ may be learned indirectly through decomposition of the form

$$f(\mathbb{X}) = \rho\left(\sum_{x\in\mathbb{X}} \phi(x)\right), \tag{2.1}$$

providing suitable transformations ρ and ϕ can be found. This is precisely the way in which many Siamese networks work [56], and this idea in fact underpins how almost all MPNNs work in how vertex neighbourhoods are aggregated.

Relating this to our message passing framework in Section 1.3, we see that Equation 2.1 is a particular instance of the message function in Equation 1.1, where as with the majority of MPNNs, the aggregation function $\bigotimes_t$ is summation, M_t is ϕ, and $\{(h_v^t, h_u^t, e_u v), u \in \mathcal{N}(v)\}$ is the set $\mathbb{X}$. In this instance, ρ becomes part of the update function (Equation 1.2), and would typically include some normalization factor and non-linearity.

This idea is further specialised in [57]as Janossy Pooling, where ρ is a normalisation function, and the summation occurs over the set of all possible permutations of the input set. Due to the factorial complexity of enumerating all possible permutations, the authors propose the use of permutation sampling and canonical orderings to provide a trade-off between learnability and computational cost.

More recently, [58] demonstrated that when working with sets of elements which contain their own symmetries, the expressivity of the architecture in [55] is unnecessarily restricted. In particular, [58] show the benefit of summing a shared representation of all other elements in the set to each processed item, *i.e.*

$$L(\mathbb{X})_i = L_1^H(x_i) + L_2^H\left(\sum_{j\neq i}^{|\mathbb{X}|} x_j\right), \tag{2.2}$$

where L_1^H and L_2^H are linear H-equivariant functions, and H is the transformation group of interest, *i.e.* the group of circular translations in the case of a set of images.

While set-based methods provide permutation equivariance/invariance, they fail to capture pairwise relations between other members of the set, and as such offer limited expressive power for processing graphs of relational structure.

2.2.2 *Spectral Methods*

Spectral Graph Theory has provided a theoretical grounding for many message passing algorithms, *i.e.* [17, 59], and prior to deep learning provided state of the art methods for the analysis of relational structure. For example, Spectral Clustering [60] leverages relational structure through the eigenvectors of the graph Laplacian matrix, $L = D - A$, where A is the Adjacency matrix, and D is the diagonal degree matrix such that $D_{ii} = \sum_j A_{ij}$.

In addition to providing a foundation for many graph kernels, for example random walk-based [61] and diffusion based [62], spectral methods have also inspired many approaches to the generalization of conventional 2D CNNs to non-Euclidean data [63].

The first case of this is seen in [64], where the spectrum of the graph Laplacian is used to define filters based on the relational structure of the input data. Following from this, [65] showed that recurrent Chebyshev polynomials could be used to avoid the explicit computation of the Laplacian eigenvectors required in [64]. Further simplifications were found in [17], in which simple filters are defined purely in terms of the local 1-hop neighbourhood of each vertex — as a message passing method, we discuss this in further detail below.

While spectral methods have provided a theoretical foundation for the analysis of relational structure and inspired many highly successful message passing methods, a serious drawback to their use is that any spectral definition of a convolution operator is dependent on the Fourier basis, which is of course dependent on the domain [39]. What this means is that the transfer of filters learnt in one domain are not trivially applicable to other domains, *i.e.* these techniques are less suitable for tasks such as graph level classification, or graph transfer learning.

2.2.3 *Language Inspired Methods*

Successes in natural language processing techniques have also inspired many methods for representing relational structure. Deepwalk [66] learns vertex representations by sampling random paths throughout a graph, then minimizing the log probability of predicting the vertex co-occurrence, analogous to Word2Vec [67] with word co-occurrence in sentences. This idea is extended in [42], where the random paths are directed with additional parameters which control the likelihood of remaining local to the start/current vertex.

As with the spectral methods discussed above, these approaches are targeted at the single-domain cases. Moreover, these methods provide a technique for learning homophily, *i.e.* vertices that are close in terms of geodesic distance will receive similar representations, but not for learning structural similarity (see Figure 2.1). Ribeiro *et al.* [68] propose an extension to [66] in which the input graph is first transformed into a multi-layer graph where each layer is weighted according to structural similarities at an increasing number of hops. This results in a graph whereby the geodesic distances of vertices now represent the structural similarities of the original graph rather than homophily. In practise however, this is very computationally expensive to compute, both in memory footprint and additional processing time.

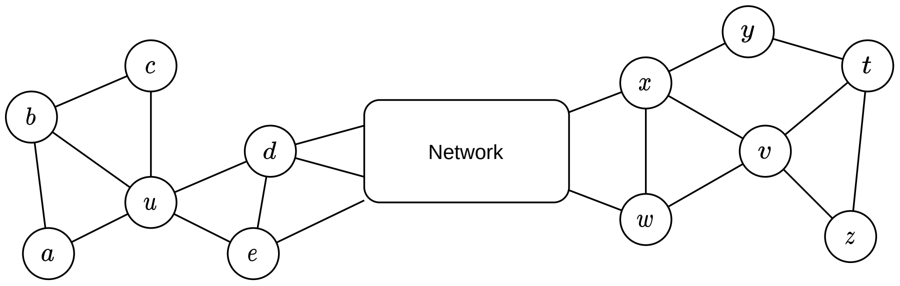


Figure 2.1: Vertices u and v are structurally similar (degrees 5 and 4 connected to 3 and 2 triangles, connected to the rest of the network by 2 vertices), but are far apart in the network. Figure from [68].

Recurrent Neural Networks (RNNs) have also been used in several methods for learning relational structure. For example, [69] propose the Graph Attention Model in which an RNN is used to select an informative sequence of vertices visit, at each step updating its current global graph representation based on the new information available and its previous state. GraphRNN [70] is another case, however this time the RNN is used to predict the addition of vertices to a base graph for the generation stage of an autoregressive model. In addition to the problem that GraphRNN loses permutation invariance on larger graphs [71], RNNs require a very large number of parameters and as such are expensive to train.

Following the success of Transformers in Natural Language Processing (NLP) settings [72, 73], there have been several attempts to generalize them for use with relational data, *i.e.* [73–77]. While it is possible to argue that Transformers are already GNNs [78], in the fact that they operate over fully connected graphs, it is this very same point that places them in direct opposition to MPNNs.

By assuming a fully connected graph, graph-based Transformers are able to identify long-range dependencies, however fail to leverage the sparsity of the existing relational structure which provides a strong inductive bias and enables efficient computation. In order to leverage the existing relational structure, graph-based Transformers must either represent existing edges with special edge types or provide some other method for structural encoding, and to capture the relative positions of vertices, graph Transformers must adopt some form of positional encoding.

Sequence Transformers use sinusoidal functions added to the input features to act as an absolute positional encoding [79], while generalizing to sequences longer than those seen at training time. However, defining positional encodings for vertices or edges in a graph is a challenging problem, since relating back to the issues of canonical labelling discussed in Section 1.1 and 2.1.2, there is no concept of absolute position within graphs that generalizes across different domains [80].

While sinusoidal functions do not trivially extend to non-Euclidean data, Dwivedi *et al.* [75] have shown that positional vertex encodings based on the eigenvectors of the Laplacian are somewhat analogous, and despite not generalising across different graphs as a true absolute encoding, they are shown to be effective in practice. The analogy becomes evident when considering a sequence as a line graph for which the Laplacian eigenvectors would produce an equivalent sinusoidal encoding [81].

An alternative approach is to use relative positional encodings. For example, analogous to the relative positional encodings of [82], Mialon *et al*. use the distance between vertices, which is computed according to some kernel (*i.e*. Random Walk, Diffusion, etc.) and then used to bias the attention scores given given by the attention heads [80].

Although positional encoding is often associated with Transformers, it is also interesting to consider the effect of positional encoding on MPNNs, which are also not inherently position-aware, i.e. two vertices which share local structure will receive the same embeddings regardless of their position in the graph. For example, You *et al*. [83] have proposed Position-aware GNNs, in which multiple random subsets of *anchor* vertices are first sampled and the distances to all vertices are computed. These distances are then used to weight a non-linear aggregation function between the anchors and all other vertices forming a position-aware embedding. By considering this positional information, the authors show that it is possible to distinguish between vertices that share local structure, and demonstrate an improvement on many link prediction and vertex classification tasks on a single domain. Dwivedi *et al*. [81] also show that while Laplacian eigenvector encodings are more effect, even random indexing for positional encoding can also improve the performance of MPNNs on many tasks.

While graph-based Transformers have achieved state of the art performance on many tasks [74, 75], they bring a great cost in terms of number of parameters and hyper-parameters, the assumption of fully connected graphs, and difficulty in training, *i.e.* requiring complex training schedules [84].

2.2.4 *Vision Inspired Methods*

In addition to the spectral generalisations of CNNs [17, 64, 65] discussed above, it can of course be argued that all GCNs, and even MPNNs, are inspired by the successes of traditional CNNs on 2D images. To this end, we discuss MPNNs below in Section 2.3, and consider only the non-message passing alternatives here.

Patchy-SAN [21] is one such case, where rather than generalizing the CNN architecture to fit non-Euclidean relational data, the data is instead first converted into a Euclidean form. To do this, the input graph(s) are labelled according to a canonical labelling algorithm (although these may be approximate) to prescribe an order over the vertices. A sequence of vertices is then sampled according to a given stride, and local neighbourhoods are sorted (again with canonical labelling algorithms) before being sampled or padded to fill a fixed size. The result is that each graph is represented by a fixed size matrix where each row indicates a vertex and its local neighbourhood.

In [6], we extended the work of [21] to employ the use of Capsule Networks [85] instead of CNNs, an idea also seen in [86]. As a non-message passing method, we do not present this work as part of this thesis.

The problem with all of these methods however is that the process of explicitly mapping the varying sized inputs to some fixed size representation (involving sampling) is not driven by the downstream task. What this means is that the ordering of vertices and their sampling cannot be optimized for the task that is to be solved in an end-to-end fashion, and so important information may be lost.

Another approach to leverage the power 2D CNNs can be seen in works that project higher dimensional objects into multiple 2D images. For example, to predict protein structural classes, Nanni *et al.* [87] take 13 different types of 3D protein visualisations and process each one with a different CNN. Due to the non-Euclidean nature of this data, *i.e.* the orientation of a protein is arbitrary, they rotate each representation uniformly about the x, y, and z axes in order to capture 125 images of each protein. A similar technique is used in [88], where 3D shape classification is performed using a set of 12 2D views, each one rotated about the z axis.

While it reasonably efficient to render objects such as B-Reps and meshes into 2D views and process them with traditional 2D CNNs, rotation of these views about only a single axis requires consistent axis orientation across all the data, which cannot always be assumed. For example, the ABC dataset [89] contains some y-up and some z-up data, without labels for which is which. In such a case, additional views with rotation in more axes would be required, thus significantly increasing the complexity.

Moreover, the best viewing angles may be task dependant. For example, in the case of style representation learning of cars, viewing angles of the front, sides and back (*i.e.* where the cars are usually observed from) would be much more important than from the top or underneath. And for more complex structures such as proteins and large molecules, it may not be possible to see all parts of an object no matter which viewing angle is selected.

2.3 MESSAGE PASSING METHODS

Having considered a variety of alternatives to message passing and the challenges they face, we now review message passing algorithms including some notable graph kernels, and in particular MPNNs. In many works MPNNs, and often GNNs in general, are categorized according to their expressive power in terms of the Weisfeiler-Lehman (WL) isomorphism test [29, 90–92], thus we provide a description of this first.

2.3.1 *Weisfeiler-Lehman Isomorphism Test*

The WL isomorphism test is an heuristic message passing algorithm for approximating the isomorphism of multiple graphs [93]. It does so by providing a canonical form for a given input graph. Multiple graphs may be reduced to their canonical form in parallel, and if at any stage the canonical forms differ, then the graphs are known to be non-isomorphic.

Correspondence of two graph's canonical forms, however, is a necessary but insufficient condition to indicate isomorphism, *i.e.* if the graphs have different canonical forms, they are guaranteed to be non-isomorphic, but if they have the same canonical form, they are only *possibly* isomorphic. In other words, there exist non-isomorphic graphs that share a canonical form (failure cases). In practice however, failure cases for the WL test are few, and from a random graph perspective, the WL test works for virtually all graphs [94].

Figure 2.2 shows an example graph reduced to its canonical form using the WL isomorphism test, the steps can be described as follows:

1. Label all vertices with the same colour (*i.e.* a discrete label).
2. Update each vertex v's colour $c(v)$ with an injective hashing function on the tuple $(c(v), \{\{c(u), u \in \mathcal{N}(v)\}\})$, where $\{\{\ \}\}$ indicates a multi-set (an unordered set which may contain repeated elements), and $\mathcal{N}(v)$ is the 1-hop neighbours of vertex v.

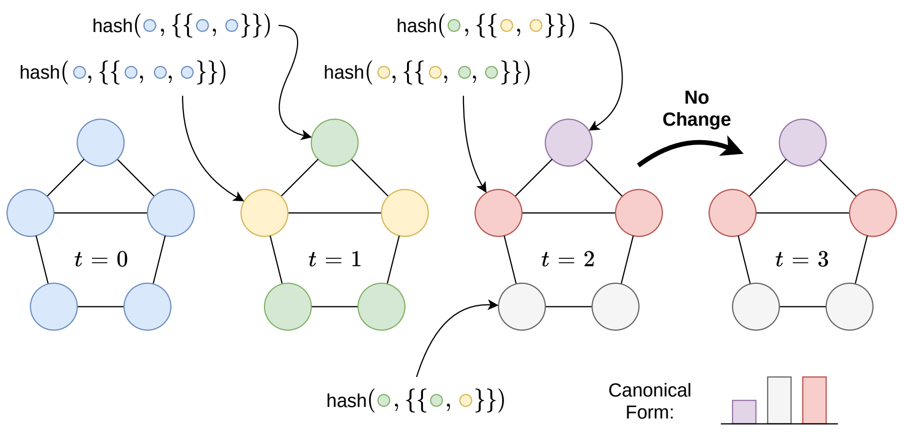


Figure 2.2: Example of the WL isomorphism test: In this case convergence is reached after 3 iterations, thus the canonical form for this graph is found. Figure adapted from [95].

3. Repeat step 2 until convergence. Convergence is guaranteed in at most $|V|$ steps, where $|V|$ is the number of vertices.

As first proposed in [91], through the types of graph structures it can and cannot distinguish, the WL test provides a useful bound for which to classify and compare the expressivity of MPNNs, and more generally GNNs.

To consider bounds beyond the WL test, higher order tests have been proposed. For example, the k-WL [96] test defines a similar colouring procedure over k-tuples of vertices, thus increasing the expressive power but also the computational complexity. Since non-local k-tuples of vertices are considered, pure message passing algorithms are unable match the power of k-WL tests beyond $k = 2$ without some form of pre-processing or data-augmentation (discussed further in Section 2.4). Note, the expressive power of k-WL tests is strictly increasing as k increases, with the exception of 2-WL tests which are equally as powerful as 1-WL tests (the variant presented above). For this reason, we do not provide further details on higher-order WL tests, but provide further discussion on the limitations of the 1-WL bound that message passing algorithms face below in Section 2.4.

2.3.2 *Graph Kernels*

Graph kernels have largely been superseded by deep learning models, however, we consider the WL Subtree Kernel [50] here due to its relevance to message passing and the fact that, because of its expressive power and runtime efficiency, it still frequently appears as a baseline for new state of the art methods, *i.e.* [76, 86, 91].

The WL Subtree Kernel, as its name suggests, is based on the 1-WL isomorphism test; and at its introduction in [50], demonstrated massive improvements over existing state of the art methods, in both computation time and accuracy. With existing methods based on features such as counting graphlets [97], shortest paths [98] and random walks [79], the use of the highly scalable WL test to generate features has obvious computational advantages since it is trivially parallelisable, and has such low memory footprint, that many graphs can be computed at the same time.

The authors demonstrate a runtime complexity of $O(Nhm + N^2hn)$, where N is the number of graphs, h is the number of iterations required, and n is the number of vertices. Comparing this to the $O(N^2n^4)$ complexity of the Shortest Path Kernel of [98] for example, we see a factor of approximately $O\left(\frac{Nn^3}{h}\right)$ improvement.

2.3.3 *Spectral Inspiration*

We discussed some significant spectral methods above in Section 2.2.2; here we focus on message passing methods developed to exploit useful spectral properties.

Power Iteration Clustering (PIC) [59] is one such algorithm, that provides a highly scalable approximation to the eigendecomposition of spectral clustering. Rather than requiring explicit computation of the eigenvectors of the graph Laplacian, PIC initialises a random vertex state vector $v \in \mathbb{R}^{|V|}$, where $|V|$ is the number of vertices and v_i indicates the state of vertex i. This vector is then repeatedly updated for T steps, such that each vertex's new state is the normalized weighted average of its 1-hop neighbours, where the weights are defined by the graph or assumed to be 1 if the graph is unweighted.

As $T \to \infty$, v converges to a constant vector corresponding to the largest eigenvalue of the Laplacian; however, more usefully, the rate at which each component of v converges is related to the smaller eigenvalues. Thus, if the algorithm is stopped before convergence, the resulting v will be largely piece-wise constant, with each 'piece' corresponding to clusters within the graph.

Not only does PIC significantly reduce the complexity of spectral clustering through the use of message passing updates, the authors also demonstrate greater robustness to noise [59].

One of the most influential graph algorithms of the last decade, the GCN as proposed in [17], is a spectrally inspired graph convolution architecture that uses only information from 1-hop neighbourhoods in each vertex update layer. Therefore, while not originally expressed as a message passing algorithm, it can be computed as such, and comes with all the advantages of a message passing algorithm.

In [17], a graph convolution is defined by the layer-wise update rule

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{2.3}$$

where $\tilde{A} = A + I$ is the adjacency matrix with added self-loops, I is the identity matrix, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, $H^{(l)}$ is the vertex states at layer l, $W^{(l)}$ is a learnable weights matrix, and σ is a non-linear activation function. Considering this in a vertex update form

$$h_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}(i)\cup\{i\}} \frac{1}{c_{ij}} h_j^{(l)} W^{(l)}\right), \tag{2.4}$$

we see the parallels between this and Deepsets [55] (see Equation 2.1), demonstrating the invariance to permutation of local neighbourhoods, and thus graph level permutation equivariance, which is an essential property for any single- or multi-domain graph algorithm. As shown in [23], the GCN [17] also fits our message passing framework in Section 1.3, with a message function of

$$M_t(h_v^t, h_w^t) = L_{vw} h_w^t \tag{2.5}$$

and an update function of

$$U_t(h_v^t, m_v^{t+1}) = \sigma\left((W^t)^\top m^{t+1}\right) \tag{2.6}$$

where L is the graph Laplacian, $\top$ is the matrix transpose, and σ is a non-linear activation function.

As with the spectral methods discussed in Section 2.2.2, both PIC and the GCN [17] are intended for use within a single domain. The GCN was in fact intended as a semi-supervised method, *i.e.* in transductive settings. Despite its dependence on the domain dependent Fourier basis, the GCN has in fact been shown to be successful in inductive settings such as graph classification across domains, *i.e.* [99, 100].

2.3.4 *MPNNs*

The MPNN framework was proposed in [23], however as demonstrated by the authors, the GCN [17] discussed above and many other existing works can be understood as specific MPNN instances. Another early example of this can be seen in Gated Graph Neural Networks (GG-NNs) [26], which, inspired by RNNs in NLP settings, perform local vertex updates based on Gated Recurrent Units (GRUs) [101].

The use of GRUs is significant, since they allow a model to learn when to retain previous state, and despite their use in some of the earliest MPNNs, they are still widely used in state of the art methods [23, 32, 57].

At roughly the same time as [23], Hamilton *et al*. proposed GraphSAGE [27] as an alternative framework for vertex feature learning. Contrary to existing transductive works, GraphSAGE was designed with inductive learning in mind, and achieved state of the art performance on vertex classification tasks across multiple domains. In order to generalise to unseen graphs, GraphSAGE learns a set of aggregator functions that each learn to aggregate information from a different number of hops, and for scalability vertex neighbourhoods are sampled to a fixed size in order to enable fixed size batches.

For unsupervised learning, the authors provide a random walk based learning objective that enables homophily based embeddings to be learned without labels. However, since this objective is based on vertex co-occurrence in short random walks, the learned embeddings do not reflect structural similarity for distance vertices, thus are not useful for tasks such as molecule classification or protein interface prediction etc.

Addressing the issue of learning vertex representations in unsupervised settings where structural information is important, [102] proposed Deep Graph InfoMax (DGI) as a generalization of Deep InfoMax [103] to non-Euclidean data. The general principle behind these methods is the maximization of mutual information between local and global representations of the input data, which forces the network to learn global summaries which capture the local details, and in the context of DGI allows distant vertices with shared local details (*i.e.* subgraph structure) to recieve similar representations.

While effective in learning unsupervised vertex representations, due to the use of the transductive GCN [17] as its encoder and mean pooling for global representations, DGI is not so effective for graph level representation learning. These shortcomings are addressed in [104], where the inductive Graph Isomorphism Network (GIN) encoder [91] (discussed below) is adopted and sum pooling is applied, demonstrating improvements in downstream tasks such as graph classification.

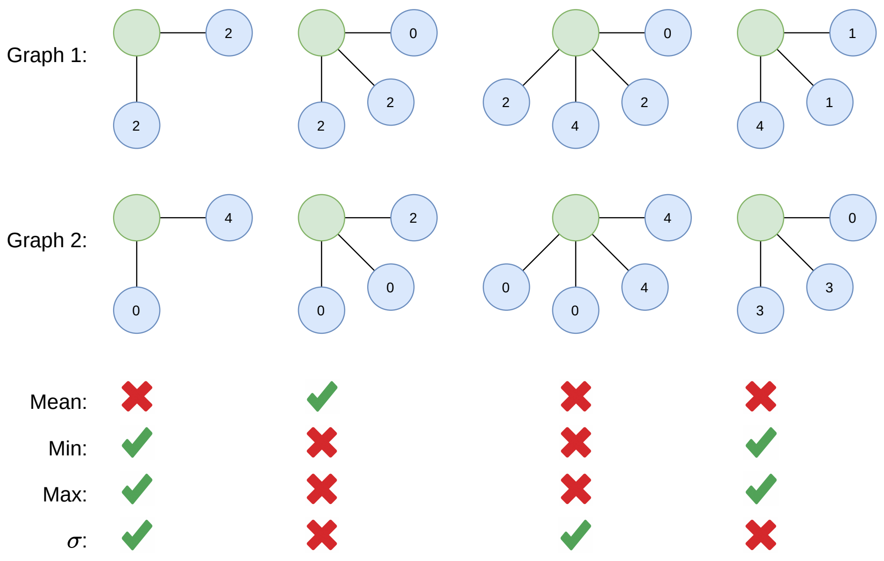


Figure 2.3: Examples of failure cases for common neighbourhood aggregation functions: Incoming messages for each green vertex indicated on blue neighbourhood vertices. Checkmarks indicate the ability of each aggregator to differentiate between graph 1 and 2. σ is standard deviation. Figure adapted from [32].

GIN was proposed in [91] in response to the observation that many popular neighbourhood aggregation functions such as mean, sum and max are not injective multi-set functions, meaning that vertices with different neighbourhoods could receive the same representations (see Figure 2.3). To address this issue, Xu *et al.* [91] extend the DeepSets result [55] and show that any function g over the pair (c, X) may be decomposed as

$$g(c, X) = \varphi\left((1 + \epsilon) \cdot f(c) + \sum_{x \in X} f(x)\right) \tag{2.7}$$

and that for infinitely many choices of ϵ and some function φ there exists a function f such that g is unique for each pair (c, X), where $c \in \mathcal{X}$, $X \subset \mathcal{X}$ is a multi-set of bounded size, and $\mathcal{X}$ is countable.

To realise this result in GIN, Xu *et al.* [91] use MLPs for the learnable function φ and show that although the constant ϵ can also be learned, in practice setting $\epsilon = 0$ is equally effective. With an injective multi-set neighbourhood aggregator, this network is maximally powerful for a message passing algorithm (*i.e.* equally as powerful as the 1 or 2-WL test) in determining non-isomorphic graphs [91]; however, is limited to discrete vertex features [32]. To extend this result to continuous vertex features, Corso *et al.* [32] propose Principle Neighbourhood Aggregation (PNA) whereby they use an outer product of multiple aggregators and degree-based scalers (identity, amplification, attenuation).

Inspired by advances in with self-attention in sequence modelling architectures [72, 105, 106], Veličković *et al.* proposed Graph Attention Networks (GAT) [30]. Due to the ability to learn which neighbours of a vertex should be aggregated into the new vertex state, GAT provides a more powerful means for inductive vertex representation learning compared to equally weighted neighbourhood aggregators such as GCN [17] or GraphSAGE [27], which the authors demonstrate on a range of transductive and inductive benchmark tasks. As a message passing algorithm operating purely on 1-hop neighbours, GAT is computable in parallel for all vertices at once, thus remaining efficient to compute.

The use of self-attention is extended further in [107], whereby additional attention coefficients are added to the multiple attention heads of [30], providing further gains in inductive classification tasks on multiple domains.

The methods discussed thus far in this section, as per the majority of research on GNNs and MPNNs, were largely designed for vertex feature learning. For graph level representation learning, any vertex learning method may be used followed by a global pooling method such as mean [36] or max pooling [65] to provide a permutation invariant graph representation. However, such functions ignore the structure of the graph [108].

This issue is precisely the motivation for our work PiNet [3], which will be presented in detail in Chapter 3.

Addressing the same problem, [109] proposed a sort pooling method, in which learned vertex representations at each layer are sorted with only the top k being carried through to the next layer. Seeking to leverage hierarchy and mimic the pooling process of conventional 2D CNNs, [110] proposed DiffPool, in which at each layer learned vertex representations are clustered, with each cluster forming a vertex in a new coarser graph. At each layer the graph becomes coarser until finally some global pooling method is applied. A similar idea is seen in Dynamic Graph CNN (DGCNN) [24], which although intended for use on point clouds, aggregates points at each layer according to proximity in the learned feature space, again defining a new graph for each layer.

Finally, in [111] Li *et al*. proposed an alternative graph level representation learning method based solely on message passing, in which a 'dummy super vertex' is added to the graph and connected to all other vertices with a directed incoming edge. While in [111] the authors also propose specialized convolutions and pooling functions, the dummy vertex approach is general and has been shown to boost graph classification performance with many models [112]. The dummy vertex, which holds a state vector of greater dimensions than the 'real' vertices, enables global graph level information to be learned through aggregation of all vertex features during each layer of the network, while the real vertices learn purely local features.

2.4 LIMITATIONS OF MESSAGE PASSING

We discussed the theoretical Turing universality of message passing algorithms in Section 1.4; however, the conditions for universality are rarely met in practice. For example, both network width and depth must exceed polynomial functions of the graph size [35], and vertices must be uniquely identifiable which is not the case for graphs such as molecules in which atoms have no canonical ordering from which to assign consistent identifiers. Thus, MPNNs without unique vertex identifiers, referred to as *anonymous* MPNNs, are bounded by the 1-WL isomorphism test [29, 91]. Interestingly, however, [113] demonstrated that despite being bounded by the 1-WL test, degree aware MPNNs such as the GCN [17] and also our parametrized GCN extension in [3] are one step head of the 1-WL test, since the first round of message passing in the 1-WL test simply acquires the vertex degrees which are already available in degree aware methods, meaning that a 2 layer degree aware MPNN has the discriminatory power bound of 3 steps in a 1-WL test.

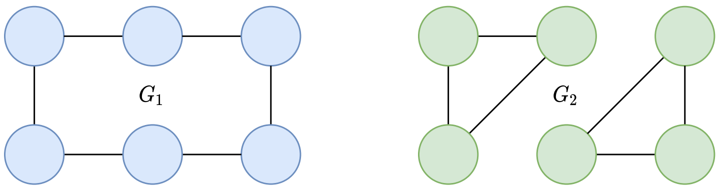


Figure 2.4: Example of two graphs not distinguishable by the 1-WL test, *i.e.* G_1 And G_2 are not isomorphic but share a canonical form. Figure adapted from [92].

While failure cases for the 1-WL isomorphism test are few, these failures are not limited to large complex graphs. For example, Figure 2.4

illustrates two very simple graphs that are indistinguishable to the 1-WL test, and thus any anonymous MPNNs.

This shortcoming of anonymous MPNNs has provided the motivation for many higher order (in terms of the k-WL test) GNNs, although these come at a higher computational cost, and while provably more powerful than 1-WL MPNNs, they do not always outperform the simpler and more efficient MPNNs, *i.e.* [91, 92], suggesting that for some tasks the increase in power is not worth the trade-off with model complexity.

The way in which these methods provide theoretical bounds greater than the 1-WL test varies greatly. For example, Beaini *et al.* [34] revisit spectral graph theory to use Laplacian eigenvectors to provide global direction over the input graphs, and Brossard *et al.* [33] extend the receptive field of each vertex beyond the 1-hop neighbourhood to enable the detection of small cycles.

Taking a different approach, Bevilacqua *et al.* [114] recently demonstrated that the expressive power of MPNNs can be increased beyond the 1-WL bound, through representing each input graph as a *bag of subgraphs*. The subgraphs are selected according to a pre-defined policy, *e.g.* one subgraph for each vertex in the input graph with all edges to that vertex removed, and then processed according to a set-based architecture such as DeepSets [55] or DSS [58] with a MPNN encoder. Thus use of a MPNN ensures permutation equivariance to vertex order (assuming subgraphs are indexed consistently), while the set-based architecture provides equivariance to the order of the subgraphs, with an architecture such as DSS [58] also taking into account shared information across all subgraphs (see Section 2.2.1). While considering all possible subgraphs for larger input graphs could become computationally intractable, the complexity of the encoder is not significantly increased, since unlike methods such as [33, 34], it preserves the locality and sparsity of computation afforded by its use of an MPNN encoder.

Moreover, the result extends beyond 1-WL MPNN encoders, such that this framework can increase the power of a 3-WL encoder to distinguish graphs indistinguishable by a 3-WL test.

To improve scalability and thus makes these models more useful in real-world settings, approximation and sampling methods have been proposed. For example, Murphy *et al.* [57] sample random permutations during training instead of considering all possibilities, Brossard *et al.* [33] sample different width receptive fields, and Bevilacqua *et al.* [114] sample subgraphs according to the chosen policy.

Additionally, we note that in cases such as molecules, positional information can increase the expressive power of MPNNs beyond the 1-WL test. For example, revisiting the WL failure cases shown in Figure 2.4, we see that by assuming the graphs exist in Euclidean space, simply labelling the vertices with the angles between the edges as shown in Figure 2.5 results in distinguishable graphs [115].

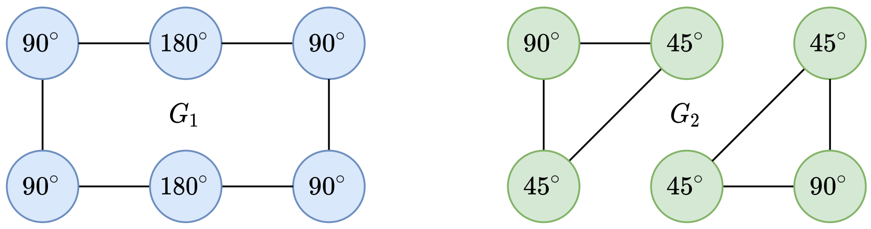


Figure 2.5: Example of two graphs that become distinguishable by the 1-WL test with the inclusion of positional information in the form of angles.

Another issue faced by MPNNs is that they are limited in depth, typically achieving best results with only 2 to 3 layers, *i.e.* [17, 27, 30]. This is contrary to their 2D image CNN inspirations, which frequently benefit from many more additional layers [14]. There are several reasons that MPNNs fail to benefit from greater depth including *over-smoothing*, in which all vertex representations tend toward the same value as depth increases [116], as well as increased likelihood of over-fitting, and vanishing gradients.

To overcome these issues, borrowing from early CNN literature, [116] proposed the use of residual connections and dilated convolutions. Other strategies have included new forms of normalization, for example Pairnorm [117] and MsgNorm [118]. However, while demonstrating significant improvement in performance, these deeper MPNNs are still outperformed by shallower alternatives on a range of benchmark tasks [112]. To tackle over-fitting, [119] proposed Free Large-scale Adversarial Augmentation on Graphs (FLAG) which is a flexible method compatible with any GNNs, which iteratively augments vertex features with gradient-based adversarial perturbations during training time, and demonstrated boosts in performance across a range of vertex and graph classification benchmark tasks [112].

2.5 APPLICATIONS

	Single Domain	Multiple Domains
Vertex Level	**Classification/regression:** Disambiguation [17], information retrieval (question/answering) [43], paper/blog subject prediction [27, 102] **Link prediction:** Recommendations (movies, products, recipes, friends, etc.) [120], protein interaction prediction [42] **Ranking:** Search engines [11] **Community detection (homophily clustering):** Consumer profiling [121], semi-supervised image classification [122] **Structural similarity:** Fraud detection [41]	**Classification/regression:** Protein interface prediction [47], protein function prediction [27, 30] **Correspondence:** Shape/image correspondence [46] **Structural similarity:** Bug (software) detection [123] **Our Contributions:** PiNet [3] (Chapter 3), UVStyle-Net [1] (Chapter 4)
Graph Level	**Network analysis (graph diameter, Hamiltonian circuit detection, spanning trees etc.):** Logistics (delivery, supply chains, routing etc.) [124], computer/communications networking [125] **Our Contributions:** IHDNs [3] (Chapter 5)	**Classification/regression:** Chemical property prediction [23, 44, 50, 91], malware detection [45], image classification [65, 126], graph isomorphism [93] **Similarity:** Shape retrieval [127] **Our Contributions:** PiNet [3] (Chapter 3), UVStyle-Net [1] (Chapter 4)

Table 2.1: Applications

Table 2.1 provides an overview of common non-Euclidean modelling applications, organised according to the domain(s) on which they operate, and whether they are for vertex level or graph level predictions.

At the time of starting this body of work, the vast majority of existing literature focussed on vertex level tasks on a single domain, with some attention turning towards inductive methods for vertex level tasks on multiple domains. Thus, we identify the need for improvements in graph level tasks on single or multiple domains.

In Chapter 3 we tackle the problem of utilizing the existing vertex level message passing methods to perform graph level predictions, in particular we focus on chemical property prediction for drug discovery, and as a vertex level task, we demonstrate the interpretability of our method in visualizing vertex attention coefficients. In Chapter 4 we again focus on graph level tasks (with a minor focus on vertex level gradient visualizations), with the application of geometric style similarity for B-Reps, for which there are no existing methods. Finally, in Chapter 5 we consider a graph and vertex level task on a single domain; however, unlike many of the existing methods we have discussed, our work simulates a dynamic domain.

2.6 SUMMARY

Reasoning with non-Euclidean relations and structure is challenging for several reasons. Foremost, is the requirement for permutation equivariance for vertex level algorithms, and permutation invariance for graph level algorithms. For working with multiple domains, it is also important to be able to represent and compare different numbers of relations and entities within a common space.

Set processing methods such as [55, 57] provide permutation invariance through symmetric aggregation functions, yet fail to capture higher order relations within the sets. Canonical labelling methods have also been applied to transform non-Euclidean data such as graphs into fixed sized tensor representations [6, 21], which can then be fed to traditional CNNs, however canonical labelling algorithms are either expensive to compute or approximate and thus may fail in particular cases. Moreover, to enable fixed size representations, these methods use sampling, which is not driven by the task at hand thus important information can be lost.

Language models such as [67] have been adapted for use on graphs [42, 66], with vertices replacing words, and random walks replacing sentences. These methods however are restricted to single domain modelling and are unable to capture structural equivalences [68]. Transformers [72] have also been applied to a variety of non-Euclidean problems [75, 76], yet by assuming a fully connected dense graph, these methods fail to leverage the sparsity of graphs thus significantly increasing computational complexity.

Permutation equivariant spectral methods [60–62], including non-Euclidean generalisations of 2D CNNs [17, 65], have been successfully applied to many single domain vertex level problems. However, their dependence on the Fourier basis render these methods difficult to apply in multi-domain settings [39]. Also inspired by the success of 2D CNNs, several inductive non-Euclidean CNN generalisations have been proposed [27, 30]. In particular, MPNNs overcome the problem of transferring to different domains, while remaining efficient to compute and offering permutation equivariance without the need for canonical labellings.

MPNNs have been shown to be Turing universal in theory [35], however in practice their anonymous formulations with limited network capacities mean such universality is not obtainable. For anonymous MPNNs, the WL isomorphism test provides an upper bound on expressive power with respect to distinguishing non-isomorphic graphs [91], however, the failure cases for the WL test are few, and the inability of provably more powerful, higher order methods to consistently outperform MPNNs [91, 92] suggests that the increase in power is not always worth the trade-off with model complexity. Moreover, labelling small cycles a priori can increase the expressive power, without an significant increase in model complexity [33].

While the WL test provides an upper bound for MPNNs, in order to meet that bound requires injective vertex aggregation functions [91]. Many existing MPNNs are built on the foundation of Deepsets for their aggregation functions, however for the case of graphs, many injective set aggregation functions are no longer injective when operating on the multi-sets prescribed by vertex neighbourhoods. To overcome this, several injective multi-set aggregation functions have been proposed [32, 91, 92].

While there have been many generalisations for convolutions to non-Euclidean domains, there has been less focus on how to pool vertex level representations into graph level representations. Based on the success of message passing in tackling vertex level tasks, there is a strong motivation for further work in addressing graph level problems.

3

PINET: ATTENTION POOLING FOR GRAPH CLASSIFICATION

The following chapter includes work published in [3]. The reviewers recognised that our approach "builds on previous work" and that the core idea is "intuitive and natural to explore". The main criticism of this work was that the proposed model "does not seem to outperform the chosen baselines" and that this work would be improved by identifying the benefits of our method. We addressed this with demonstration of the interpretability of the learned attention coefficients (see Section 3.4.5). We thank all reviewers for the contribution to this work.

3.1 INTRODUCTION

Starting with the multiple domain case for graph level tasks, here we consider graph classification — the task of labelling each graph in a given set, which has applications in many diverse application areas ranging from chemo-informatics [128] and bio-informatics [129], to image classification [130] and cyber-security [131]. In this work we focus in particular on the task of chemical property prediction for drug discovery.

In recent years, CNNs have led the state of the art in many forms of pattern recognition, *i.e.* in images [132] and audio [133]. Essential to the success of CNNs in representation learning is the process of pooling [134], in which a set of related vectors are reduced to a single vector (or smaller set of vectors). As discussed in Chapter 2, an important property of a pooling operator for non-Euclidean data is invariance to different orderings of the input vectors. In vertex level learning tasks such as link prediction and vertex classification, GCNs achieve invariance by pooling neighbours' feature vectors with symmetric operators such as feature-weighted mean [17], max [27], and self-attention weighted means [30].

Standard approaches to use these permutation equivariant models for graph level classification are to use symmetric global operators such as as mean [36] or max pooling [65]. As discussed in Section 2.3.4, while such pooling operators do provide graph level permutation invariance, they fail to take into account the relational structure of the graph [108], thus treating all vertices with equal significance.

To enable graph classification with MPNNs whereby the structure of the graph is used to inform the pooling process we propose PiNet, a differentiable pooling mechanism with which any permutation equivariant MPNN may be extended for graph level tasks. Inspired by the attention mechanisms of RNNs [135] and GAT [30], PiNet consists of an attention-based aggregation method which weights the importance of each vertex in the final graph representation.

3.2 CONTEXT

We have reviewed the relevant related works in Chapter 2. At the time of starting this research, there was little attention to utilising MPNNs for graph level tasks, with a typical approach being to take a global mean of vertex embeddings to form a graph level representation [18].

For fair comparison of our method, we compare against the best available methods at the time, *i.e.* [21, 110, 136]. Based on advances since this work was done, we propose a possible extension to PiNet in Section 6.3.1.

3.3 METHOD

3.3.1 *Learning Vertex Importance*

PiNet is a generalised end-to-end deep neural network architecture that utilizes the vertex-level permutation invariance of graph convolutions in order to learn graph representations that are also invariant to permutation, while taking into account the structure of each graph when pooling to a single representation. We provide an overview of the architecture in Figure 3.1.

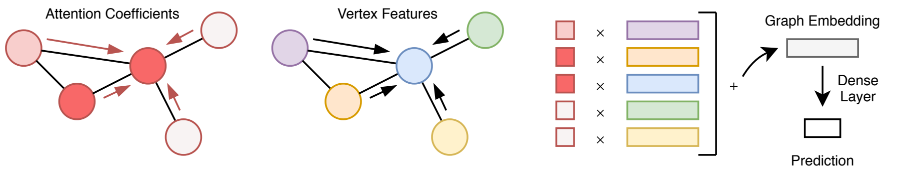


Figure 3.1: Overview of PiNet: One message passing network learns vertex features, the other learns attention coefficients. The final graph representation is a sum of the learned vertex features weighted by the attention coefficients. For multiple attention dimensions per vertex, the graph embedding becomes a matrix where the rows are concatenated to form a single vector.

Let $G = (A, X)$ be a graph from a set $\mathcal{G}$ with adjacency matrix $A \in \mathbb{R}^{N \times N}$ and vertex features matrix $X \in \mathbb{R}^{N \times F}$, and $\psi_X : (\mathbb{R}^{N \times N}, \mathbb{R}^{N \times F}) \rightarrow \mathbb{R}^{N \times F'}$ be any message passing convolution network (*i.e.* the GCN [17]). PiNet may then be defined by the output for a single graph,

$$z(G) = \sigma_S\left[\alpha \cdot \psi_X\,(A, X)\,W_D\right] \in \mathbb{R}^C, \tag{3.1}$$

where σ_S is the softmax activation function, $\cdot$ is a matrix product, $W_D \in \mathbb{R}^{F' \times C}$ is a weights matrix for a fully connected dense layer (bias terms are omitted for brevity), C is the number of target classes, and α is the learned vertex attention coefficients representing the importance of each vertex in the final graph representation such that

$$\alpha = \sigma_S \left(\psi_A(A, X)^\top\right) \in \mathbb{R}^{1\times N} \tag{3.2}$$

where ψ_A is a separate message passing network for learning the attention coefficients. Note, ψ_A and ψ_X do not share weights and may contain any arbitrary numbers of layers. The softmax in Equation 3.4 constrains the attention coefficients such that $\sum_i \alpha_i = 1$ and prevents them from all falling to 0, while the softmax in Equation 3.3 may be replaced with sigmoid for mult-label classification or regression tasks.

3.3.2 *Multi-Head Attention*

The method proposed above can be considered a single-headed attention mechanism, whereby a single vertex coefficient is learned per vertex. This may be easily extended to multi-headed attention by adapting ψ_A to learn multiple coefficients per vertex in parallel, and concatenating the result as per [72] and [30]. Using multi-head attention allows the model to jointly consider information from different representation subspaces at different positions [72], *i.e.* in this case, at different vertices.

To realise the multi-headed attention, we modify ψ_A such that $\psi_A : (\mathbb{R}^{N\times N}, \mathbb{R}^{N\times F}) \to \mathbb{R}^{N\times H}$, where H is the number of attention heads, and replace Equation 3.1 such that

$$z(G) = \sigma_S\Big[g\Big(\alpha \cdot \psi_X\,(A, X)\Big)W_D\Big] \in \mathbb{R}^C, \tag{3.3}$$

with Equation 3.2 becoming

$$\alpha = \sigma_S \left(\psi_A(A, X)^\top\right) \in \mathbb{R}^{H\times N} \tag{3.4}$$

where $g : \mathbb{R}^{H \times F'} \rightarrow \mathbb{R}^{HF'}$ is a flattening function that concatenates the rows of a matrix to form a vector, and $W_D \in \mathbb{R}^{HF' \times C}$ is a fully connected dense layer (again bias terms omitted for brevity).

3.3.3 *Extended Message Passing Operator*

We extend the message passing matrix of [17] in which we add two additional trainable parameters, thus vector state is propagated by the matrix

$$\tilde{A} = (pI + (1-p)D)^{-\frac{1}{2}}(A + qI)(pI + (1-p)D)^{-\frac{1}{2}}, \tag{3.5}$$

where I is the identity matrix, D is the diagonal degree matrix, and A is the graph adjacency matrix. p allows the model to optimise the extent to which to apply symmetric normalisation of the adjacency matrix, and q (as originally supposed for further work in [17]) allows the model to optimise the trade-off between keeping a vertex's own state and aggregating the states of its neighbours. Note that p and q are learned indirectly through optimising p' and q' with sigmoid to give $0 \leq p, q \leq 1$.

3.3.4 *Geometric Features of Molecules*

The architecture presented in this work is general to many domains, however, we are motivated by the use case of predicting molecular properties. Thus in the interest of applying this method in practical settings, we now discuss how the geometric features of molecules may be incorporated.

Since a molecule has no canonical orientation, and many bonds (typically single bonds) are free to rotate, it is not possible to provide canonical absolute positions for atoms that generalize across different molecules, even when considering a molecule's lowest energy state. While relative positions between atoms would provide invariance to rotations or translations of a complete molecule, they would not provide invariance to rotation of bonds within a molecule. However, distances between atoms would satisfy all required invariances.

It is also possible to include labels for the types of bonds. Thus each edge should have a feature vector containing the distance between the atoms, as well as a one-hot representation of the bond type.

Since PiNet is general to all message passing operators, an encoder which incorporates edge features could be used. For example, GIN [91] (since it is a maximally powerful MPNN) could be used with edge features embedded and added to node embeddings as proposed by Hu *et al.* [137].

While encoding distances between atoms provides the required invariances to translations and rotations, two pieces of information are lost: the angles between atoms, and the chirality. While chirality can be labelled as a vertex feature a priori, encoding the angles is more complex. We refer the reader to [115] for an example of how this may be achieved.

3.4 EXPERIMENTS & RESULTS

To evaluate the effectiveness of our proposed method, we investigate four aspects. First, using a synthetic dataset described below, we perform an isomorphism test in order to measure how well each method is able to distinguish isomorphic graphs. Second, we measure the effectiveness of our proposed extension to the GCN [17] as a vertex aggregation method for graph level learning, considering both a manual parameter search and learning values for p and q. Third, for the MU-

TAG dataset we investigate the effect of the number of attention heads on overall classification performance as well as the interpretability of the learned attention coefficients α. Finally, we compare our proposed method against existing state of the art methods on a set of chemo-informatic benchmark datasets.

All hyper-parameters are detailed in Section 3.4.2.

3.4.1 *Datasets*

For the isomorphism test (3.4.3) we use a generated dataset available from our repository. To generate the data we sample 5 unique undirected Erdõs-Rényi graphs [138] with edge creation probability 0.15, and ensure equal vertex degree distributions (we discard and regenerate the graphs until the degree distributions match) - this ensures a high level of challenge and prevents trivial classification. Each graph contains 50 vertices, and each vertex is assigned one of two classes uniform randomly. The 5 unique graphs are then copied 99 times each and the vertex ids are permuted randomly on all of the graphs since we wish to test the ability to recognise isomorphic graphs even with different vertex orderings. We show a sample of the generated graphs in Figure 3.2.

	MUTAG	NCI-1	NCI-109	PTC	PROTEINS
$\lvert\mathcal{G}\rvert$	188	4110	4127	344	1113
Max. N	28	111	111	109	620
Mean N	18	29.8	29.6	25.56	39.06
d	7	37	38	18	3
% of +ve	66.49	50.05	50.38	39.51	59.57

Table 3.1: Benchmark binary classification molecule datasets. $\lvert\mathcal{G}\rvert$ is the number of graphs, N the number of vertices, and d the dimensions of the vertex features.

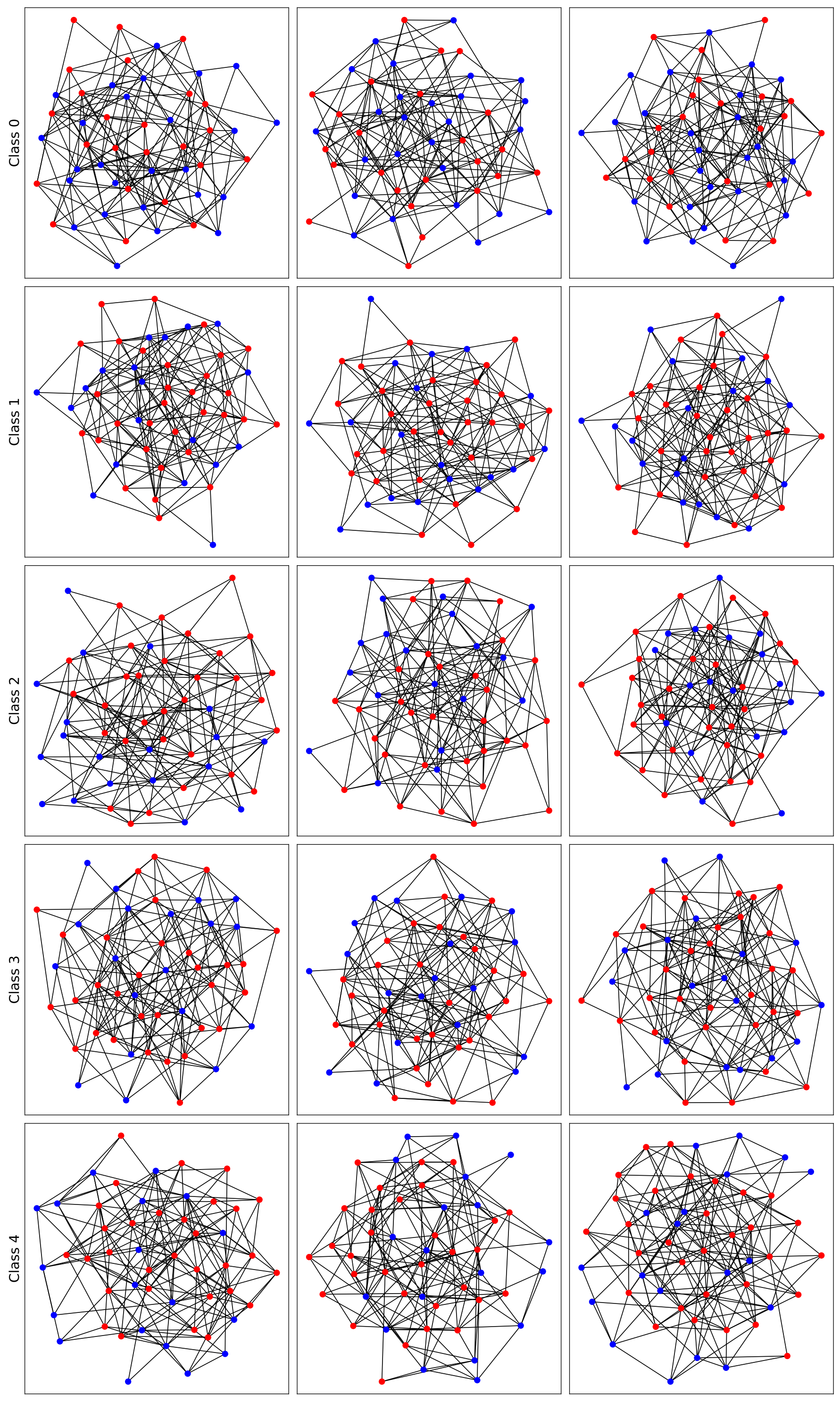


Figure 3.2: Example graphs from our generated Isomorphism dataset. Each row contains three isomorphic graphs of a different class. Colours indicate vertex labels.

We note that the graphs generated in our isomorphism test dataset are all distinguishable by the 1-WL Isomorphism Test. Since we are comparing MPNNs, the use of graphs not distinguishable by the 1-WL test would mean that all methods tested would fail to outperform a random classifier. Thus the graphs are theoretically distinguishable by an MPNN, while remaining challenging.

All other experiments are performed using a standard set of chemo-informatic benchmark datasets[1] detailed in Table 3.1.

3.4.2 *Hyper-Parameters*

In all experiments we use categorical cross-entropy for loss, and fix learning rate to 10^{-3}. For the imbalanced datasets MUTAG, PTC and PROTEINS, we weight the loss inversely proportional to the class sizes.

- PiNet (GCN): hidden sizes $\{32, 64\}$ for each layer in each head (two layers).
- GCN + Dense & GCN + Mean: hidden sizes $\{32, 64\}$ for each layer (two layers).
- DiffPool: assign-ratio in $\{0.1, 0.2, 0.3\}$, hidden layer sizes in $\{30, 40, 50\}$ (for two layers)
- DGCNN: hidden sizes in $\{64, 96, 128\}$ and 3 sort pooling values selected according to the size of each dataset.
- Patchy-SAN: labelling procedures: NAUTY [139] and Betweenness Centrality [140].

1 Available at `https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets`.

3.4.3 *Isomorphism Test*

We start by testing the ability of our proposed method to distinguish isomorphic graphs. While PiNet is a general framework that may utilize any permutation equivariant graph convolution operator, here we opt for the simplest case and use the GCN [17] (see Section 2.3.3). In detail we set

$$\psi_X = \sigma_R(\tilde{A} \cdot \sigma_R(\tilde{A}\; XW_X^{(0)})\; W_X^{(1)}) \tag{3.6}$$

$$\psi_A = \sigma_R(\tilde{A} \cdot \sigma_R(\tilde{A}\; XW_A^{(0)})\; W_A^{(1)}) \tag{3.7}$$

where $\tilde{A} = D^{-\frac{1}{2}}\hat{A}D^{-\frac{1}{2}}$, D is the diagonal degree matrix of $\hat{A}$, $\hat{A} = A + I$, I is the identity matrix, and σ_R is the Rectified Linear Unit (ReLU) activation function. Then ψ_X and ψ_A may be substituted into equations 3.3 and 3.4 respectively. We refer to this as PiNet (GCN). To evaluate the performance of our proposed architecture directly, we compare against a GCN [17] with a dense layer applied to the concatenated vertex vectors and a GCN [17] with a dense layer on the mean of its vertex vectors.

We also compare with three state of the art graph classifiers: DiffPool [110], DGCNN [136], and Patchy-SAN [21] (see Section 2.3 for details). We vary the number of training examples using stratified sampling and report the mean validation accuracy of 10 trials.

Figure 3.3 shows the mean classification accuracy of each model as we vary the number of training examples per class. Our result demonstrates that PiNet is able to distinguish the isomorphic graphs even with as few as 2 examples per class while many of the other methods fail. With the exception of DiffPool, all models improve as the number of examples of each class increases, with the surprising result that the GCN [17] with only a mean pooling operator shows the greatest rate of

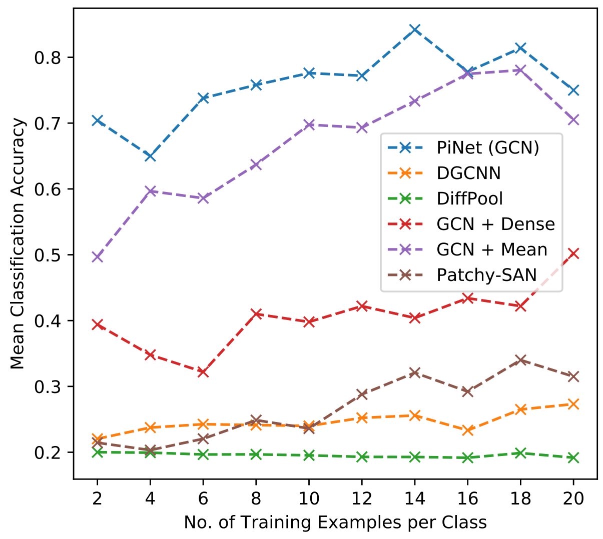


Figure 3.3: Mean classification accuracy over a range of training set sizes on the isomorphism dataset.

increase. This is likely due to the simplicity of the model. Interestingly, this data presents a worst-case scenario for DiffPool which is unable to distinguish the different graph classes at all.

While PiNet outperforms all other tested methods at all numbers of training examples per class, the relative improvement over the other methods is most apparent as the number of training examples is smallest, indicating that PiNet is more sample efficient than the other methods.

We note that PiNet is not theoretically more powerful than the other methods tested with respect to the 1-WL isomorphism test. Thus, a likely reason for the improvement of PiNet (GCN) over the other methods tested is in the optimisation of its loss function. We hypothesise that PiNet's aggregation method is able to amplify the necessary signals in order to better avoid local minima in this classification task, while the other methods tested become trapped more easily. This would explain why increasing the number of examples per class generally improves the performance of each of the methods, while PiNet performs well with few examples.

3.4.4 *Message Passing Mechanism*

To evaluate our proposed extension to the message passing mechanism of [17] in the context of graph level learning (Section 3.3.3), we compare the classification accuracy of the extreme cases of p and q ($A, A+I$, $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, and $D^{-\frac{1}{2}}(A+I)D^{-\frac{1}{2}}$) against the learned p and q for each layer in each head on the set of benchmark datasets detailed above. We use the extreme values of p and q in order to consider all possible discrete components of Equation 3.5. Following the methodology of [31] and [110] we perform 10-fold cross validation, reporting the mean validation accuracy for the single best epoch across the folds.

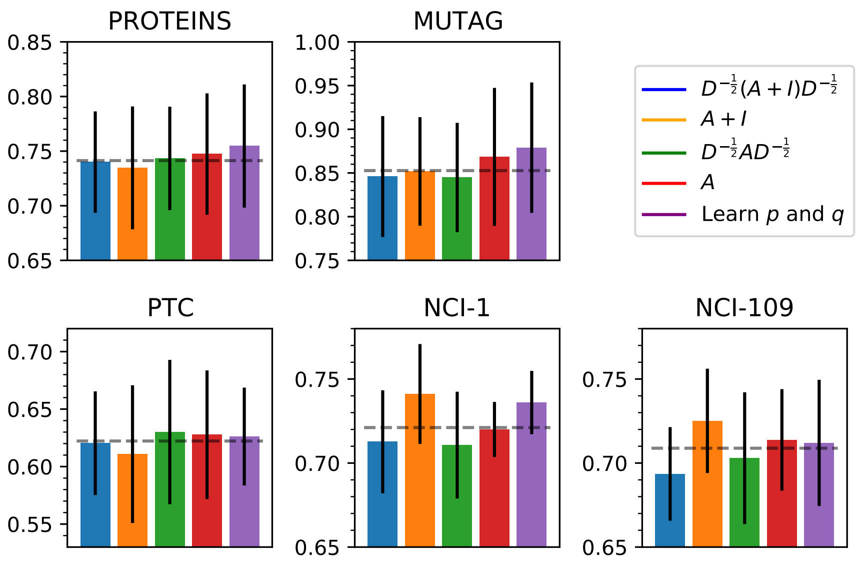


Figure 3.4: Mean classification accuracy for each message passing matrix within PiNet. Dashed lines indicate mean accuracy of manual search.

Figure 3.4 shows that while the optimal parameters p and q are not always found, the result of learning p and q offers better performance than the average of a manual search over the extreme values in all cases thus suggesting it is a suitable technique to reduce hyper-parameter searching.

3.4.5 *Attention*

Next we consider the effect of the number of attention heads on overall classification performance. We focus the experiments of this section purely on the MUTAG dataset due to its smaller size, which enables us to visualize and interpret our results in the original problem domain of predicting the chemical property of mutagenicity.

We use PiNet (GCN) with p and q as trainable parameters, and report the mean validation accuracy for each number of attention heads in $\{1, 2, 4, 8, 16, 32, 64\}$. As above we perform 10-fold cross validation and select the single best epoch across all trials.

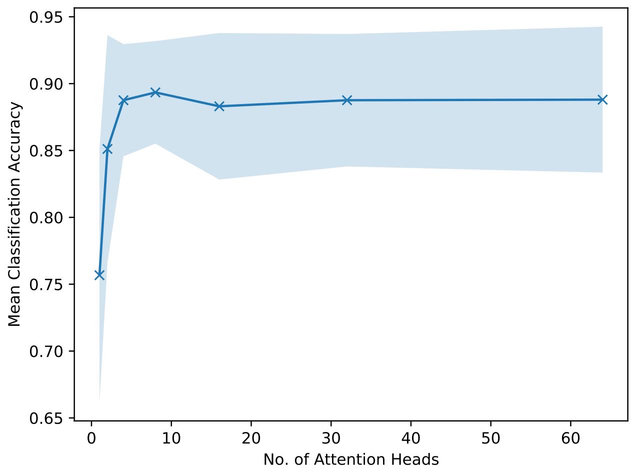


Figure 3.5: Mean classification accuracy for a range of number of attention heads for MUTAG dataset.

As demonstrated in Figure 3.5, using multiple attention heads over a single head improves the classification accuracy significantly, with 8 heads showing an improvement of approximately 18%.

To understand this improvement and investigate the interpretability of the learned attention coefficients, we select 4 visually similar aromatic amines (two positive and two negative) from the dataset, and plot the graphs corresponding to each compound. Since PiNet (GCN) does not consider edge information, we plot all bonds as a single undirected edge regardless of the bond type/valency. Vertex (atom) types are shown with text labels for Carbon (C), Nitrogen (N), and Oxygen (O).

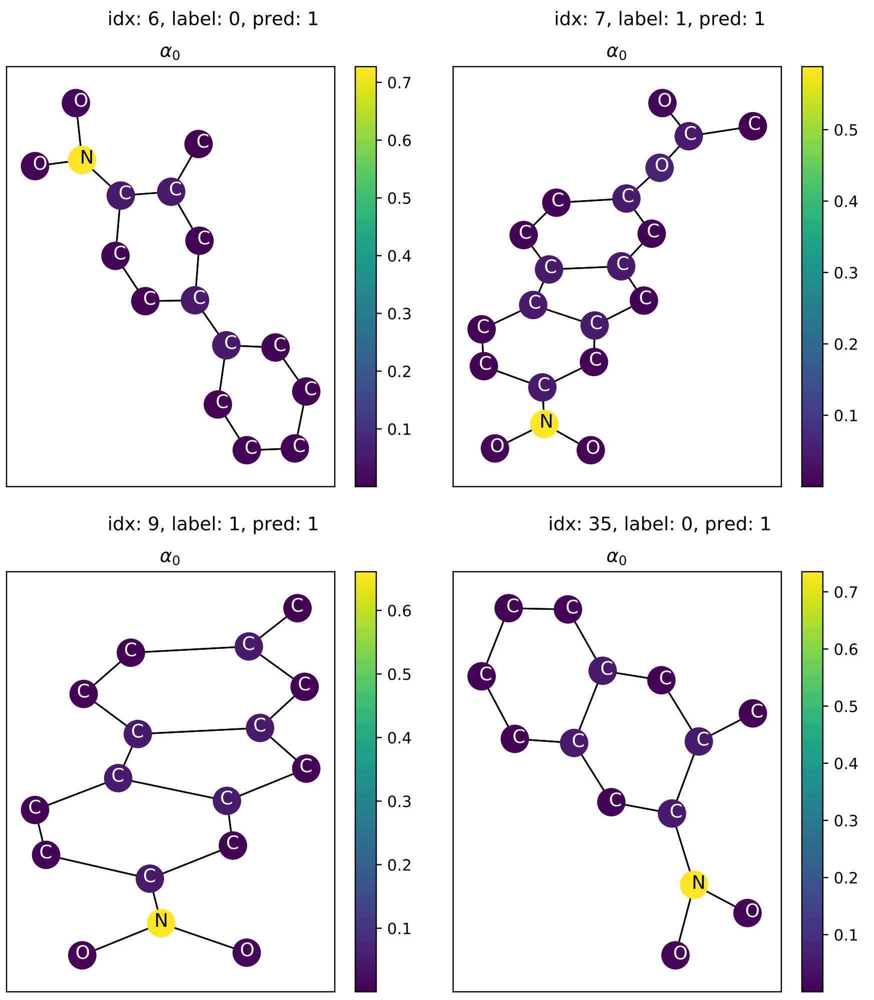


Figure 3.6: Attention coefficients for MUTAG molecules 6, 7, 9 and 35 trained with a single attention head.

As shown in Figure 3.6, in each case approximately 60-70% of the attention is placed on the nitrogen atom, with the rest of the attention shared uniformly across all remaining vertices. In the context of mutagenicity, this is an interesting result since nitrogen groups (a nitrogen connected to a ring and two other atoms) are reactive functional groups [141] and are known to attack DNA [142].

In the case of compounds 7 and 9, it is the presence of the nitrogen group that causes these compounds to be mutagenic, however there is also another necessary factor that the model has missed causing it to misclassify compounds 6 and 9 as false positives, *i.e. steric hindrance*, which is the congestion caused by the physical presence of the surrounding atoms, which may slow down or prevent reactions at the atom in question [143]. As shown in Figure 3.7, while the nitrogen group is in itself reactive, to bind with DNA requires sufficient physical space around it to allow to get close enough into the DNA structure [144].

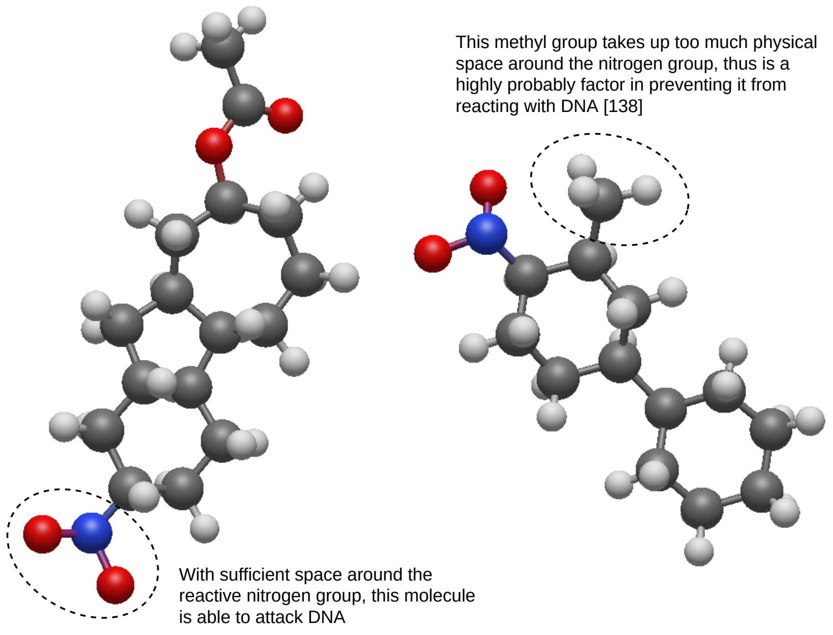

This methyl group takes up too much physical space around the nitrogen group, thus is a highly probably factor in preventing it from reacting with DNA [138]
With sufficient space around the reactive nitrogen group, this molecule is able to attack DNA

Figure 3.7: 3D views of molecules 7 (left) and 6 (right) including hydrogen atoms demonstrating steric hindrance [145]. Diagram created with [146].

To interpret the attention coefficients learned with 8 heads, we first perform Principal Component Analysis (PCA) on the attention coefficients across all graphs (see Figure 3.8), and select the three most diverse cases (attention heads 0, 5, and 7) alongside the most centred case (head 1).

As shown in figures 3.9–3.12, the model now correctly classifies compounds 6 and 9. In each case, heads 0, 1 and 7 rank the importance of the methyl groups (CH_3 groups bonded to ring) highly, while heads

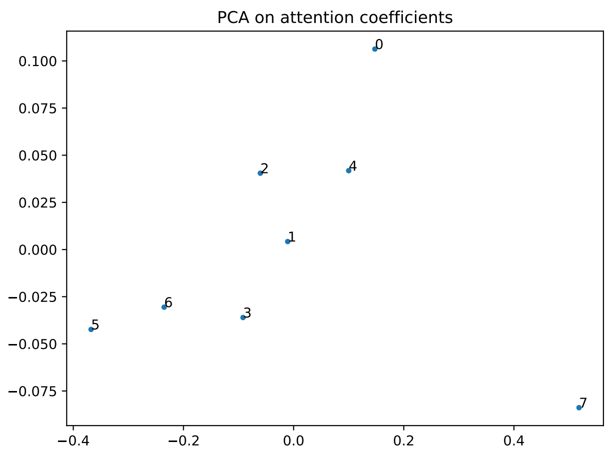


Figure 3.8: PCA on the learned vertex attention coefficients across all graphs.

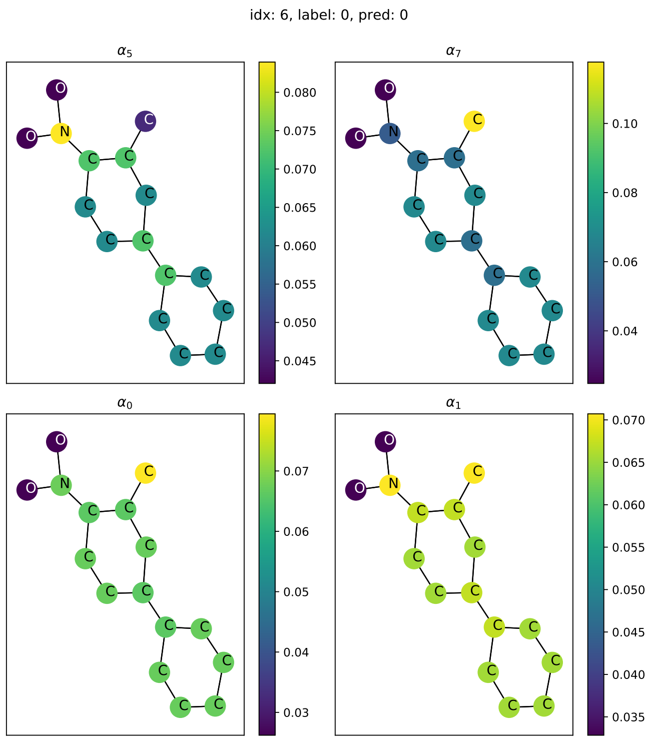


Figure 3.9: Attention coefficients for molecule 6 at most diverse attention heads (trained with 8 heads).

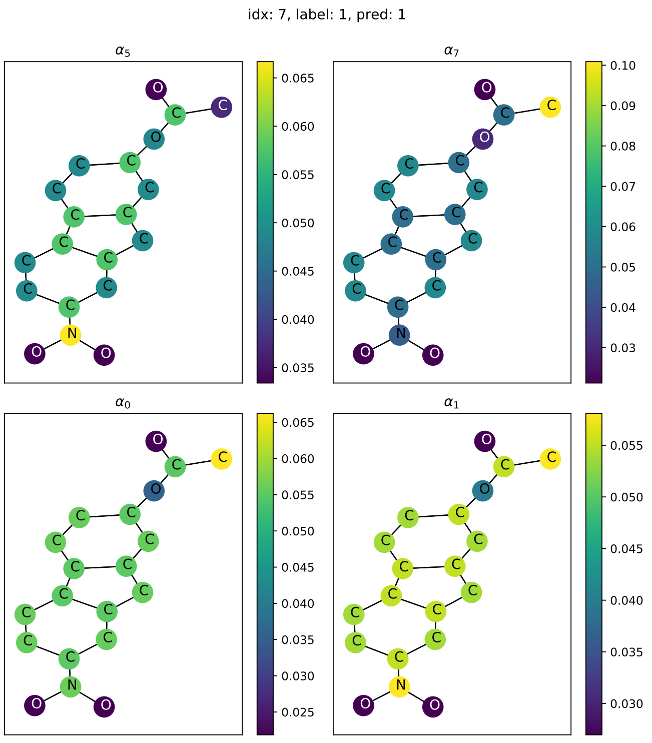


Figure 3.10: Attention coefficients for molecule 7 at most diverse attention heads (trained with 8 heads).

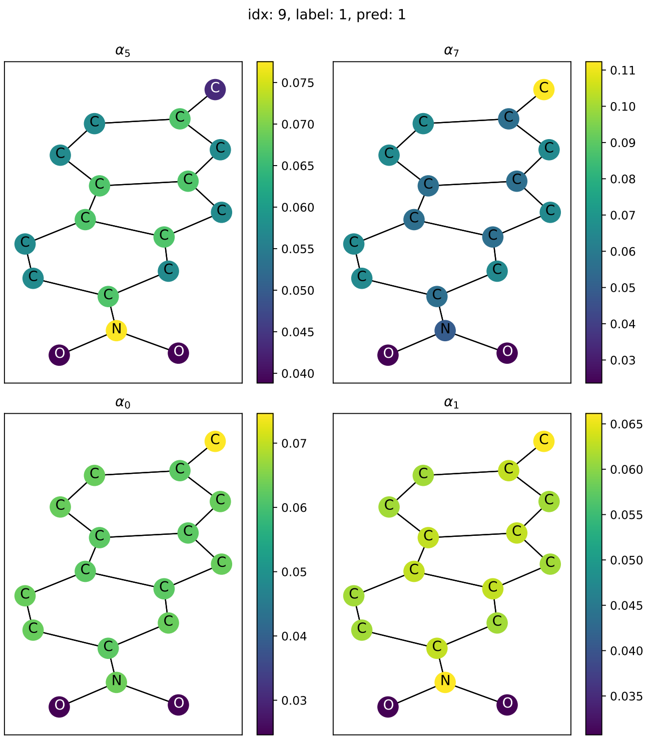


Figure 3.11: Attention coefficients for molecule 9 at most diverse attention heads (trained with 8 heads).

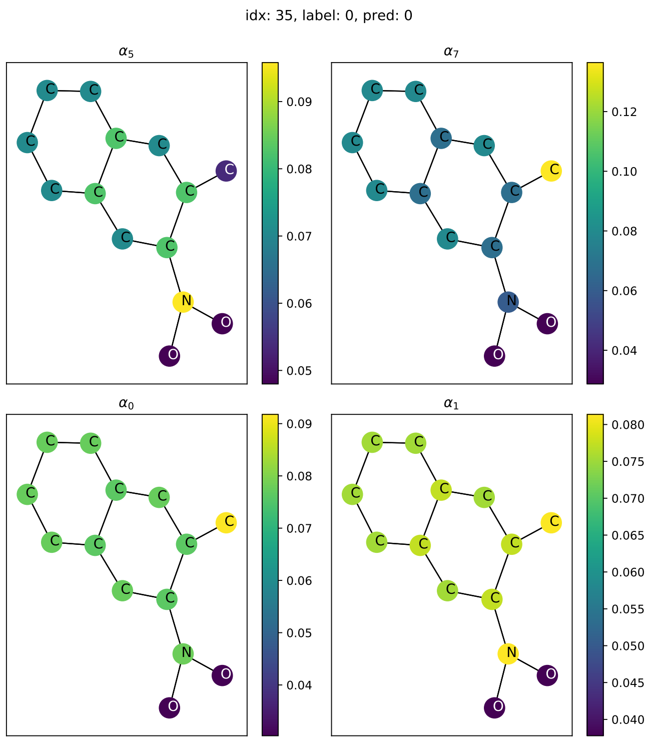


Figure 3.12: Attention coefficients for molecule 35 at most diverse attention heads (trained with 8 heads).

1 and 5 rank the nitrogen highly. This result suggests it is an awareness of stereo hindrance that enables this model to correctly classify compounds 6 and 9 as negatives, while the single head version failed.

While this result clearly indicates the importance of the methyl groups in determining the mutagenicity of these compounds, it is their relative position to the nitrogen group that determines the stereochemistry and thus is of significance. Since our formulation of PiNet in this experiment is formed of two 2-layer GCNs, the information available in any vertex's representation before pooling can only contain information of other vertices at most 2-hops away. Thus, the information in the nitrogen vertex and methyl group vertex representations alone are not enough to determine their relative distances. Therefore, the vertices in the path between them will also need to be considered. We hypothesize that this is why the attention heads 0 and 5, for example, also allocate a fair amount of attention on the joining (ortho and meta) carbon atoms.

To understand why the trend of correlation between number of heads and classification accuracy does not continue beyond about 8 heads, we plot the mean explained variance (10 cross-fold trials) of the two principle components of the attention heads across all graphs.

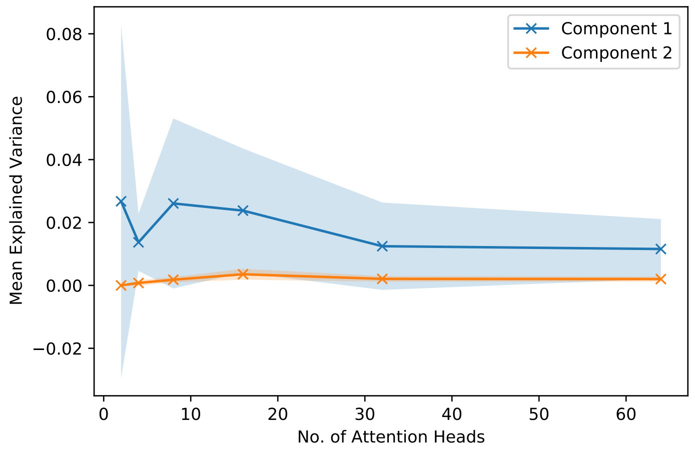


Figure 3.13: Mean explained variance for top two principle components of attention coefficients across all MUTAG graphs.

As shown in Figure 3.13, in all cases there is only one dominant source of variation and as the number of heads increases beyond 8 the diversity in these learned attention coefficients actually begins to reduce, with new heads learning more similar coefficients. This result is of course limited to this dataset of small compounds, thus it is possible that with a larger dataset of larger compounds and a more difficult task, increasing the number of heads beyond 8 may bring further gains in classification performance.

3.4.6 *Benchmark*

Finally, to compare our proposed method against the existing state of the art on realistic data we benchmark the performance of PiNet with the original [17] and extended GCNs (3.4.4), on the benchmark datasets described above. For baseline we again compare against the GCN [17] with a dense layer and mean pooling, and for state of the art we compare against Patchy-SAN [21], DGCNN [136] and DiffPool [110].

As above, we perform 10-fold cross validation and adopt the methodology of [31] and [110] reporting the mean validation accuracy for the single best epoch.

	MUTAG	NCI-1	NCI-109	PROTEINS	PTC
GCN + Dense	0.86 ± 0.06	0.73 ± 0.03	0.72 ± 0.02	0.71 ± 0.04	0.63 ± 0.07
GCN + Mean	0.84 ± 0.07	0.68 ± 0.03	0.67 ± 0.03	0.74 ± 0.02	0.63 ± 0.04
Patchy-SAN	0.85 ± 0.06	0.58 ± 0.02	0.58 ± 0.03	0.70 ± 0.02	0.58 ± 0.02
DGCNN	0.86 ± 0.07	0.73 ± 0.03	0.72 ± 0.02	0.73 ± 0.05	0.61 ± 0.06
DiffPool	0.91 ± 0.08	0.73 ± 0.02	0.72 ± 0.03	0.80 ± 0.05	0.64 ± 0.07
PiNet (GCN)	0.85 ± 0.07	0.71 ± 0.03	0.69 ± 0.03	0.74 ± 0.05	0.62 ± 0.05
PiNet (GCN*)	0.87 ± 0.08	0.74 ± 0.03	0.73 ± 0.03	0.75 ± 0.06	0.63 ± 0.06
PiNet (GCN**)	0.88 ± 0.07	0.74 ± 0.02	0.71 ± 0.04	0.75 ± 0.06	0.63 ± 0.04

Table 3.2: Benchmark results. * indicates GCN [17] with extended message passing with manual p and q, and ** with learned p and q.

As shown in Table 3.2, we observe competitive performance with (within one standard deviation or better than) all of the state of the art methods for all datasets.

3.5 SUMMARY

We have introduced PiNet, a generalised attention-based pooling mechanism for utilizing vertex-level convolution operators for graph level representations. We have demonstrated its ability to distinguish 1-WL distinguishable isomorphic graph classes with greater accuracy than existing state of the art methods on our generated isomorphism test dataset, and demonstrated results competitive with current state of the art methods on standard benchmark datasets. Moreover, the interpretability of the learned attention coefficients demonstrates a clear advantage to our method over the alternatives.

For further work we propose further study of PiNet with different convolution operators, as well as the use of skip connections to add greater flexibility to the learned vertex representations prior to graph level pooling, and as detailed in Section 6.3.1 we propose an adaptation to the attention process to enable hierarchical pooling.

Having investigated our proposed method for graph level pooling, we have shown PiNet to be effective in the multiple domain graph level task of graph classification as well as the multiple domain vertex level task of generating meaningful visualizations of vertex significance in interpreting the model's predictions. Next, we continue the investigation considering multiple domains for graph and vertex level tasks, but in the entirely different context of CAD.

4

UVSTYLE-NET: UNSUPERVISED FEW-SHOT LEARNING OF 3D STYLE SIMILARITY MEASURE FOR B-REPS

The following chapter presents work published in [1]. The reviewers gave recognition of "extensive experiments" with "solid experimental results and excellent method analysis (both quantitatively and qualitatively)". They also recognise B-Rep style as an "interesting problem" and that our "method presents good retrieval results compared to other representations ... as well as baseline networks". Finally, they note that a clear discussion of limitations "makes it a reliable work to follow with".

The key criticism of this work identified in the review process was the lack of an intuitive explanation of the Gram matrices in the context of B-Reps, which we addressed with Section 4.3.1. Prior to submission at this venue, a previous review process rejected parts of this work on the grounds of lacking quantitative comparison against baselines on real-world data, which we addressed for this submission with Table 4.6. We thank all reviewers for their contribution to this work.

4.1 INTRODUCTION

Continuing in the multiple domain case for graph level tasks, here we consider the use case of style similarity for B-Reps, also with a focus on the vertex level task of producing helpful gradient visualizations for feedback to designers.

B-Reps are the de facto standard for industrial design, and the representation most widely used in the consumer product and automotive industries where style is of great importance [147]. B-Reps offer unparalleled editability in a compact, memory efficient representation, they are not discretized/sampled (as per mesh/point cloud) offering precise boundaries with continuous smooth surfaces/edge curves. For example, to represent a sphere, a B-Rep would only need to store a radius and centre, whereas a mesh or point cloud would need to store the coordinates for many discrete sampled points from the sphere's surface.

As a widely used, compact non-Euclidean representation for solid models rich with relational structure, B-Reps are a highly suitable, yet relatively unexplored data structure for the application of message passing algorithms and thus a pertinent choice for investigation in this thesis.

There are many use cases for a B-Rep style similarity measure, *i.e.* finding architectural parts that are inkeeping with the style of a building, or selecting parts for a car that fit with the manufacturer's existing range. Moreover, the gradient of a style similarity measure can be used to generate helpful visualizations or modify the input 3D shape a la Gatys *et al.* [148].

Geometric style is inherently subjective and may have a different meaning in different object class domains, *i.e.* the boundary between style and content is unclear. For example, in the context of chair designs, number of legs could be considered either style or content depending on the particular use case. Thus, an effective geometric style measure must cater for these different interpretations of the end user.

While existing methods use hand-crafted features [149, 150] or crowdsourcing [151–154] to pre-define and measure geometric style, we propose a user-defined few-shot style metric learning method that lever-

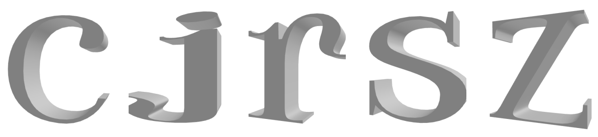


Figure 4.1: Lower case examples from font 'Viaoda Libre'. While 'j' and 'r' share some stylistic features, they are not obviously similar to 'c', 's' or 'z', *i.e.* font classes provide a ground truth for style compatibility (as perceived by their designers) yet only a *weak* label for style itself.

ages the range of style signals available in the activations of a pre-trained 3D object encoder through second order statistics (Gram matrices). The relative importance of each layer's Gram matrix is then learnt through selection of just a few examples of what style means to an end user (see Figure 4.3).

Despite the abundant use of B-Reps in industrial settings, there is a fundamental lack of publicly available B-Rep data for training machine learning models — in particular, at the time of writing, there are no existing B-Rep datasets that include a reliable ground truth for style. To overcome this challenge, we provide an adaptation to SolidMNIST [127], which improves the style consistency within font classes for the evaluation test set. The font classes, however, still provide only a weak label for style (see Figure 4.1), and as such we propose an unsupervised method and use the font labels purely for quantitative evaluation to justify design choices of our method. For comparison against existing state of the art on real-world data we also provide evaluation with the unlabelled ABC dataset [89] and a manually labelled subset of it.

In summary, we introduce a geometric style similarity measure for 3D solids that may be used in completely unlabelled settings for arbitrary object classes, with user subjectivity handled by few-shot learning given only a very small number of examples. While our method is adaptable for all 3D input types, we demonstrate the benefits of our approach with B-Reps (over meshes and point clouds) both quantitatively and qualitatively.

4.2 background & related work

Since this chapter focuses on the specific application of style similarity for B-Reps, we provide additional background and related work relevant to this domain here. We start with a brief introduction to B-Reps (4.2.1), we then review 3D geometric feature learning methods (4.2.2), traditional 3D geometric style similarity methods (4.2.3), conventional 2D style transfer methods (4.2.4), and finally 3D style transfer methods (4.2.5).

4.2.1 *Introduction to B-Reps*

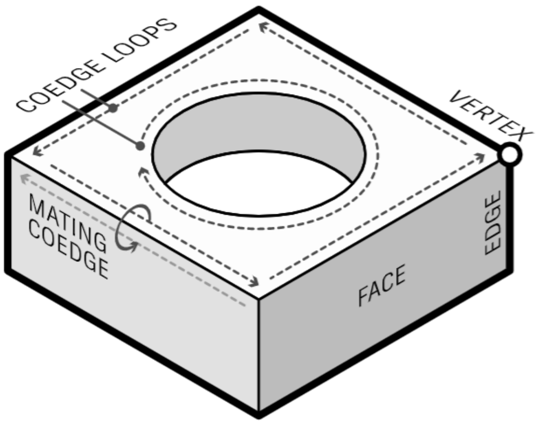


Figure 4.2: The B-Rep data structure: Faces are defined by parametric surfaces, bounded by loops of trimming curves. Each trimming curve is owned by a topological entity called a coedge, which stores adjacency relationships between faces. Figure from [155].

B-Reps are loosely analogous to 2D Scalable Vector Graphics (SVGs) for 3D. The precise implementation details vary between different CAD software packages, below we describe the general principles relevant to all B-Reps.

As shown in Figure 4.2, B-Reps are collections of parametric curves and surfaces along with topological information which describes the adjacency relationships between them [155]. They are typically used to describe closed volumes (solids), but can also represent 2D manifolds (sheets) and curve networks (wire bodies). Each face of a B-rep body is defined by a parametric surface which is divided into "visible" and "hidden" regions by a series of trimming loops. The loops comprise an ordered cycle of coedges, which store pointers to "mating" coedges on adjacent faces. The loop ordering and coedge-coedge adjacency information provides a full description of the bodies topology, while the parametric curves and surfaces provide the geometric information [156].

B-Reps differ from point clouds and meshes since the are precise representations with continuous smooth surfaces and edge curves — they are not sampled/discrete. Consequently, complex solids may be expressed with low memory requirements without loss of detail [40].

The relational structure in the underlying graphs which define the topology of connected geometric parts makes B-Reps a prime candidate for message passing techniques, and thus is a key motivation for the work in this chapter.

For further information on B-Reps see [40, 155, 156].

4.2.2 *Geometric Feature Learning*

Geometric feature learning has seen many successes for both Euclidean representations, *i.e.* multi-view [88], projections [157], volumetric [158], and non-Euclidean representations, *i.e.* point clouds [24, 159, 160] and mesh [161, 162]. For a detailed review of geometric feature learning we refer the reader to [39, 163, 164]. Despite the prevalence of B-Reps in industrial and creative design applications, however, geometric feature learning for parametric representations remains largely unexplored.

In addition to their wide use, there are many advantages to working with B-Reps as 3D geometric representations. Not only do B-Reps typically require less memory than point clouds or meshes (depending on the sampling resolution/detail of the model), but they also provide richer information about a solid, including the precise boundaries of every surface and the topology of these surfaces. It is this rich relational information that makes B-Reps particularly suitable for message passing algorithms.

The benefits of B-Reps over discretized representations are demonstrated in Jayaraman *et al.* [127], where each face is sampled uniformly in its parameter domain to form a regular grid then passed through a 2D CNN. The CNN face representations are then fed to a GNN which uses the face adjacency matrix of the original B-Rep.

4.2.3 *Geometric Style Similarity*

Existing geometric style similarity learning methods are typically trained in a supervised setting, requiring a set of hand-labelled triplets (A, B, C) in which the pair A and B are believed to be closer in style than A and C [149–154]. To account for style subjectivity, examples are labelled through crowd-sourcing methods and thus result in a generally accepted definition for style.

For example, Liu *et al.* [150] use hand-crafted features (*i.e.* curvature histograms) with a supervised triplet loss to learn furniture compatibility, while Lun *et al.* [149] apply a similar method by first segmenting input models into sub-parts to compute geometric features for independently.

Geometric style feature learning has been demonstrated by Lim *et al.* [151] and Pan *et al.* [152] whereby 3D meshes are first projected into multiple 2D views which are then processed with a traditional triplet image CNN. Polania *et al.* [153] adopt a similar approach, where the learned style representations are then passed to a GNN for compatibility prediction.

Rendering 3D solids into 2D (even with multiple views) is problematic since stylistic features can be lost or occluded and selecting the best views without making assumptions on the orientations of the data is non-trivial. Pan *et al.* [154] overcome this using curvature-guided sampling directly from the solids to generate element-level style features which are then aggregated to global style representations using a triplet network.

The reliance of these methods on crowd-sourced, hand-labelled style triplets creates two problems: Firstly, there is limited labelled data available in 3D style domain, and no labelled B-Rep data. Secondly, and more importantly, the definition of style (an inherently subjective concept) is pre-defined according to a consensus, hence may not be compatible with an end-user's particular taste or application.

4.2.4 *Style Transfer*

Contrary to the geometric style learning methods above, the style transfer literature has largely adopted the use of first and second order activation statistics from deep pre-trained image classifiers in order to represent and quantify style. Gatys *et al.* [148] showed that feature co-occurrence in the different layers of a CNN effectively captures elements of style at different scales of abstraction. In the finest layers where features are most local, the style representation given by the Gram matrix captures color and texture information, yet deeper into the network, the Gram matrices capture higher level structure and patterns eventually crossing into semantic content.

Following from this, Huang *et al.* [165] and Babaeizadeh *et al.* [166] demonstrate that first order activation statistics (channel-wise mean and variance) are also able to capture elements of style through the use of Adaptive Instance Normalization (AdaIN). Karras *et al.* [167] illustrate the relationship between layer depth and the style/content tradeoff by swapping the inputs to a generator at varied depth. Swapping at lower layers renders image interpolations of low level texture/colour information, and swapping at deeper layers interpolates semantic content.

Many further works utilize and extend the use of first order statistics of network activations to improve style transfer results, *e.g.* Generative Adversarial Network (GAN) based methods [168–170]; however, these methods rely on a generator to align the activations to these statistics while generating an output image, with the main focus on the quality of the output images rather than the interpretability of the statistics in defining an explicit style distance metric for arbitrary inputs. To explicitly disentangle style and content for arbitrary inputs Park *et al.* [171] propose an auto-encoder architecture that adopts the technique of swapping inputs at various layers and a GAN based encoder and discriminator that is able to effectively separate structure and texture.

Azadi *et al.* [172] propose a few-shot learning approach for font style transfer in which stacked conditional GANs are used to generate unseen characters in a target style from a small number of observed examples. This method is, however, specific to font generation and relies on supervised pre-training using the style labels.

4.2.5 *3D Style Transfer*

Recently, Liu *et al.* [173] showed that style could be learned from one mesh model and transferred to another using a neural subdivision surface scheme. Cao *et al.* [174] generalised the second order statistics approach of [148] to 3D point clouds, adopting the use of a Pointnet [159] encoder pre-trained for classification on ShapeNet [175]. Following the trend of 2D style transfer Segu *et al.* [176] extend this work using GAN methods to produce a generative model with better disentanglement of content and style. There are no existing style transfer/unsupervised approaches to style metric learning for B-Reps.

4.3 METHOD

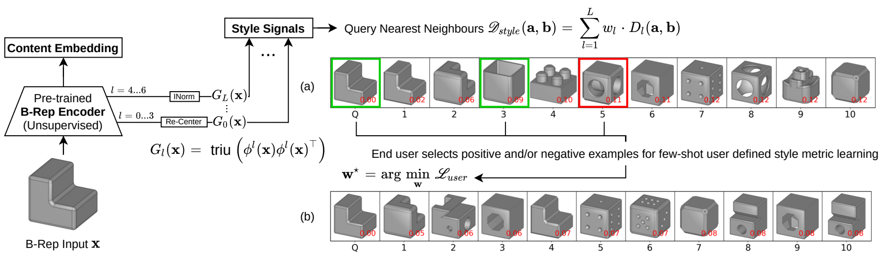

Content Embedding
Style Signals
Query Nearest Neighbours $\mathscr{D}_{style}(\mathbf{a},\mathbf{b}) = \sum_{l=1}^{L} w_l \cdot D_l(\mathbf{a},\mathbf{b})$
Pre-trained
B-Rep Encoder
(Unsupervised)
$l = 4...6$
INorm
$G_L(\mathbf{x})$
$l = 0...3$
Re-Center
$G_0(\mathbf{x})$
$G_l(\mathbf{x}) = \text{triu}\left(\phi^l(\mathbf{x})\phi^l(\mathbf{x})^\top\right)$
(a)
(b)
Q
End user selects positive and/or negative examples for few-shot user defined style metric learning
$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \mathscr{L}_{user}$
B-Rep Input $\mathbf{x}$

Figure 4.3: Overview of UV-StyleNet: Grams of activations are normalized and extracted for each layer. The weights applied to each layer define the meaning of style. (a) Top-10 query results using uniform layer weights $\mathbf{w}$ (b) Top-10 query results using $\mathbf{w}^\star$ based on the user-selected examples (positive in green, negative in red). In this example, $\mathbf{w}^\star \approx [0, 0, 0, 1, 0, 0, 0]^\top$. (We recommend zooming in to see stylistic details such as fillets/bevels.)

Based on our hypothesis that the relational structure present in B-Reps will provide a rich source of information for learning geometric and topological features suitable for representing style, we propose an approach that combines a pre-trained message-passing B-Rep encoder with techniques adapted from image-based style-transfer literature. Specifically, our approach uses second order statistics of the activations from the pre-trained B-Rep encoder to form a flexible style representation. We provide an overview of the architecture in Figure 4.3, with a detailed description below.

For the encoder we use UV-Net [127], which processes each face of a solid with 3 layers of 2D convolutions, and propagates the projected pooled features of each face in a face-adjacency graph using 2 GIN [91] layers. Each face is represented by a 10×10 grid (image) of 7 dimensions containing the absolute 3D position (xyz) of each UV sample, the normal for each sample, and a mask indicating whether each sample lies within or outside of the trimmed face. We use UV-Net due to its state of the art performance on B-Rep classification and its parallels to conventional 2D CNNs. We also hypothesise that the use of a message passing network over encoded face representations will force a greater separation of low and high level features which will be advantageous in separating style from content (see Figure 4.4).

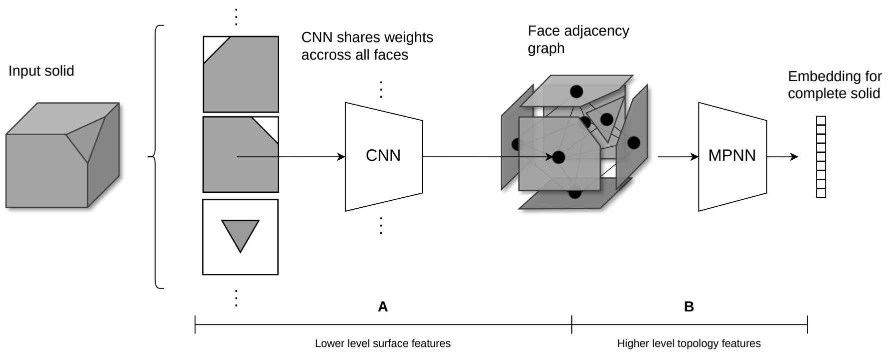

Input solid
CNN shares weights accross all faces
CNN
Face adjacency graph
MPNN
Embedding for complete solid
A
B
Lower level surface features
Higher level topology features

Figure 4.4: General overview of UV-Net encoder architecture. A shared CNN is first used to encode each B-Rep face (A), with the face representations then passed through an MPNN according to the face-adjacency graph (B). Encoding each face prior to passing messages between the faces introduces additional inductive biases which enable the separation of low level surface details from higher level features such as face composition. Figure adapted from [127].

For B-Rep model $\mathbf{x}$, we extract the normalised, flattened upper triangle of the Gram matrix for each layer l:

$$G_l(\mathbf{x}) = \operatorname{triu}\left(\phi^l(\mathbf{x})\phi^l(\mathbf{x})^\top\right) \tag{4.1}$$

where $\phi^l(\mathbf{x}) \in \mathbb{R}^{d_l \times N_l}$ is the normalised feature map of a pre-trained classifier given input $\mathbf{x}$ such that $\phi^l_{ij}(\mathbf{x})$ is the normalized activation of filter i at position j in layer l, d_l and N_l are the number of distinct filters and non-masked samples in layer l respectively, and triu : $\mathbb{R}^{d_l \times d_l} \rightarrow \mathbb{R}^{\frac{d_l(d_l+1)}{2}}$ returns the flattened upper triangle of a matrix.

For the first (features) layer, samples corresponding to the positions that do not lie on the surface of a trimmed face are masked, and the gram matrix is calculated accordingly. In the GIN layers, we have a single vector per face (*i.e.* vertex), thus instance normalization [177] is applied across the solid prior to computing the Grams. For each of the features (non-masked positions and normals) and activations of each convolution layer's filters, we leverage the grouping of samples into faces which is unique to B-Reps (compared to meshes and point clouds), whereby we re-center (subtract the mean of) the UV samples by face. This can be interpreted as per-face instance normalization without division by the standard deviation.

Face re-centering/instance normalization are applied to the activations after extraction from the encoder, but the raw (un-normalized) activations are passed to the next layer of the encoder, thus imposing no requirements on the encoder architecture in terms of normalization strategies.

Analogous to style-transfer with 2D images [148], for a pair of B-Reps $\mathbf{a}$ and $\mathbf{b}$ we define the style distance:

$$\mathcal{D}_{style}\left(\mathbf{a},\mathbf{b}\right) = \sum_{l=1}^{L} w_l \cdot D_l\left(\mathbf{a},\mathbf{b}\right), \tag{4.2}$$

where

$$D_l\left(\mathbf{a},\mathbf{b}\right) = 1 - \frac{G_l(\mathbf{a}) \cdot G_l(\mathbf{b})}{\|G_l(\mathbf{a})\|\|G_l(\mathbf{b})\|} \tag{4.3}$$

and $\mathbf{w}$ is a weights vector that controls how much each layer contributes to the style distance measure. We deviate from Gatys *et al.* [148] in use of the cosine distance (rather than Euclidean) due to simplified normalization and an observed improvement in our initial experiments.

Given a set of user selected examples from a target style (*i.e.* positive samples) T, and a set of user selected counter-examples (*i.e.* negative samples) T', we define the user-defined loss:

$$\mathcal{L}_{user} = \sum_{l=1}^{L} w_l \cdot E_l \tag{4.4}$$

where

$$E_l = c_1 \cdot \sum_{\substack{\mathbf{t}_i, \mathbf{t}_j \in T \\ i \neq j}} D_l(\mathbf{t}_i, \mathbf{t}_j) - c_2 \cdot \sum_{(\mathbf{t}, \mathbf{t}') \in T \times T'} D_l(\mathbf{t}, \mathbf{t}') \tag{4.5}$$

is a layer-wise energy term, c_1 and c_2 are normalization constants, and to prevent trivial solutions $\mathbf{w}$ is constrained such that $\sum_{l=1}^{L} w_l = 1$ and $\mathbf{w} \succeq 0$. Due to these constraints, we note that even with only positive examples T (*i.e.* $T' = \emptyset$), E_l is sufficiently determined, and in such a case the second term may be omitted. However, to reduce the risk of overfitting, a large number of negative examples may be drawn randomly from the remaining dataset. This is of particular benefit in real world settings without access to labelled datasets, where an end user may select only a handful of positive examples that share style as they perceive it.

We find the optimal weights for an end-user

$$\mathbf{w}^{\star} = \arg\min_{\mathbf{w}} \sum_{l=1}^{L} w_l \cdot E_l \tag{4.6}$$

subject to the above constraints, and substitute them into Eq. (4.2) to produce the final user style distance metric.

We observe that E_l is constant w.r.t. $\mathbf{w}$, thus Eq. (4.6) is simply a linear combination and its intersection with the hyperplane $\sum_{l=1}^{L} w_l = 1$ results in a twice-differentiable convex optimization which we solve using Sequential Least Squares Quadratic Programming (SLSQP) [178].

4.3.1 *Intuition Behind the Gram Matrices*

Analogous to [148], we may understand the Gram matrices extracted from the pre-trained B-Rep encoder as follows: At the features layer (input of the encoder), the Gram matrix models the distribution of the position and surface normal of the sample points on the faces with second order statistics. Looking at the filter receptive fields, the 1st layer Gram is modelling the distribution of local simple curvatures (*e.g.* flat/saddle/doubly curved), and the next layer capturing correlations between more complex curvatures (*e.g.* s-shaped), then leading into correlations of patterns of these lower level features and eventually into content. Adjustment of weights for each layer's Gram in the loss enables subjective control over the definition of style [166] by controlling the contribution of shared local curvatures versus more complex surface features.

4.4 EXPERIMENTS & RESULTS

We start by testing if our message passing-based 3D generalization of 2D style techniques for B-Reps is able to capture 3D style signals (4.4.3), and quantify the presence of this information at each layer. We also compare our approach against other encoders that operate on point clouds and mesh, where the information regarding the relational structure of surface primitives is not present. Next, we evaluate our proposed method for disentanglement of style from content with a vertex level gradient visualization (4.4.4), thus demonstrating a practical use-case in which a designer may utilize the feedback from the model. We then test few-shot learning of our style metric in its ability to capture an end-user's subjective requirements (4.4.5). Finally, we assess the effectiveness of our approach when content labels are not available with completely unsupervised encoder pre-training (4.4.6). For ablation, we consider the effect of our proposed face re-centering normalization, which is only applicable for B-Reps given the relational information included in the representation, as well as the effect of dimension reduction on the extracted Gram matrices (4.4.7).

For quantitative evaluation we use SolidMNIST [127], which is a collection of extruded letters from a variety of fonts including labels for both content (*i.e.* letter class) and style (*i.e.* font class) (Table 4.1). This is a good choice of data for initial validation of our design decisions, as the 2D nature of the elements of style in 3D shapes simplifies the analysis and debugging while the generation process of these 3D letters mirrors the most typical CAD modelling approach — drawing a 2D wire body, then extruding to 3D and potentially filleting/bevelling the edges.

In all cases (for SolidMNIST) we pre-train the classifier on the training set to predict the letter classes, and perform model selection for the content classifier with the validation set. Following the methodology of Cohen *et al.* [179] and Jayaraman *et al.* [127], we perform pre-training using 26 classes (combining upper and lower case examples). The dataset provided by Jayaraman *et al.* [127] includes randomness in the fillet size, and extrusion depth and angle. For the held-out test set used in all our evaluations, we regenerate the letters to remove sources of randomness (extrusion angle/amount and fillet size) within font classes, hence strengthening the style labels. For further detail, see Section 4.4.1.1.

	Train	Validation	Test
Examples	40,402	10,100	13,339
Letter Classes	26	26	26
Font Classes	1,664	1650	378
Random Extrude/Fillet	✓	✓	✗

Table 4.1: Details of SolidMNIST dataset [127]. The test set is regenerated without sources of randomness within font classes to strengthen the associated style labels used for evaluation.

After pre-training, all experiments are performed using the held-out test set. We note in particular that no examples of the test fonts are included in the training/validation sets, and that font style labels are used purely for evaluation and not during pre-training.

For comparison with other representation types and encoders, we use MeshCNN [161] for meshes, and Pointnet++ [180] for point clouds. We use Pointnet++ over DGCNN [24] or Pointnet [159] since we are drawing upon 2D style literature. DGCNN aggregates intermediate layer activations according to locality in feature space rather than coordinate space, and Pointnet does not perform hierarchical pooling, thus Pointnet++ is a closer point cloud generalization of the 2D CNN approach used in [148]. In mesh and point cloud representations, there is no information regarding local grouping of samples, thus it is not possible to apply face-wise re-centering, so we use instance normalization for the extracted activations throughout.

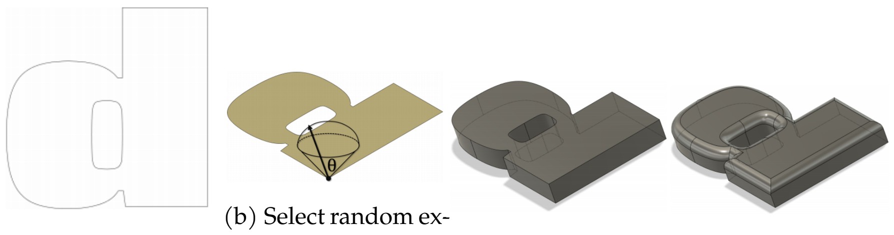
For comparison against existing state of the art, knowing of no existing unsupervised B-Rep style learning methods, as a baseline we use the geometric style embedding of PSNet [174], without the colour inputs, which we refer to as PSNet*. PSNet performs geometric and colour style transfer on point clouds without surface normal. Its use of a pre-trained encoder allows us to adapt it to completely unsupervised settings by pre-training the encoder through point cloud reconstruction rather than content classification as proposed. Further details in Section 4.4.2.

4.4.1 *Data*

4.4.1.1 *SolidMNIST Test Set Generation*

For SolidMNIST, the training data is generated as per [127] using code and font wires provided by the authors. The key steps are illustrated in Figure 4.5.

The held-out test set is regenerated to strengthen the associated style labels by removing inconsistent sources of randomness within font classes. The extrusion depth and angle are fixed across all fonts. Filleting size is also fixed, and is applied only to fonts where it possible to apply it to all examples of that font. Filleting is not possible for some examples due to the complexity of the solids. If filleting is unsuccessful on any example, all examples of that font are left without fillets.



(a) 2D Font wire (b) Select random extrude angle (c) Extrude (d) Fillet

Figure 4.5: Steps for generation of SolidMNIST dataset. For test set, extrude angle and fillet amount are fixed. Figure from [127]

4.4.1.2 *ABC Style Labels*

There is a fundamental lack of publicly available labelled B-Rep data, with no existing B-Rep datasets containing style labels. To enable quantitative evaluation of our method and promote further work in this area we contribute a set of manually assigned style labels for a subset of the

ABC solid models. We selected categories with distinct styles while containing diverse content. Examples of each category are shown in Figure 4.6 and details of the class sizes in Table 4.2.

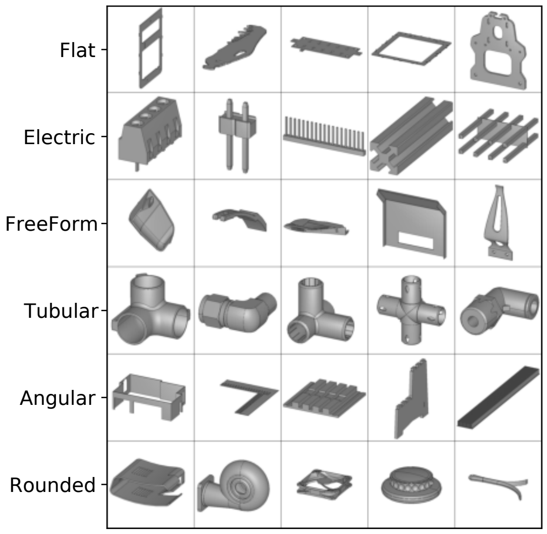


Figure 4.6: Examples of each ABC style subset classes. Each style is selected to be visually distinct, and while some classes contain the same types of objects, *i.e.*, 'Tubular', the overall shapes (the content) are diverse.

ABC Subset	Examples
Flat/Electric	389/58
Free Form/Pipe	241/24
Angular/Rounded	834/106

Table 4.2: Manually labelled ABC style subsets.

4.4.2 *Model Details & Hyperparameters*

For MeshCNN [161] we use the authors' code from `https://github.com/ranahanocka/MeshCNN`, for Pointnet++ [180] we use `https://github.com/erikwijmans/Pointnet2_PyTorch`. All experiments performed on AWS p3.2xlarge.

Table 4.3 shows details about the model hyper parameters and meta information. For MeshCNN, we remeshed each solid to 15000 edges and for Pointnet++ we used the multi-scale grouping (MSG) setup. Other parameters and architecture choices not mentioned here, are set to default.

Model	LR	N	F	BS	Opt
UV-Net	1e-4	BN	7	128	Adam
PSNet*	1e-4	BN	3	128	Adam
Pointnet++	1e-3	BN	6	32	Adam
MeshCNN	2e-4	GN	5	4	Adam

Table 4.3: Hyper-parameters and meta information about the models for SolidMNIST runs. LR denotes learning rate, N type of norm (*i.e.* batch norm or group norm), F input feature dimension, BS batch size and Opt, the type of optimizer used.

For PSNet* [174] we use the Pointnet [159] implementation from `https://github.com/WangYueFt/dgcnn` and extract the Gram matrices from the first 4 layers as detailed in [174]. While PSNet works with geometry and color, we use only the geometric part in our comparisons.

All point clouds are sampled with 1024 points.

4.4.3 *Measuring Style Signal*

We adopt the Linear Probe methodology [181] to measure the amount of style signal present in the Gram matrices of each layer of the pre-trained network. We train a linear classifier on each layer's Gram matrix G_l with ground truth font labels on a subset of the SolidMNIST test set. We select four visually distinct fonts in order to strengthen the style labels with respect to style over style compatability (see Figure 4.1), and due to many fonts in the test set containing almost identical variants. Each encoder is pre-trained with only letter classes as labels, and the four test fonts used in this evaluation are previously unseen. Since the dimensions of the Gram matrices are very large (*i.e.* in some cases $> 2^{19}$), but we have only 137 examples, we perform logistic regression with L2 regularization and 5-fold cross-validation to prevent overfitting. We report the mean validation accuracies.

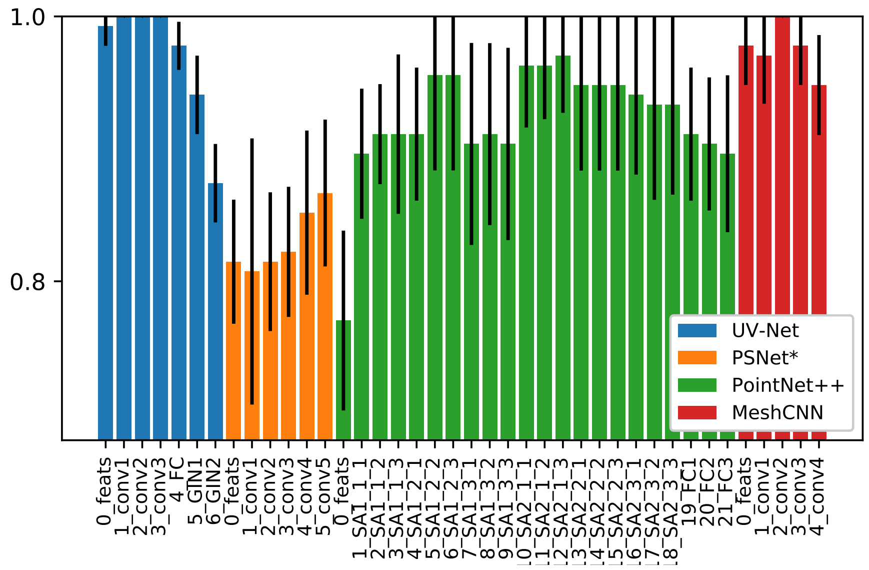


Figure 4.7: Linear probe classification accuracy scores for each encoder using font labels for evaluation (no font labels used during pre-training). All fonts used here are previously unseen by the networks. Random baseline: 0.25.

Figure 4.7 shows the mean validation accuracy using the extracted Gram matrices from each layer in all four pre-trained models. Compared to random baseline at 0.25, we observe significant indication of style being present in the signals extracted from all layers (including features) for all models. For UV-Net we see the greatest amount of style information in the lowest layers, with the signal reducing deeper into the network. This aligns with our assumption that second order activation statistics transition from style to content representations as network depth increases, as shown for 2D images in [148, 167].

Comparing the distributions of classification accuracy over each layer with the median as shown in Table 4.4, we see a clear separation of UV-Net and MeshCNN which perform message passing over existing relational structure, from PSNet* and Pointnet++ which do not. For Pointnet++, the relational structure over which to perform mes-

Encoder	Relational Structure	Median Accuracy	Rank
UV-Net	FG + FA	0.99	1
PSNet*	None	0.81	4
Pointnet++	Inferred	0.92	3
MeshCNN	Mesh	0.98	2

Table 4.4: Comparison of classification accuracies for each pre-trained encoder with attention to the relational structure which defines the message passing. UV-Net and MeshCNN leverage existing structural information, whereas Pointnet++ must infer the structure, and PSNet* treats all points as a set with no relational information. FG = Face-grouping of sampled points, FA = Face adjacency.

sage passing must be inferred from the points' *xyz* positions, whereas MeshCNN leverages the relational structure prescribed by the topology of the mesh, and UV-Net leverages both the relational information of which samples belong to which B-Rep face as well as the topological information prescribed by the face-adjacency graph. PSNet*, which does not perform any message passing over relational structure, performs substantially lower than all three message passing methods.

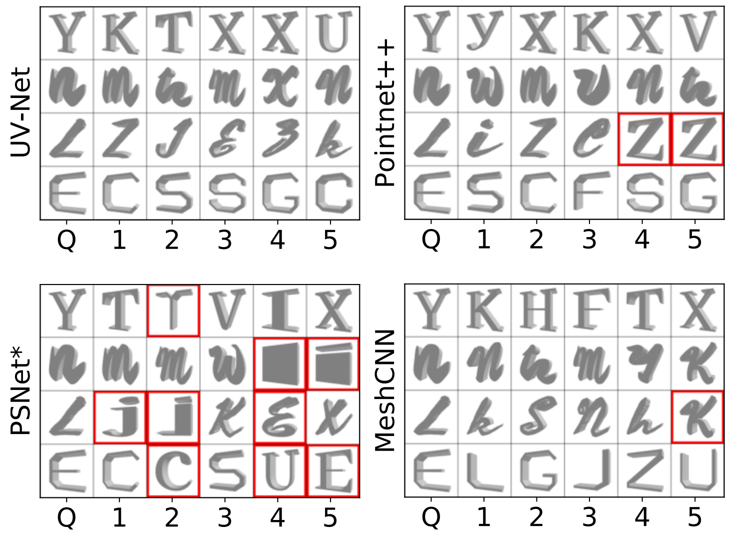


Figure 4.8: SolidMNIST Font Subset: Top-5 queries for a letter from each font, with all weight distributed uniformly over the first $\frac{L}{2}$ layers. Red box indicates result does not match query font.

For a qualitative evaluation of our design choices, we perform a top-k query for an example from each font distributing all weight uniformly over the first $\frac{L}{2}$ layers. As shown in Figure 4.8, with this particular style definition, the style features provided by the pre-trained Pointnet++ model suggest the 'Z' from another font is close in style to the query 'L', while all UV-Net query results match the target font, and MeshCNN makes only one less obvious mistake. PSNet* has the highest number of errors.

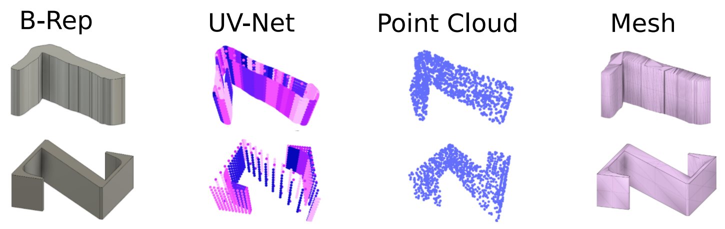


Figure 4.9: Visualization illustrating the sampling bias advantage of UV-Net, whereby the details in the long surfaces of the 'L' are sampled more densely (each face in the B-Rep is sampled with a uniform 10x10 grid) than the simple flat surface of the 'Z' making it much easier to differentiate between the different styles than with the uniformly sampled point cloud.

In addition to the differences in the amount of relational information utilized by each encoder, we hypothesize that this result may be partly due to the sampling strategy of each method. As Figure 4.9 illustrates, UV-Net samples a fixed size grid for each face, thus large faces (such as the long diagonal stem of the 'Z') will contribute less to the style features extracted than in PSNet* and Pointnet++ where the point cloud is sampled with uniform density. Therefore, the large diagonal faces have larger influence with Pointnet++ features as network depth increases. Lack of a CNN hierarchy and surface normal inputs may explain the lower performance of PSNet* versus PointNet++.

Figure 4.10 shows the top-k query for the same letter 'L' using the style distance from single layers ($l = 0, l \approx \frac{L}{4}$, and $l \approx \frac{L}{2}$). Supporting our hypothesis above, we see that in this particular scenario, the font is better matched by Pointnet++ in the lower layers. Within the first layer of the network, the features extracted will contain more informa-

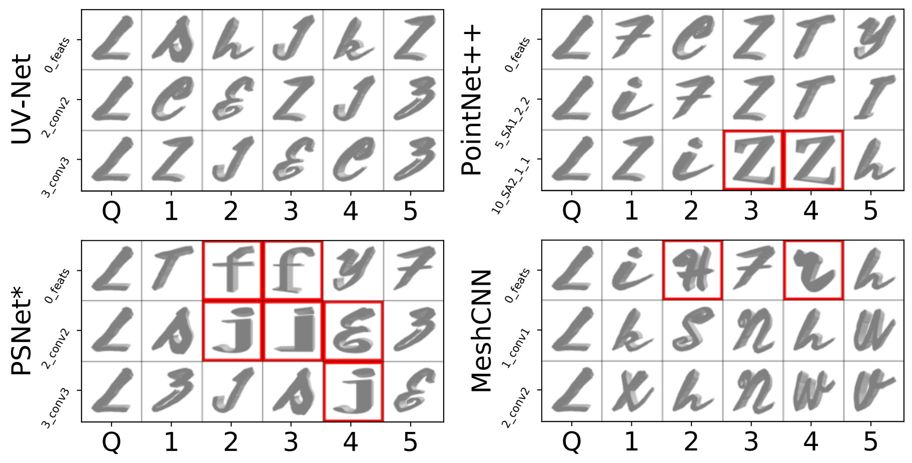


Figure 4.10: SolidMNIST Font Subset: Top-5 queries for the same letter for $l = 0, l \approx \frac{L}{4}$, and $l \approx \frac{L}{2}$. Red box indicates result does not match query font.

tion about low level structure, *i.e.* bumpy rather than smooth surfaces. Interestingly, for $l \leq \frac{L}{2}$, MeshCNN performs worst with the features $(l = 0)$. We hypothesise this is due to the rotation and scale invariance in the MeshCNN features, whereas UV-Net/Pointnet++ features contain global information.

Encoder	L	Parameters	Time	Size
UV-Net	7	**645,596**	93min/**88s**	**199 KB**
PSNet*	5	813,914	165min/115s	1.08MB
Pointnet++	22	1,746,420	**43min**/603s	3.32 MB
MeshCNN	5	1,322,982	29hr/38min	305 KB

Table 4.5: Comparison of 3D encoder methods. L is total number of layers (including features), times given are pre-training/style inference on complete SolidMNIST test set. Size is the memory required for a single style embedding (containing one Gram per layer) for a single solid - note this is not dependent on the size of the input solid. For style inference UV-Net is the most compute and memory efficient. MeshCNN suffers from small batch size due to necessarily large meshes, and Pointnet++ suffers from larger Gram matrices.

Finally, comparing with the computational costs of PSNet*, PointNet++, and MeshCNN we observe that the UV-Net encoder with only 645K parameters is 23, 85, and 96 percent faster for style inference, and the Gram matrices require 82, 94, and 35 percent less memory per solid respectively. The superior inference speed of the UV-Net encoder is likely due to the sparsity of the face adjacency graph over which messages are propagated. Whereas the Pointnet++ and MeshCNN message passing graphs are relatively large compared to the number of points or edges in the input solids, the face adjacency graphs of B-Reps are substantially smaller, as even complex faces can be represented efficiently with a single vertex. Full details in Table 4.5. Based on the above results and computational costs, we perform further experiments using the UV-Net encoder only.

4.4.4 *Gradient Visualization*

In Figure 4.11 we visualize our proposed pairwise style distance metric for each B-Rep $\mathbf{x}$ by computing

$$\nabla_{xyz} = \frac{\partial \mathcal{L}_{style}}{\partial \mathbf{x}_{xyz}} \in \mathbb{R}^{N_0 \times 3} \tag{4.7}$$

where N_0 is the grid size (number of unmasked UV samples), and $\mathbf{x}_{xyz}$ is the absolute positions of the UV samples. For easy interpretation, we plot the vectors $-k \cdot \nabla_{xyz}$ centered at the samples $\mathbf{x}_{xyz}$ with black lines to indicate the direction in which a UV sample point should be displaced in order to better match the style between the pair, and k is a constant scaling factor that aids visualization.

In Figure 4.11 (left) we fix the content and compare different styles. The xyz gradients suggest that the samples of the left example should be moved outwards to match the squarish style on the right, and the samples of the example on the right should be moved inside the solid to match the curves on the left. Figure 4.11 (right) confirms our approach is able to disentangle style from content, as we compare different content and different style. The gradients on the left example are similar to

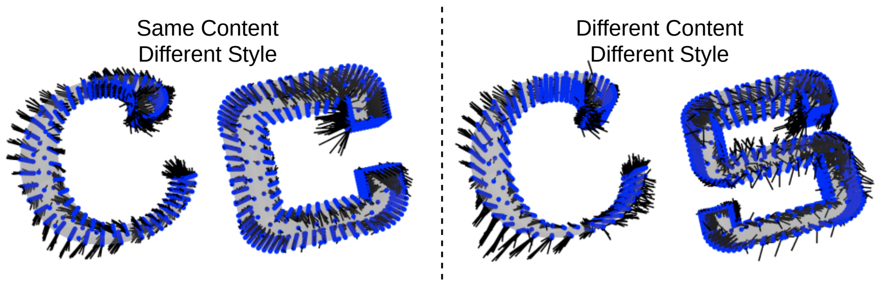


Figure 4.11: Gradient visualizations of pairwise $\mathcal{D}_{style}$ loss (Eq. (4.2)) with uniform weight on the first 4 layers (including features), *i.e.* $\mathbf{w} = [\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, 0, 0, 0]^\top$. Black lines show $-k \cdot \nabla_{xyz}$, *i.e.* the direction in which to move the point to match the style between the pair.

(left), confirming that the style is matched despite a different content example to compare with.

The ability of our method to generate these meaningful style gradients demonstrates a key advantage to our use of a message passing network. As discussed in Chapter 2, alternatives to message passing for representing non-Euclidean data, such as multi-view 2D projections, would make it non-trivial to project the gradients back to the original problem space (*i.e.* a 3D solid), whereas by adopting a message passing approach which operates directly on the non-Euclidean data we are able to back-propagate the gradients directly to the solids in the 3D coordinate space.

4.4.5 *Few-shot Learning of User-Defined Style Measure*

We evaluate few-shot learning of our user-defined style loss on the complete unseen test set, by measuring the mean Precision@10 for each example from a selected font for a range of number of positive and negative user-selected examples. We evaluate on 6 visually distinct fonts. Examples of each font are given in Figure 4.12. Precision@10 is calculated as the proportion of top-10 neighbors that match the target font. For baseline, we compare against the mean Precision@10 with uniform layer weights (one positive and no negative examples). For computational reasons, we reduce the dimensions of each layer's representation G_l to $\min(d_l, 70)$ using PCA.

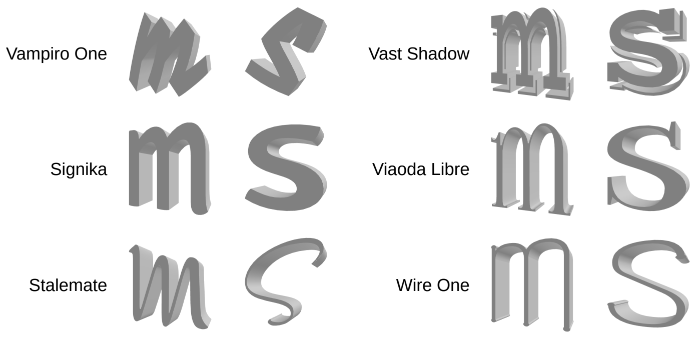


Figure 4.12: Examples of fonts for few-shot evaluation.

As shown in Figure 4.1, the font name provides only a weak style label, and as such we are concerned more with the improvement in the mean Precision@10 score than the absolute values. We also consider upper and lower case within the same font as separate labels to further strengthen the associated label, yet also increasing the difficulty of the task as the number of classes doubles to 756.

Positive examples are randomly drawn from the same font and case, and negatives are drawn randomly from all remaining examples. For each number of positives and negatives we perform 20 trials (different positives and negatives each time). We report the mean Precision@10 across all examples of the positive font across all trials, *i.e.* for each number of positives and negatives, every example of a chosen font is queried and evaluated, and this process is repeated 20 times.

Figure 4.13 shows the absolute mean Precision@10 scores over a range of number of positive and negative examples of each of the unseen fonts we tested. 1 positive and 0 negative indicates baseline using equal layer weights.

For the most visually distinct fonts (*i.e.* 'Vampiro One' and 'Vast Shadow'), the equal weights baseline is highest. The amount of improvement is dependant on the self-consistency of style within the font

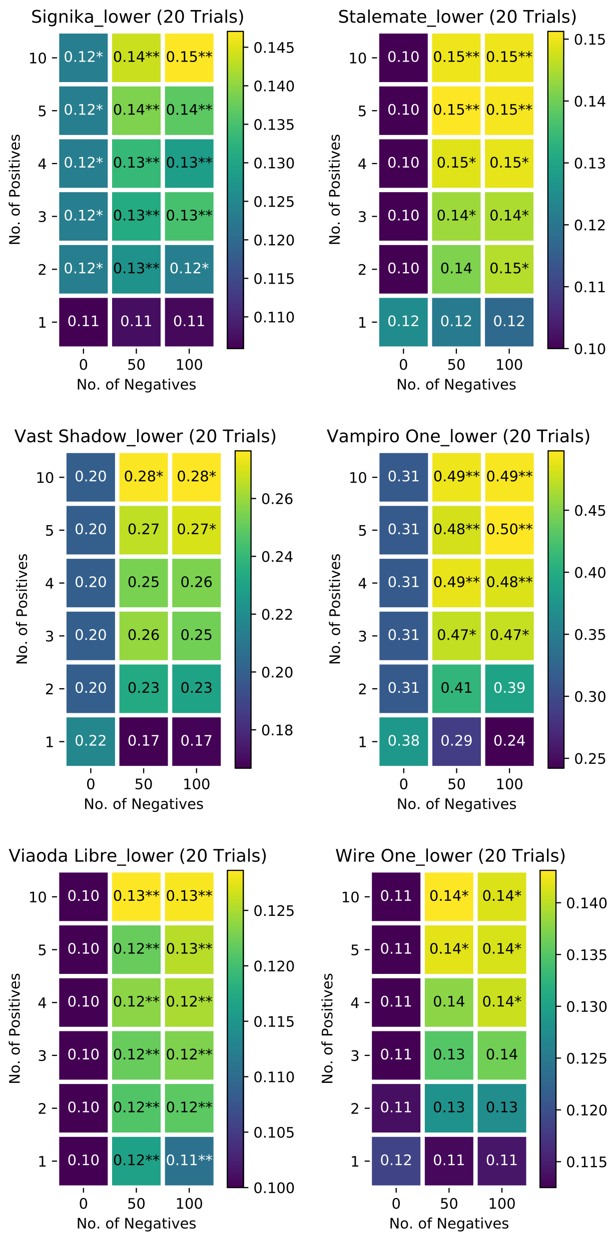


Figure 4.13: Mean Precision@10 score for each example of the specified font after few-shot learning of $\mathbf{w}^\star$ given a range of number of positive and negative examples. 1 positive + 0 negatives provides baseline using uniform weights. * and ** indicate a 10% and 5% statistically significant improvement over baseline respectively.

and the number of similar fonts in the test set. We observe greater self-consistency within 'Vampiro One' and 'Vast Shadow' while being distinct from the rest of the test set. While the other fonts still show improvement, we expect lower results due to their inconsistency or lack of distinct stylistic features, *i.e.* in 'Stalemate' the 'm' and 's' appear to

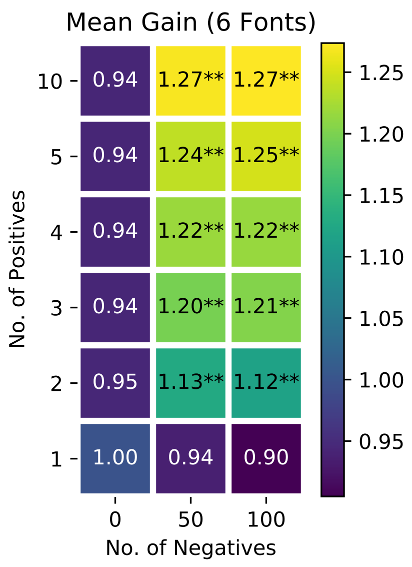


Figure 4.14: Mean gain in Precision@10 (ratio to baseline) for all six fonts. * and ** indicate a 10% and 5% statistically significant improvement over baseline respectively.

be stylistically compatible, but the max curvatures of the 'm' are much greater than in the 's' - the style is not obviously the same.

Figure 4.14 shows the mean gain in Precision@10 (ratio to baseline) for all six fonts. For all combinations of number of positives and negatives greater than 0, we observe a significant improvement in the mean Precision@10 score over the uniformly weighted baseline. Moreover, since negatives are selected randomly from the remaining dataset, we also confirm that providing only positive examples is sufficient to obtain a significantly improved style measure based on the end-user's requirements.

4.4.6 *Unsupervised Pre-training*

Another advantage of our method over existing approaches is that it may be used in unsupervised settings. This is particularly important for B-Reps, since there are no publicly available B-Rep datasets with style labels. We evaluate our approach using the ABC dataset, which contains no content or style labels. For the UVStyle-Net/PSNet* encoder pre-training we use an auto-regressive approach with point cloud reconstruction [127]. Again, we reduce the dimensions of the style representations G_l to $\min(d_l, 70)$ using PCA.

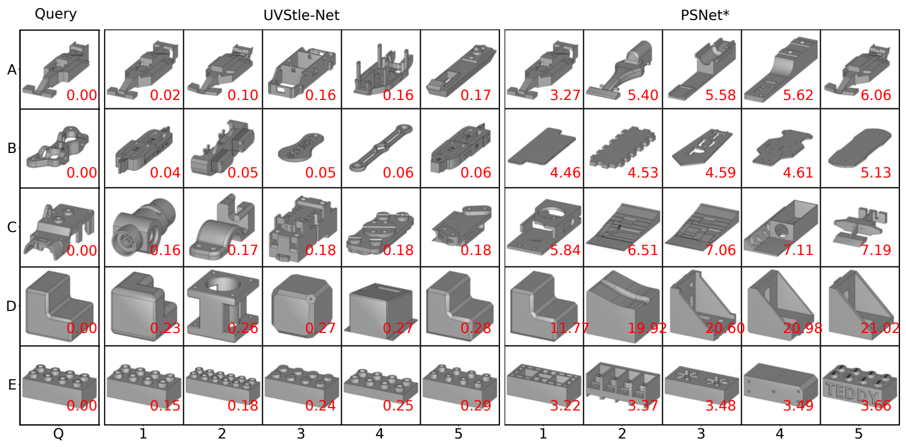


Figure 4.15: Top-5 query results for ABC dataset from UVStyle-Net and PSNet* pre-trained (unsupervised) with point cloud reconstruction. For UVStyle-Net $\mathbf{w} = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0, 0, 0, 0]^\top$. (We recommend zooming to see important stylistic details such as bevels/fillets.)

Figure 4.15 shows a selection of top-5 queries in the style embedding space, with uniform weight on only the lowest 3 layers. For PSNet* queries we use Euclidean distance as this is the metric optimized in [174]. We observe that UVStyle-Net matches surface style with more variation in content, while in many cases PSNet* matches shapes that roughly occupy the same regions in space as the query, *i.e.* the content.

For example, in row A: the UVStyle-Net results have more flat surfaces and similar angles matching the query, while PSNet* results have more curved surfaces (not present in the query). For row C: UVStyle-Net has more matching curved surfaces (cylindrical parts), and in row D: UVStyle-Net matches more shapes with fillets (as per the query), while PSNet*'s closest query result has the exact same content but with a different (bevelled) style (this same bevelled example is in position 5 for UVStyle-Net rather than 1 for PSNet*). Finally, in row E: UVStyle-Net finds blocks with the matching notch style (even with different block size or numbers of notches), whereas PSNet* matches similar sized blocks without the notched style. For completeness, we show the same queries for PSNet* using the cosine loss in Figure 4.16.

The effectiveness of UVStyle-Net over PSNet* is most likely due to the hierarchical feature learning architecture enable by message passing. In UV-Net, the composition of a CNN face encoder followed by 2 GIN [91] layers enables the CNN to focus purely on low level surface features, with higher order features such as face composition learnt higher in the network in the message passing layers (see Figure 4.4). In this case, for matching style over content, being able to focus solely on surface level details is clearly advantageous. Contrary to this, PSNet* treats all points as a set with no relational information to assist in breaking the solid down into hierarchically composed parts, thus lacks this key relational inductive bias.

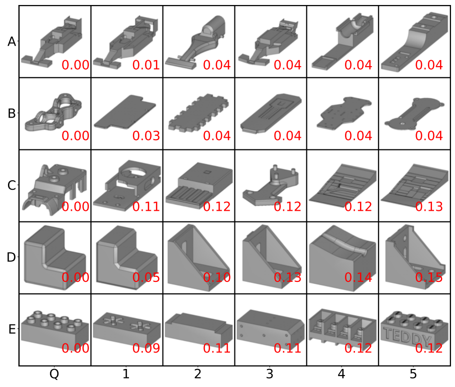


Figure 4.16: Top-5 queries for PSNet* with cosine distance.

Figure 4.17 shows a comparison of the same queries for UVStyle-Net with different layer weights. Of particular interest is row D column 5 (left) and column 2 (right). As discussed above, this solid is almost identical to the query but with bevels instead of fillets. We observe that adding weight to the fourth layer moves this result closer to the query, while when only weighting the first three layers, the difference in surface detail is enough for other filleted examples to be closer in the embedding space to the query than the bevelled version.

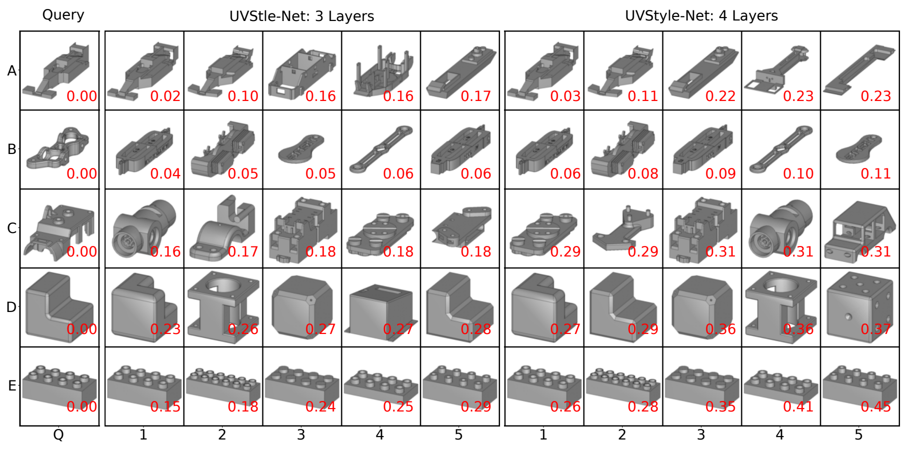


Figure 4.17: Comparison of top-5 queries with different weights for UVStyle-Net on ABC dataset with unsupervised pre-training. 3 Layers: $\mathbf{w} = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0, 0, 0, 0]^\top$, 4 Layers: $\mathbf{w} = [\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, 0, 0, 0]^\top$.

In Figure 4.18, we again show the same queries, however this time with all weight on the final layer. As expected, we see that weighting the upper layers of the network moves the definition of style closer to content, where the distance measure is more about the general shape and size and global features, and less about the fine details and local features. Again, this is most likely due to the face-adjacency message passing architecture, where each message in the MPNN layers contains information regarding a larger local neighbourhood of adjacent faces as depth increases.

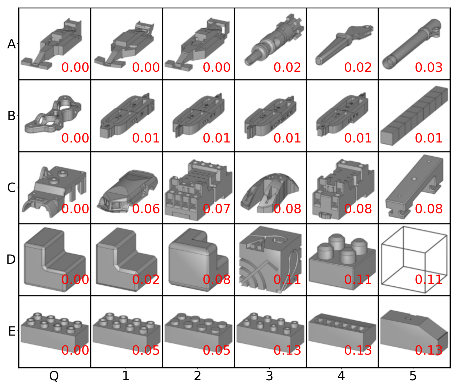


Figure 4.18: Top-5 query results for ABC dataset from UVStyle-Net with unsupervised pre-training. $\mathbf{w} = [0, 0, 0, 0, 0, 0, 1]^\top$.

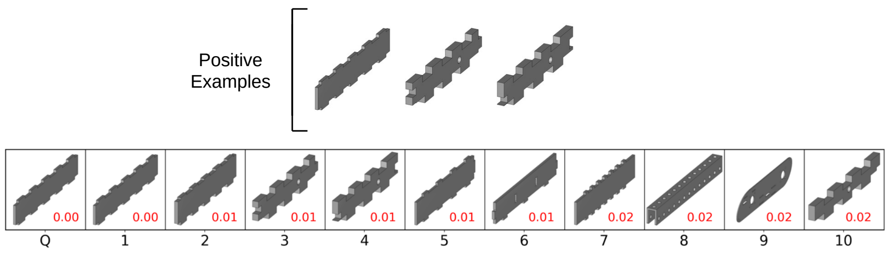


Figure 4.19: Optimizing $\mathcal{L}_{user}$ with positive examples matching in content results in layer weight distributed over the upper layers. $\mathbf{w}^{\star} \approx [0,0,0,0,0,\frac{1}{3},\frac{2}{3}]^{\top}$.

Evaluating our few-shot user-defined style measure, Figure 4.3 shows the nearest neighbour queries for a given target after optimizing the style loss for the user selected examples shown. Selecting filleted solids for positive and a bevelled solid for negative improves the nearest neighbours to the target by pushing away the nearest neighbour of (a) which matches closely in content but not the filleted style.

Figure 4.19 shows our few-shot loss optimized for examples with matching content. Supporting our hypothesis that the Gram matrices transition from style in the lower layers to content in the higher layers, we observe that all weight is distributed in the last two layers, with the majority of the weight on the final layer.

For quantitative evaluation on a real-world dataset, we use subsets of ABC for which we manually curate style labels (details in Section 4.4.1.2). For each model we perform logistic regression on the extracted style embeddings from the pre-trained encoders. Again we train the encoders using point cloud reconstruction on the complete ABC dataset. We perform 5-fold cross validation with L2 regularization and report the mean validation weighted F1 scores for the best parameters, summarised in Table 4.6 showing UVStyle-Net significantly outperforms PSNet* on all subsets.

ABC Subset	UVStyle-Net	PSNet*
Flat/Electric	**0.789 ± 0.034****	0.746 ± 0.038
FreeForm/Tubular	**0.839 ± 0.011****	0.808 ± 0.023
Angular/Rounded	**0.805 ± 0.010****	0.777 ± 0.020

Table 4.6: Weighted F1 scores for each manually labelled styles subset of ABC. ** indicates 5% statistical significance.

4.4.7 *Ablation*

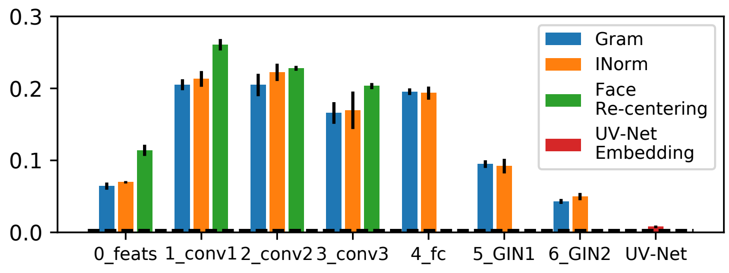


Figure 4.20: Linear probe scores on complete SolidMNIST test set with and without instance/face normalization. Dashed line indicates random classifier baseline.

Figure 4.20 illustrates the impact of face re-centering and instance normalization using the complete SolidMNIST unseen test set. Adopting the linear probe methodology as above, we compare the mean classification accuracy of each layer for predicting all fonts using 5-fold cross-validation. While instance normalization is tested on all layers, face re-centering is not possible beyond the third convolution layer since each face is already represented by a single vector.

The significantly higher scores in the lower layers (excluding features) confirms our assumption that style transitions into content deeper in the network. We also see empirical justification for the use of instance normalization, and in particular face re-centering, which is not possible when working with meshes or point clouds. This supports our hypothesis that the additional relational structure of B-Reps is advantageous for learning style representations. Comparison with the

UV-Net content embedding shows than any of the layer-wise style representations (G_l) as proposed in our method are better suited to capturing style information.

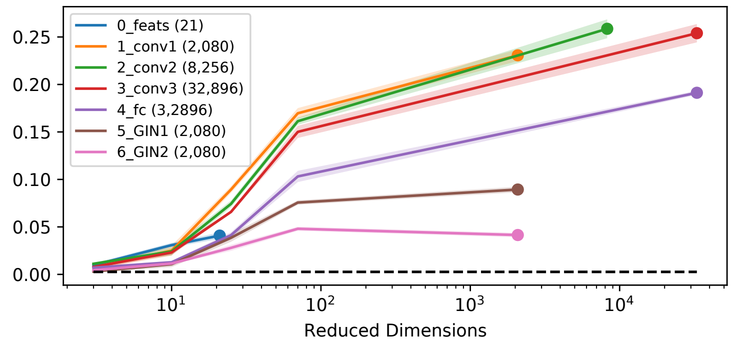


Figure 4.21: Linear probe scores for each UV-Net layer on complete SolidMNIST test set as number of dimensions are reduced. Original dimensions shown in parentheses and marked with •.

Figure 4.21 shows the effect of PCA on the layer-wise style representations (G_l) to test the significance of style as a source of variation in each layer. Again, we use linear probes to quantify the style information. In line with our assumptions, the lowest layers $(l = 0 \ldots 3)$ show the greatest amount of style information when the dimensions are sufficiently low, thus indicating that the font style signals are the most significant source of variance in these layers.

4.5 SUMMARY

We have proposed UVStyle-Net, a 3D style similarity measure for B-Reps which caters for the end-user's subjective definition of style through few-shot learning based on user selected examples, and a completely unsupervised pre-trained encoder. As a purely data driven style measure for B-Reps, which does not require style or content labels yet is adaptable to end-users' requirements, our approach is unique from all existing methods. Moreover, through leveraging the additional relational structure available in B-Reps through point sampling in the surface parameter domain and face re-centering followed by face-adjacency message passing, we have demonstrated superior performance compared to other non-Euclidean methods without access to this information, or compared to B-Reps without utilizing the face membership groupings of the samples.

Using the SolidMNIST font labels for evaluation, our results have demonstrated the applicability of 2D image style principles and assumptions for 3D shapes, and quantified the advantages of our method with B-Reps over alternative methods on meshes and point clouds. In particular, we have confirmed that even for non-Euclidean representations, the second order statistics of 3D encoder activations in the first few layers contain style information as the greatest source of variance. We have also shown that our method generates meaningful vertex level style gradients, and that the UV-Net sampling strategy and leveraging the face boundary information unique to B-Reps, particularly through face re-centering, significantly improves the style measure.

For a range of 3D fonts and real-world CAD models, we have demonstrated that our proposed method for few-shot learning of user-defined style is effective in improving the style measure for a specific task, even with a minimal number of positive (and optionally, negative) examples. We also demonstrate the benefits of our approach over an existing state of the art method on the real-world ABC dataset where even content labels are not available for encoder pre-training.

A limitation of our method can be seen when considering solids with very similar content. Further work should consider stronger disentanglement of style from content in such cases. We hypothesize that other unsupervised methods for the encoder pre-training may capture greater detail in the network activations, and therefore improve the style measure on very similar content. We also observe that the current formulation of the few-shot learning often puts all weight on one layer.

For future work, we propose investigation into regularization of the few-shot user loss and further investigation into sophisticated distance measures for comparing feature distributions, as well as incorporating further relational structure regarding face-adjacency angles/edge features, and the natural next step of B-Rep style transfer. We discuss some of these areas in further detail in Section 6.3.

Having now investigated two application areas for message passing with multiple domains on both vertex and graph level tasks, next we consider the case of a dynamic domain in tracing cell lineage in cancerous tumour growth.

5 MESSAGE PASSING FOR INTERACTING DYNAMIC NETWORKS

The following chapter presents work published in [4]. The reviewers commended the use of a "popular graph database" as an execution environment, as well as the ability to query the model post-execution. They also note the novelty of our approach, and the relevance of the problem domain. The most significant criticism of this work was a "lack of clarity" in explaining the model architecture, which we addressed with the inclusion of additional figures and more detailed text. We thank all reviewers for their contribution to this work.

5.1 INTRODUCTION

Having considered the multiple domain case for graph level tasks including graph classification and style representation learning, and the multiple domain task of gradient visualizations, we now consider the single domain case on a dynamic domain of hierarchically composed systems.

Hierarchy is overwhelmingly evident in every aspect of life, emerging in any imaginable circumstance as a direct consequence of evolution. Simple structural hierarchies can be seen in everything from the organisation of the stars to object-oriented programs. However, in biological contexts, the hierarchies that emerge are often *dynamic*, and involve complex dependencies between components that do not exist at the same scales.

A typical example of dynamic hierarchy is that of proteins, cells, and biological organisms. In this case, the building blocks (proteins) have a direct impact on the behaviour of the higher order entities (cells and higher still, organisms), but higher order functions equally have an impact on the configuration of the building blocks [182].

Moreover, interactions between the lower order entities may also affect the entity indirectly, and its composition may change. For example, it is the interaction of particular proteins that facilitate the protein aggregation mechanism by which the neuronal degeneration of Huntingdon's disease is caused [183]. These interactions within and between biological hierarchical networks make them highly dynamic, potentially changing every part of themselves from the organisation to the functioning of the components.

Due to their abundance in nature, the utility in simulating complex dynamic networks cannot be overstated; with modelling applications in medicine, biology, macro-economics, and other ecological sciences. However, capturing the complexities of interacting multi-scale systems that are able to change their internal configurations and behaviours dynamically in a computational model is not a trivial task.

In this chapter we propose an original, graph-based model of computation for the simulation of Interacting Hierarchical Dynamic Networks (IHDNs), where the representation of components of different scales combined with a novel cross-layer message passing system enables the simulation of complex adaptive systems across any scales of abstraction.

Message passing is an appropriate choice to enable the simulation of such systems, since it offers a simple yet efficient way in which to leverage the dynamic relational structure within the networks to inform the resulting behaviours. In this work, we leverage this relational structure through voting messages which are aggregated and filtered according to properties held by each vertex, as well as updating these vertex properties according to the messgaes that are recieved. In doing so, our method enables a form of simulation that directly corresponds to the original problem space, thus simplifying the process of post analysis.

In order to present the proposed model, first we review relevant literature not considered in Chapter 2. Then after presenting the concepts and architecture of the model, we demonstrate its ability to emulate the behaviour of living systems with the simulation and analysis of tumour growth in a dynamic evolving cell network, thus demonstrating the effectiveness of message passing in capturing the dynamic structural information required to model a biological system. We verify our results against an existing model of this phenomenon [38], that has also been used to test another unconventional model of computation [184], then we show the advantages of the model for deeper analysis.

Source code for both the computational platform and the demonstrated aneuploid tumour growth simulation are available for reference at `github.com/meltzerpete/ihdn`.

5.2 BACKGROUND

The modelling of complex dynamic systems has employed the use of many solutions; including ABMs [185], and more recently Dynamic Networks [186]. In this section we review a sample of these solutions.

Although emerging from the object-oriented programming paradigm, ABMs have many parallels with Cellular Automata, which have also been used to model complex systems [187, 188]. ABMs have been particularly popular in modelling complex social systems in order to observe emergent and collective behaviours [189].

The modelling methodology for ABMs typically begins with (deductive) observations of real world phenomena in order to derive agent state update rules. With the agents' update rules defined, simulations can be executed wherein (inductive) analysis of emergent properties can be made. Consequently, [189] and [190] argue that ABMs offer a distinct "third" scientific method *i.e. generative* science.

In more recent years, Multi-Agent Systems have been developed [191], and ABMs have been combined with Reinforcement Learning [192], in which agents' policy functions are optimised to minimize the distance between simulated and real-world observed data. In an attempt to better capture the inherent hierarchy in naturally occurring complex systems, models such as [193] and [194] define layers for hierarchical organisation of agents. However, these models only allow for a finite number of layers and configurations; and hence, as with ABMs in general, are restricted in their representation of dynamic systems.

Different in their approach, Dynamic Networks have been applied to modelling complex phenomena found in epidemics [195], social networks [196], and neuroscience [197]. However, these applications are typically concerned with either the changing states of fixed topology networks (of which conventional Artificial Neural Networks are a prime example), or the changing properties of a network based on topological transformations alone.

Contrary to this, [186] provides a framework for the uniform representation of state-topology co-evolution via graph-rewritings, with a demonstration of automated rule discovery using real-world observed network evolution data [198]. However, as a consequence of decoupling the representation of entities and their behaviours, these models do not achieve the same expressiveness of ABMs in describing the effects of small changes in individual systems on the dynamics of the whole [37].

5.3 INTERACTING HIERARCHICAL DYNAMIC NETWORKS

5.3.1 *The Model of Computation*

The purpose of this work is to apply message passing to leverage the existing relational structure in dynamic hierarchical systems. In order to do that we will create a useful model of computation, which relies purely on message passing to inform the dynamic behaviours of the entities in question. Thus we require a data model which expresses hierarchical and compositional relations as a graph, therefore allowing message passing to capture this structural information.

To enable simulation involving dynamic behaviours and emergent effects we also have the following additional requirements:

- An efficient means of interpreting the graph-based data model during computation
- The ability to update the functions of, and allow for the dynamic creation and deletion of the simulated components
- The ability to model multiple scales of abstraction with a simple, uniform data model.

To create a homogeneous yet general data model, we define a single component of computation: the *system* (see Figure 5.1). As in a property graph, a system S may have a set of any number of labels L^S, and a set of any number of properties P^S in the form of key value pairs. In addition, a system may optionally be given a vote function F_V^S, a filter vector I^S, and a vote vector V^S. An ordered set of all possible system functions F is shared by all systems, with $|F|$ denoting the total number of defined functions.

The system may have a set of any number of child systems $C^S = \{C_1^S, C_2^S, ...\}$ and (excluding ROOT systems) will have always at least one

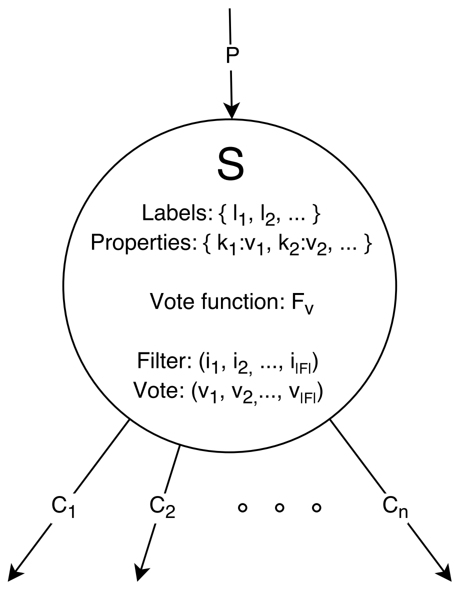

P
S
Labels: { l₁, l₂, ... }
Properties: { k₁:v₁, k₂:v₂, ... }
Vote function: F_v
Filter: (i₁, i₂, ..., i_|F|)
Vote: (v₁, v₂,..., v_|F|)
C₁
C₂
Cₙ

Figure 5.1: The abstract IHDN system model.

parent system P^S. Relationships indicating compositional hierarchy are labelled with the `CONTAINS` relationship type, while any other user defined relationship types may exist between any pair of systems. Typical operations for topology mutation include `deepClone` and `transfer` (see Figure 5.2).

Computation (here the update of system state - internal and/or structural), then proceeds via a depth-first traversal over the `CONTAINS` relationships initiated at every `ROOT` system (in a random order), once per *iteration*. The traversal facilitates a message passing system that enables systems to influence the selected actions of others at different levels in the hierarchy.

When each system is selected to perform an action, the functions are selected from F probabilistically, according to the messages received by that system. There are two types of message passed between systems in the hierarchy: filters are passed down the tree; and votes are passed upwards. The elements of both the filter and vote vectors correspond directly to the functions of F.

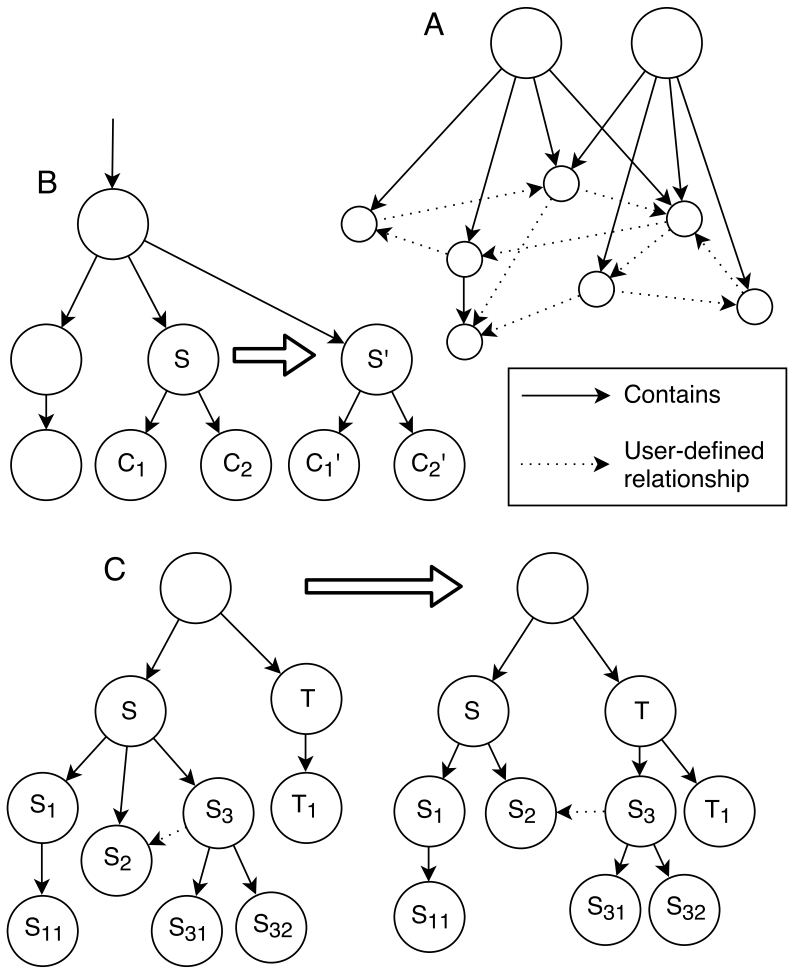


Figure 5.2: (A) Systems may have multiple parent systems, and user defined relationships may exist between systems of the same or different scales. (B) A `deepClone` operation on system S recursively copies contained systems, while membership in higher order entities is inherited. (C) The system S performs a `transfer` operation on S_3 to the system T.

To select a function for a system S to perform, the filter vector from the current parent system $I^P$and the set of vote vectors V^{C^S} are both considered. As filters are passed down the tree, they are combined with the element-wise product, enabling a system to set any chosen function's probability of selection to 0 regardless of the received votes and filters of lower level systems. Equally filters may introduce an overall bias to be applied to the received votes and to hierarchically bias

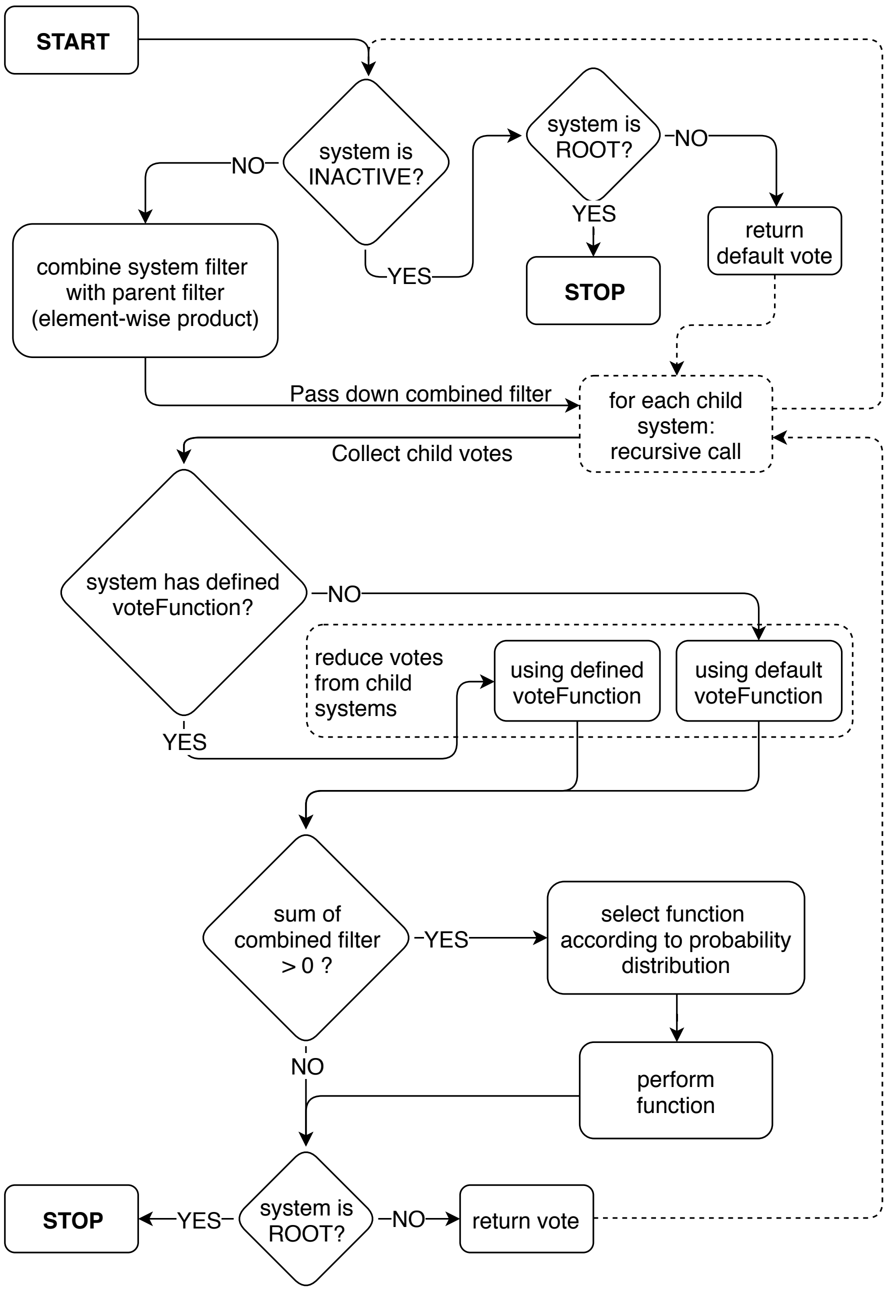

START
system is INACTIVE?
NO
YES
system is ROOT?
NO
YES
STOP
return default vote
combine system filter with parent filter (element-wise product)
Pass down combined filter
for each child system: recursive call
Collect child votes
system has defined voteFunction?
NO
YES
reduce votes from child systems
using defined voteFunction
using default voteFunction
sum of combined filter > 0 ?
YES
NO
select function according to probability distribution
perform function
STOP
YES
system is ROOT?
NO
return vote

Figure 5.3: Recursive compute algorithm executed at each ROOT system once per iteration.

the actions of lower constituent systems. The default behaviour is to reduce V^{C^S} with addition to give

$$V^S = V^{C_1^S} + V^{C_2^S} + \cdots + V^{C_n^S}$$

to then calculate the element-wise product $V^S \odot I^P$. The result is a vector of length $|F|$, which is used as a probability distribution relative to the sum of its elements to select with bias the function $f \in F$ to perform.

While the default behaviour is to combine child systems' votes with addition, and adding the system's own vote before passing the vote up the tree, the votes may be intercepted and reduced differently, or even completely discarded by defining a new vote function for any chosen types of system.

Figure 5.3 provides an outline of the recursive compute function, and demonstrates the order in which filters and votes are combined and how they are passed up and down the tree.

5.3.2 *Implementation*

The IHDN prototype implementation used for demonstration here is written in Java and exposes a simple Application Programming Interface (API) for the development of simulations on top of an embedded Neo4j [199] instance. If votes and filters are not specified for a given system, the defaults of $(0, 0, \dots)$ and $(1, 1, \dots)$ are used respectively, contributing no bias or restriction on function selection. Likewise, any `ROOT` systems are given the default filter of $(1, 1, \dots)$ as their parent filter during computation.

On completion of a simulation, the graph database is stored and can then be queried directly with Cypher [200], or visually using any existing Neo4j compatible tools. The investigation that follows employs the use of the Neo4j multi-platform desktop browser.

5.4 TRACING CELL LINEAGE IN SIMULATED ANEUPLOID TUMOUR GROWTH

To investigate the utility of message passing in leveraging relational structure in the context of biological modelling, we now consider the specific use case of tracing cell lineage in aneuploid tumour growth. This is a suitable use case for our investigation since the behaviour of cells determining their growth is a function of the genes of which they are composed, and there are existing baselines for comparison.

In particular we evaluate our method against an existing biological model of cancerous tumour growth implemented on a conventional computer [38], which was also used to test another unconventional computing platform [184]. The rest of this section describes the simulation according to the methodology given in [37].

The simulation is concerned with role of chromosome missegregation in cancerous tumour growth; and since cancer is progressive via heritable change to cells, the relevance of tracing this change in understanding and hence treating cancer is especially evident. Better understanding of cell lineage affords greater understanding of how a particular cancer will progress, how susceptible it will be to treatment, and the likely-hood of its return [201, 202].

5.4.1 *Biological Observations*

During normal mitotic cell division, each chromosome is duplicated and the resulting set of chromosomes is segregated equally (in a direct one to one correspondence) between the resulting new cells. However, it is estimated for human cells that an average of one in one hundred cell divisions spontaneously missegregates [203], *i.e.* fails to separate the chromosomes into two identical sets, resulting in one cell with extra chromosomes (and hence extra copies of the contained genes), and the other cell with fewer.

As a consequence of this phenomenon and the configuration of which genes are present in the gained and lost chromosomes, cells with different properties and behaviours arise, which can lead to the evolution of cancerous cells that divide highly and do not die naturally.

5.4.2 *The Model*

To explain the way in which aneuploid tumours develop requires four main abstractions of physical systems: the tissue, the cell, the chromosome, and the gene. Contrary to previous implementations, here these abstractions are adopted directly (each as a tree of systems in the hierarchy), affording an intuitive correspondence between the problem domain and the computational model.

5.4.3 *Components*

For this particular simulation, it is necessary to model three particular gene abstractions - the apoptosis regulatory gene (a tumour suppressor gene that regulates cell death), the cell division regulatory gene (an abstraction of proto-oncogenes that promotes cell growth and progression), and the chromosome segregation regulatory gene (an abstraction of genes that control the fidelity of cell division and reduce the likelihood of chromosome missegregation). To capture the sensitivity of the cell behaviours on the initial genetic configurations and gene linkage (the membership of which genes are encoded in which chromosomes), three different chromosome distributions are modelled (Figure 5.4).

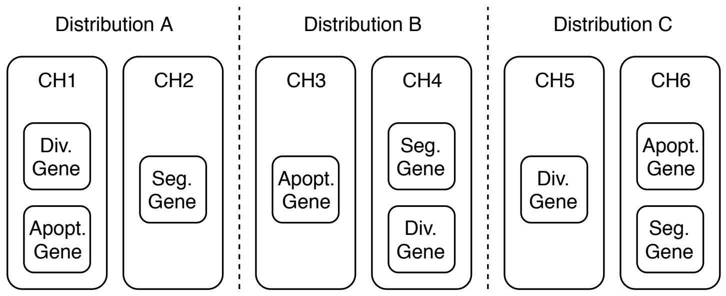


Figure 5.4: Three possible chromosome distributions, formed as combinations of six unique chromosome configurations.

A complete list of the entities modelled in this experiment can be seen in Table 5.1. As will be discussed below, each physical entity is not represented by a single system, but rather the composition of a hierarchy of systems. Thus each physical entity (*i.e.* the tissue, cell, etc.) is actually represented by a tree of systems, with the corresponding system as its root.

To simplify any analytical computation, the possible chromosome configurations are labelled with `CH1` to `CH6` according to the six different possible combinations (Figure 5.4). Additionally, upon completion of the simulation each `CELL` system is given a `genome` property in the form of a vector representing the number of each type of gene that it contains. However, neither the additional labels or property are required during the computation.

To capture the concepts of evolutionary heritage (*i.e.* cell lineage), `FROM` and `WAS` relationships between `CELL` systems are used. These indicate heritage of regular cell division and heritage of division in which missegregation occurs respectively. Other metrics essential in tracing the ancestry of evolved cells are recorded using the cell properties: `start`, indicating the iteration in which a cell first came to exist; `nDivs`, recording the number of times a cell has divided; and `missegrega-`

1 All filters, votes, and vote functions are set to the defaults unless otherwise stated.

IHDN System	Notes
Tissue	Labels: `TISSUE`, `ROOT`
Cell	Labels: `CELL` Properties: `start`, `nDivs` Vote function: `cellVote`
Dead cell	Labels: `CELL`, `INACTIVE` Properties: `start`, `nDivs`, `inactiveAt`
Cell copy	Labels: `CELL_COPY`, `INACTIVE` Properties: `start`, `nDivs`, `missegregationAt`
Chromosome	Labels: `CHROMOSOME`, one of {`CH1, …, CH6`} Filter: $(0, 0, 0)$
Apoptosis gene	Labels: `GENE`, `APOPT_GENE` Vote: $(0, 1, 0)$
Division gene	Labels: `GENE`, `DIV_GENE` Vote: $(1, 0, 0)$
Segregation gene	Labels: `GENE`, `SEG_GENE` Filter: $(0, 0, 0)$

Table 5.1: The set of all IHDN system types used in this simulation.[1]

`tionAt`, indicating the iteration in which chromosome missegregation has occurred.

5.4.4 *Organisation*

Figure 5.5 shows the hierarchy of a tissue system composed of cells with chromosome distribution B. The `TISSUE` system groups the contained cells to provide an entry point for the recursive algorithm. The horizontal (in the context of the hierarchy tree) `FROM` and `WAS` relationships are created between cell systems as shown in Figure 5.7.

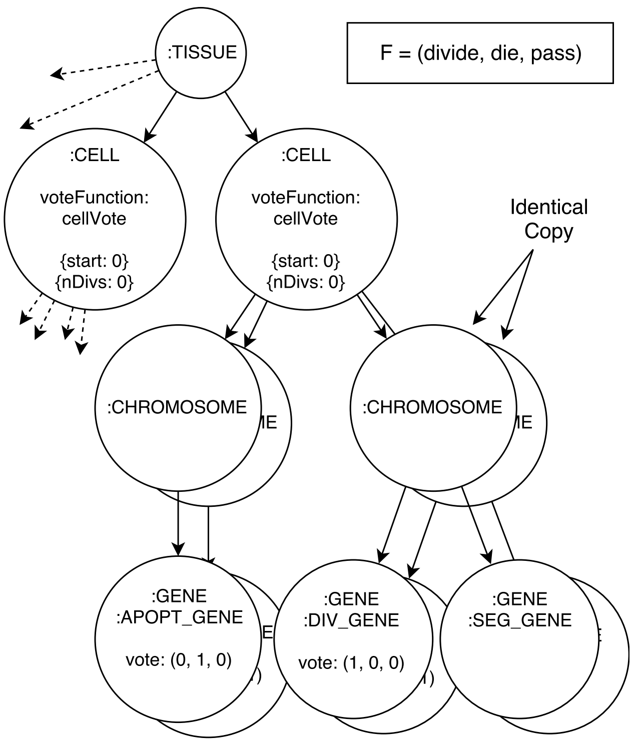


Figure 5.5: IHDN Tissue to Gene Model. Apoptosis and division gene systems influence function selection according to their votes. The presence of segregation genes is queried during cell division in order to calculate the required probability of missegregation.

5.4.5 *Interaction*

While there are three cell behaviours to model in this simulation (cell division, cell death, and chromosome missegregation), since missegregation may only occur in the context of a division, it is not treated as a distinct function, rather it is incorporated into the division function. To enable cells to abstain from any behaviour in a given iteration, a `pass` function is also given resulting in the ordered set

$$F = \{\text{divide}, \text{die}, \text{pass}\}$$

The `divide` function performs a `deepClone` of the current system, such that any contained systems are recursively copied, and any incoming `CONTAINS` relationships are also copied (Figure 5.2). The result is an exact copy of the structural hierarchy and composition, without duplication of any user-defined relationships (*i.e.* the `FROM` and `WAS` relationships). After cloning, assuming no missegregation has occurred, the `start` property of the clone is set to the current iteration and the `nDivs` property in each system incremented.

For the `die` function, a `inactiveAt` property is set to the current iteration and the system is labelled `INACTIVE` to exclude it from further computation.

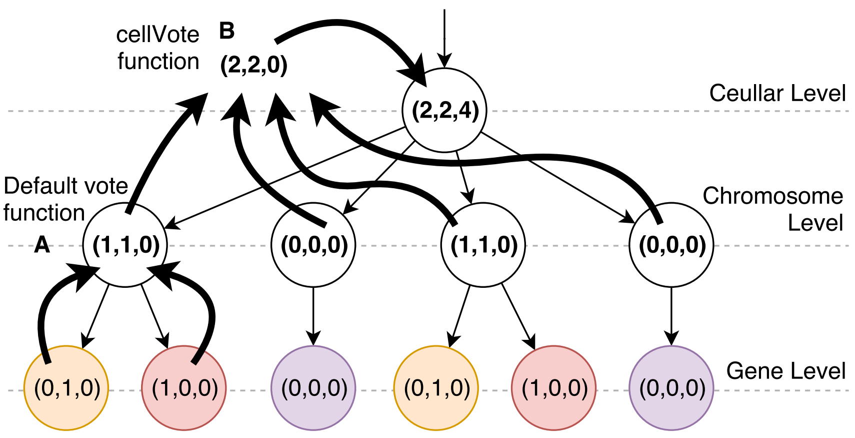


Figure 5.6: Demonstration of message passing system from lower to higher scale systems in a diploid cell of Distribution A: (A) the votes of the gene systems are summed with the chromosome system's own vote resulting in $(1, 1, 0)$; (B) the `cellVote` function intercepts the votes from the chromosomes, reduces them with the default vote function, but then assigns the total of all elements to the position corresponding to the `pass` function.

The `cellVote` vote function ensures that the probability of selecting the `pass` function is always 0.5, independent of the number of contained genes or their configuration (Figure 5.6). The remaining 0.5 is then shared (as per the default behaviour) between the `divide` and `die` functions proportionally to the number of votes for each, resulting in the following probability distribution for cell function selection:

$$\begin{aligned} P(f = \text{pass}) &= \frac{1}{2} \\ P(f = \text{divide} \mid f \neq \text{pass}) &= \frac{d}{a+d} \\ P(f = \text{die} \mid f \neq \text{pass}) &= \frac{a}{a+d} \end{aligned}$$

where a and d are the number of contained `ADOPT_GENE` and `DIV_GENE` systems respectively. We deviate from the reference model [38] under advice from the authors, to provide conditional independence for the `divide` and `die` functions given that `pass` has not been selected. In the original reference model, cells are only considered for division after they have been considered for death, thus the cell division function is selected according to a probability distribution which is also conditioned on cell death. This change will decrease the required number of iterations (which is arbitrary in the original reference model) to observe any emergent effects, while maintaining the key properties necessary to compare the different starting chromosome configurations, *i.e.* the purpose of this modelling technique is to investigate the difference in outcomes between the different configurations such as which configuration leads to exponential growth the fastest, and the evolutionary paths of the cells that lead to such growths, rather than the exact number of iterations taken.

Since presence of the abstracted segregation gene does not bias function selection, but rather influences the extent to which the chromosomes are correctly segregated during cell division, the number of contained `SEG_GENE` systems is queried via a Cypher call during the `divide` function directly without need for voting or an additional function. As with [38], the conditional probability of cell missegregation used is:

$$P(\text{missegregation}|f = \text{divide}) = \frac{1}{100} \times (4 - s)$$

where s is the number of contained SEG_GENE systems. Note, s may only change ± 1 per division, and only when $P(\text{missegregation}) > 0$, it is therefore guaranteed that $0 \le s \le 4$.

In the case that a cell division is subject to chromosome missegregation, a copy of the configuration prior to division is made. The resulting system's CELL label is replaced with CELL_COPY and INACTIVE to prevent its inclusion in any further computation. The resulting pair of aneuploid cells are each linked to it with a WAS relationship (see Figure 5.7). The iteration in which the missegregation occurred is recorded with the missegregationAt property on the CELL_COPY system.

The TISSUE system is the single ROOT system, thus every contained system is visited for computation once per iteration.

5.5 EXPERIMENT

We start by verifying that our novel message passing based form of computation is able to effectively capture the relational information present in the hierarchical graph, and that the expected dynamic behaviours emerge. To do this, we compare our tumour simulation against the results of [38] (discussed below), considering cell count against time, as well as genome diversity, and the ratio of apoptosis to division genes for each of the three chromosome distributions. Then, as an investigation into the additional capabilities of our model afforded by a message passing system that directly maps the problem domain, we analyse the evolutionary heritage (*i.e.* the cell lineage) of the most prolific aneuploid tumour cells that arise during the simulations.

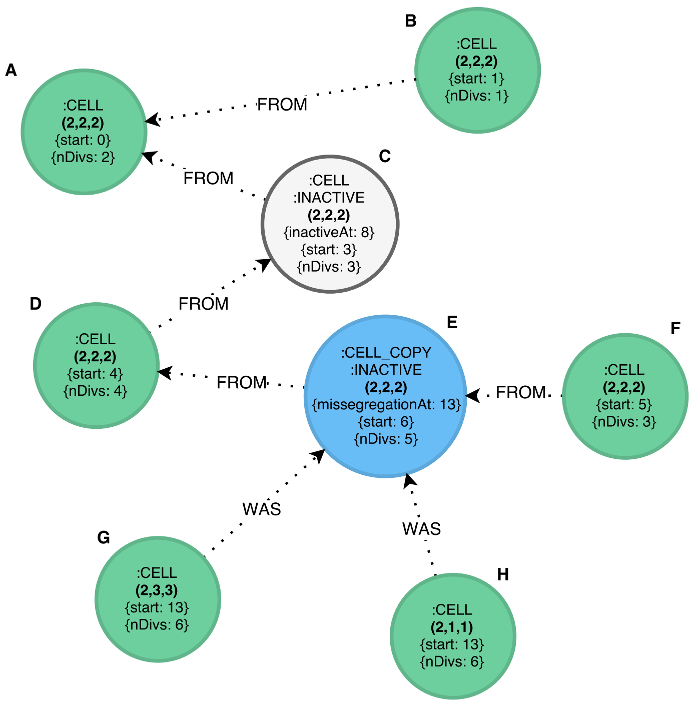


Figure 5.7: Example simulation output graph demonstrating connections between systems on the cellular level of abstraction, where $(\mathbf{a}, \mathbf{d}, \mathbf{s})$ represents the number of contained apoptosis, division, and segregation genes respectively. This particular graph implies the sequence of events: A divides producing B, A divides producing C, C divides producing D, D divides producing E, C dies, E divides producing F, E divides but missegregates resulting in G and H.

The simulation is started with 100 identical diploid cells, and executed 20 times for each of the three possible chromosome configurations (Figure 5.4). The simulation is executed until the tissue exceeds 7,000 living cells (cell count is monitored at the end of each complete iteration), or 100 iterations are reached (whichever occurs first).

5.5.1 *Reference Model*

Graphs of reference model simulation included in Figures 5.8, 5.9 and 5.10 are reproduced here with permission and original data from the author as used in [204] and [38].

The original simulation data recorded the total number of each cell configuration at each time step for 100 steps. However, it does not contain any information regarding the lineage of these cells, or any information regarding which exact cells died at any given time step.

Data and original C++ code are available upon request from Arturo Araujo at arturo@cancerevo.org.

5.6 RESULTS

5.6.1 *Verification*

The simulation results (Figure 5.8 to 5.10) of the IHDN model show the expected growth behaviours as demonstrated in the reference model [38]. When the apoptosis and division genes are distributed in the same chromosome (Distribution A), we see an expected homoeostasis in the size of the tissue. However, when the cells are able to evolve the number of contained copies of the apoptosis and division genes independently, we see the tissue grow in size exponentially. This behaviour is due to the evolution of cells that are 'fitter' (*i.e.* more prolific and less likely to die) than the initial population of diploid cells. Thus, we see the growth of a tumour.

While Distributions B and C both demonstrate exponential growth, it is observed in the reference model that the rate of growth is faster in C [204]. By comparing the mean number of iterations until the tissue size exceeds $7,000$ cells (dashed vertical line in Figure 5.8), we observe the same result. As discussed in Section 5.4.5, due to the difference in the probability distributions for the selection of the copy function, the iteration numbers do not correspond, with our implementation requiring less iterations as expected; however, the purpose here is to validate the life-like evolutionary behaviour of the tissue and cells, and to demonstrate the relative differences in the time taken to reach the exponential inflection point in tumour growth as a consequence of the

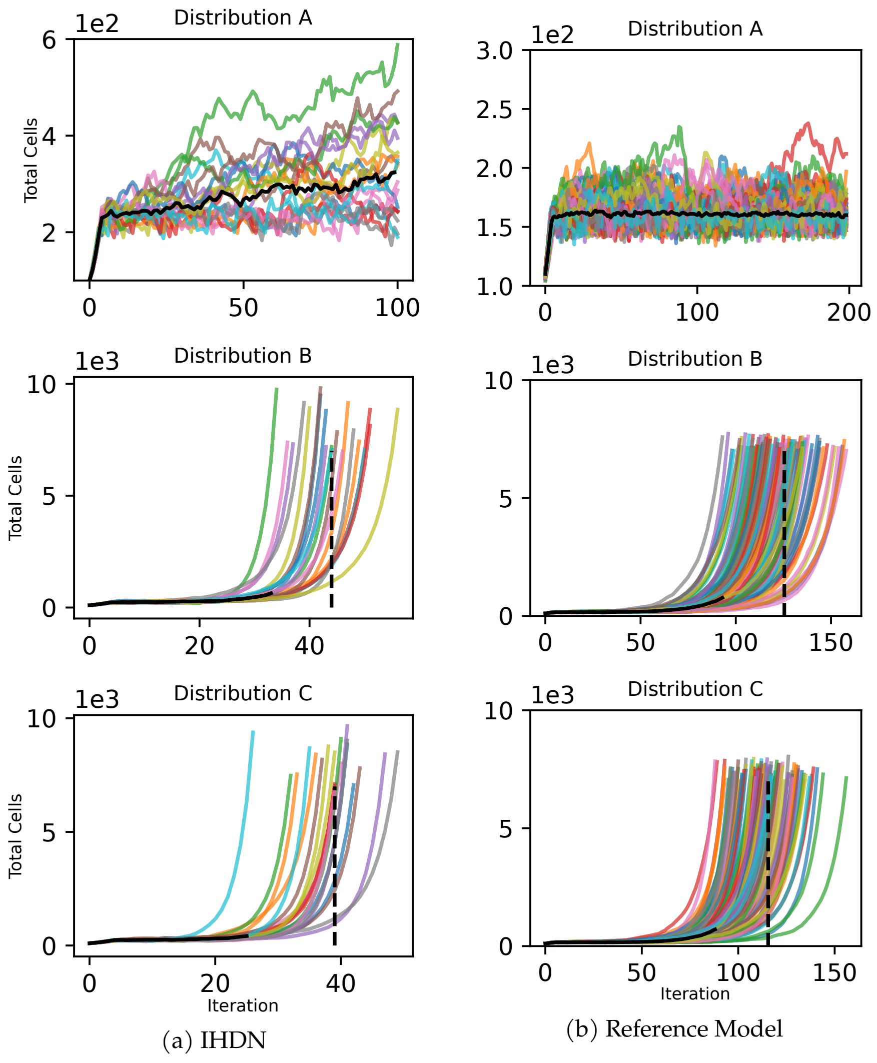
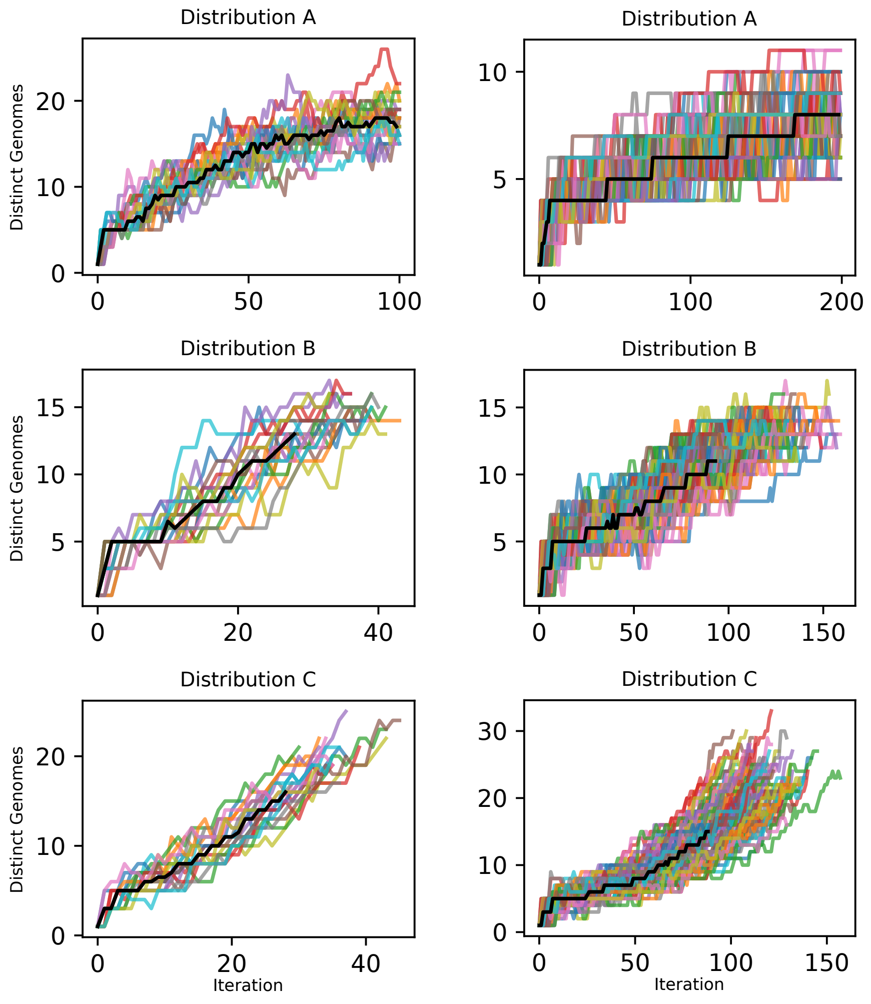


Figure 5.8: Tissue Size – Total number of living cells per iteration (left: IHDN, right: Reference model). Dashed line indicates median iterations to tissue size $> 7,000$ cells.

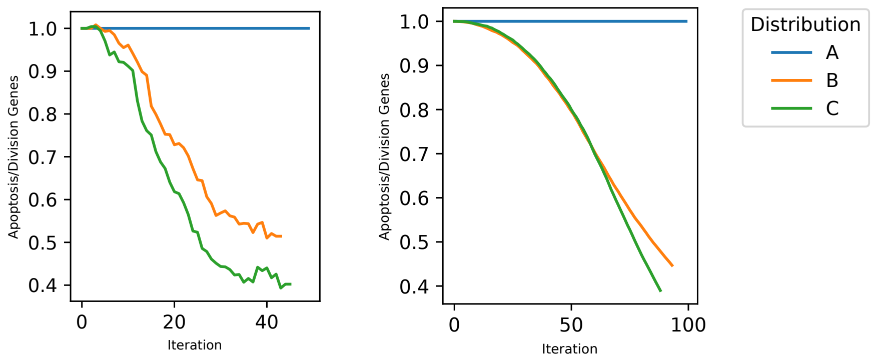
starting chromosome configurations, and not the precise figures of the reference model.

Having verified our model of computation against the reference model, we have demonstrated that message passing is in fact able to effectively model dynamic biological systems. In each of the initial chromosome configurations, the difference in the starting state of each simulation lies purely in the arrangement of the hierarchical relations — each cell contains exactly two of each gene, however their arrangement within the chromosome pairs differs, *i.e.* the chromosome to gene



(a) IHDN (b) Reference Model

Figure 5.9: Chromosome Diversity – Number of distinct genome types per iteration (left: IHDN, right: Reference model).



(a) IHDN (b) Reference Model

Figure 5.10: Mean ratio of apoptosis to division genes (left: IHDN, right: Reference model).

`CONTAINS` relationships. Therefore, it is solely the difference in relational structure that is responsible for the change in behaviours (exponential cell growth vs. homeostasis), with this relational structure captured through the voting and filtering messages sent between connected components (see Figure 5.11).

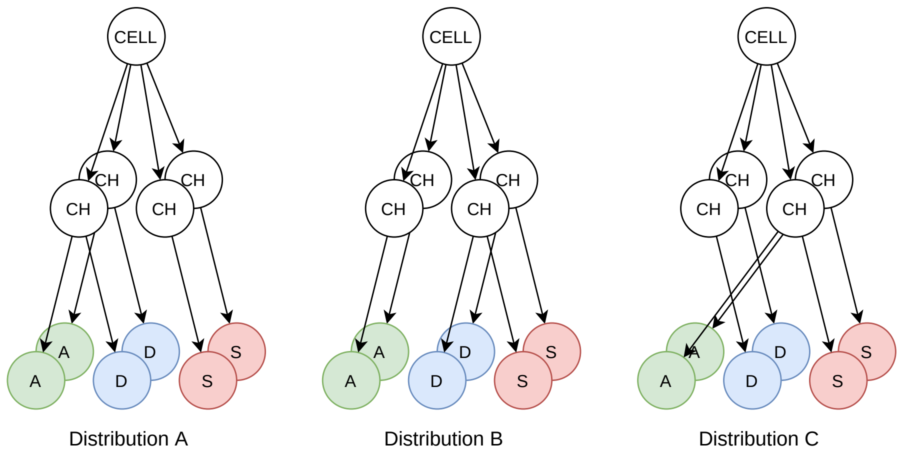


Figure 5.11: Each cell configuration contains exactly the same set of genes, differing only in their relational composition. CH = Chromosome, A = Apoptosis gene, D = Division gene, S = Segregation gene.

5.6.2 *Cell Lineage*

Having verified the behaviour of our model against an existing conventional implementation and confirmed that message passing is a useful tool in modelling dynamic biological systems, we now demonstrate its advantages over conventional approaches. Throughout the remainder of this section, (a, d, s) denotes the number of contained apoptosis, division, and segregation regulatory genes respectively.

A key advantage to our message passing approach over the reference model is that at any stage during the computation, not only does the model have a direct correspondence back to the original problem domain, but since we are operating directly on non-Euclidean data, it is trivial to store additional non-Euclidean relations and information alongside the model as it unfolds, significantly simplifying the process of post-analysis and tracking the history of changes.

As an example of this, using graph matching algorithms (in our case Neo4j Cypher queries) to search the graphs for particular patterns, the complete evolutionary paths of any cell can be traced. Figure 5.1 demonstrates an example query (visual result in Figure 5.12) to show three evolutionary paths from the start cell configuration, $(2, 2, 2)$, to the highest occurring cell configuration (a particularly harmful cell) of Distribution C, $(0, 2, 0)$.

```
match (c:CELL) where (not (c:INACTIVE)) and c.genome=[0,2,0]
with c match p = (c)-[:FROM|WAS*]->(o)
  where
    ((o:CELL) or (o:CELL_COPY))
    and not (o)-[:FROM|WAS]->()
    and not (c)<-[:FROM|WAS]-()
return p limit 3
```

Listing 5.1: Example Cypher query to return three distinct paths of genome evolution from the initial $(2, 2, 2)$ to the cancerous $(0, 2, 0)$.

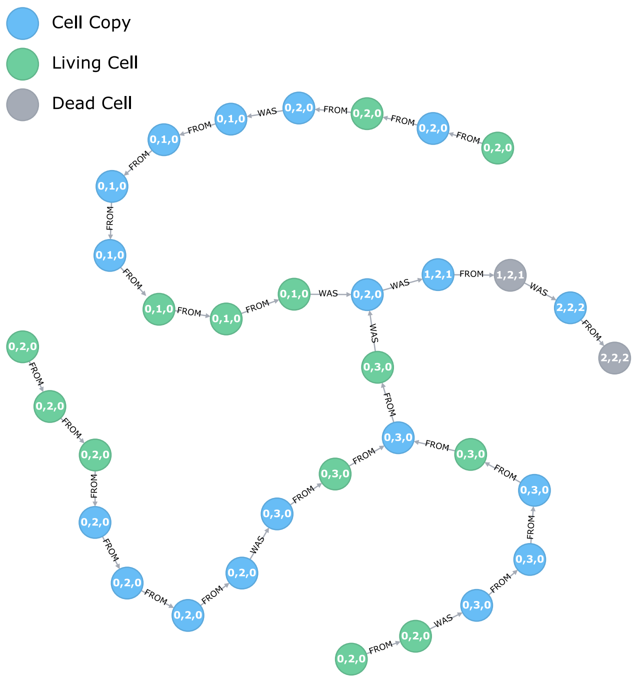


Figure 5.12: Visual result of the example query (Figure 5.1). (Graphic produced by the Neo4j Browser).

Going further, for each of the configurations we query all distinct genome evolutionary paths to each of the arising cell configurations, where consecutive matching genomes are removed from the returned sequences. We see that approximately two thirds of the $(0, 2, 0)$ cells of Distribution C followed the simplest possible route (Table 5.2); because of the higher probability of chromosome missegregation in these cells, many demonstrate increased exploration and oscillation between configurations in their genome ancestry. However, for the most prolific arising cell configuration of Distribution B, $(0, 2, 2)$, it can be seen that a much greater proportion took the shortest evolutionary path, as the probability for missegregation, and hence evolution, in these cells is much lower.

Distribution B		**Distribution C**	
Path	%	Path	%
(2,2,2),(1,2,2),(0,2,2)	86.3	(2,2,2),(1,2,1),(0,2,0)	67.3
(2,2,2),(1,2,2),(1,1,1),(0,1,1),(0,2,2)	3.81	(2,2,2),(1,2,1),(0,2,0),(0,1,0),(0,2,0)	7.47
(2,2,2),(1,2,2),(0,2,2),(0,1,1),(0,2,2)	3.54	(2,2,2),(1,2,1),(0,2,0),(0,3,0),(0,2,0)	7.32

Table 5.2: Proportion of most abundant final cell configuration that followed the most commonly occurring distinct evolutionary paths.

By considering the mean number of distinct paths, we also see that ancestry of the arising $(0, 2, 0)$ genome configurations (Distribution C) is the most diverse (12.45) across all distributions, followed closely by $(0, 3, 0)$ with 12.05. While these forms of analysis require no additional tooling with IHDNs, they were not possible in previous implementations (without $O(|E| \ln |V|)$ additional work — see below) and could help inform the open debate (for example, [205] and [206] are two opposing views) over the evolution of such cells.

5.6.3 *Comparison*

A significant difference between our proposed approach and the reference model is that with IHDNs, the model is non-Euclidean and directly maps the problem domain. Thus post processing involving non-Euclidean results is simplified by keeping the data in a non-Euclidean form.

For example, as can be seen in Figure 5.12 of the cell lineage experiment, the number of times any cell can divide is not fixed, thus storing and processing this type of cell ancestry query over a columnar representation would require complex join operations or look-up queries to reconstruct the non-Euclidean relations present in this data.

Alternatively, assuming the simplest possible conventional approach of logging to file every time a cell divides to record which cell divided and which cells were created, we can see that the computational complexity of reconstructing the relations in a non-Euclidean form in order to perform shortest path queries as above is of $O(|E| \ln |V|)$ time complexity, where $|E|$ is the number of edges (in this case representing cell ancestry), and $|V|$ is the number of vertices (in this case cells). The factor $|E|$ is required in order to read every edge in the log, and the factor $\ln |V|$ since for each edge we must check the existence of the tail vertex. This $O(|E| \ln |V|)$ step is of course not required at all in our proposed approach, as the new relations may be stored at the time of the creation of the new vertices when all necessary references are already held, therefore we can consider the complexity of our method as $O(1)$ for this stage.

Moreover, the pattern matching techniques we have demonstrated in this use case are not restricted to post-analysis; any system functions may take full advantage of the optimised pattern matching Neo4j Cypher query engine during execution, thus enabling systems to interact or adapt their behaviour according to the detection of complex network structures.

As an example of this, we consider an extension of this model in which we incorporate the effects of particular treatments on the cells during tumour growth. Thanks to the non-Euclidean structure of our modelling technique, cells may be configured with physical location coordinates, with additional relations indicating nearest neighbours. Then it is trivial to update cells behaviours according to the states of nearby cells, as well as model treatments such as surgery and/or radiation treatments while being able to determine the distance between cells and the incision/radiation points efficiently.

5.7 SUMMARY

We have introduced the IHDN model for simulating complex dynamic systems, and verified the effectiveness of its novel, cross-scale message passing system in capturing the dynamic hierarchical dependencies of living systems. Through demonstrating the difference in behaviours (exponential growth vs. homeostasis) as a consequence of hierarchical structure alone, we have proven that message passing is able to leverage the relational structure in order to dictate the expected resulting behaviours. Moreover, by leveraging the relational structure present in the original problem domain in order to simulate the dynamic emergent behaviours, our method provides a direct correspondence from the program outputs to the original problem space, thus aiding in post-analysis, while also enabling the tracking of additional non-Euclidean relations such as cell ancestry during computation.

Having demonstrated its application in simulating aneuploid tumour development we observe the expected growth behaviours for all three chromosome distributions. We have also shown that through integration with a graph database the IHDN model facilitates powerful 'out of the box' analysis not possible in prior models, demonstrated here through tracing the evolutionary paths of arising cell configurations.

6 CONCLUSION

6.1 AIM

As presented in Chapter 1, the aim of this work is

> to develop novel message passing techniques that leverage relational structure in order to improve on existing state of the art methods for modelling non-Euclidean data.

Based on our review of existing solutions to non-Euclidean modelling provided in Chapter 2, with the majority of existing message passing methods having targeted single domain vertex level tasks, we identified the motivation for three diverse applications beyond this. First, in Chapter 3 we considered the multiple domain graph level task of graph classification for predicting chemical properties, with the vertex level task of interpreting the attention coefficients. Second, in Chapter 4 we considered multiple domain graph and vertex level tasks for style similarity in B-Reps, and finally, in Chapter 5 we considered single, dynamic domain graph level tasks in modelling cell lineage in cancer tumour growth.

Having investigated each of these application areas, we now relate the contributions made in each chapter back to our central aim.

6.1.1 *PiNet*

With PiNet (Chapter 3), we introduced a new message passing based global pooling operator, which unlike popular existing methods such as global mean/max pooling takes into account the structure of the inputs. The novelty of our approach lies in the use of an auxiliary MPNN which leverages the relational structure of each input graph in order to learn vertex attention coefficients which weight the importance of each vertex in the final graph representation.

We demonstrated superior performance over existing state of the art graph classifiers on a graph isomorphism test with a synthetic dataset, which was designed to be especially difficult by fixing node degree distributions and providing only a small number of permuted training examples.

Having benchmarked our proposed method on a range of widely used chemo-informatic datasets, we observed competitive performance with a range of state of the art graph classifiers. However, unlike the state of the art methods compared against, due to the relational information captured through our message passing pooling operator, we were also able to demonstrate the utility of the learned attention coefficients in interpreting the predictions made by the learned models.

In addition to the contributions detailed above which directly support our central aim, during the process of this investigation we also recognise the following contributions: The creation of a challenging graph isomorphism dataset that may be used for testing future message passing methods as well as non-message passing methods, and the analysis of a set of aromatic amines from the widely used MUTAG dataset including domain knowledge of steric hindrance nitrogen groups with respect to predicting mutagenicity.

6.1.2 *UVStyle-Net*

With UVStyle-Net (Chapter 4), we introduced a new technique for unsupervised style metric learning for B-Reps as well as a few-shot process for learning to cater to an end user's subjectivity. As a novel contribution to message passing techniques we identified the benefit of using the additional relational information present in B-Reps over meshes and point clouds in order to learn style representations. We leveraged this additional relational information in two ways: First, by using the UVNet encoder [127] we used the face boundary information to perform 2D convolutions on points sampled from the surface parameter domains to represent each face, followed by an MPNN which propogates face information across a face-adjacency graph in order to learn higher order representations of the composition of these faces. Second, we used the face boundary information to re-center the sampled points of each face, to provide some invariances to the sampled points prior to entering the MPNN.

Alongside our investigation of style metric learning for B-Reps, we also considered a similar approach for working with mesh and point cloud representations. In doing so we contributed to further understanding of the inductive biases of each approach. For example, we showed that for UVStyle-Net (with face boundary information), large flat surfaces contribute less to the final representation while more complex surfaces formed of several smaller boundaries contribute more, whereas for PointNet++ [180] and PSNet* [174] on point clouds large flat surfaces dominate the representation despite their relative insignificance compared to other details in the style of the object.

With no existing unsupervised style metric learning methods for B-Reps, our approach is especially novel, thus we compared our proposed method against the closest available state of the art method on point clouds (PSNet* [174]) as well as against a similar method to ours applied to point clouds and mesh using the Pointnet++ [180] and MeshCNN [161] encoders respectively. We demonstrated improvement over these baselines quantitatively in terms of the amount of style information present in the 2nd order statistics of each layer of the encoders on SolidMNIST [127], as well as in terms of classifying several subsets of the public industry-used ABC dataset [89]. We also demonstrated the superior performance of our method qualitatively using a range of nearest neighbour queries on both datasets.

With our investigation in Chapter 4, we also recognise several contributions beyond our main aim. First, we demonstrate that the second order statistics (Gram matrix) approach used in 2D image style literature can be generalized to B-Rep and mesh 3D shapes. Second, we introduce a general few-shot learning method for capturing a subjective end-user's definition of 3D style and demonstrate its effectiveness on B-Reps. Finally, with respect to useful datasets for style, we create and share a new version of the SolidMNIST [127] dataset in which we strengthen the associated style labels in order to aid the design and evaluation of 3D style methods, and we share a subset of the ABC dataset [89] which we manually label for style in order to compare 3D style metrics quantitatively on real-world data.

6.1.3 *IHDNs*

With IHDNs (Chapter 5), we proposed a novel message-passing based model of computation able to simulate biological systems. Our method improves over state of the art by enabling a direct correspondence between a non-Euclidean problem domain and the model domain by representing related entities as a graph over which vote and filter messages are propagated in order to prescribe the behaviours of the modelled systems.

Having simulated three different cell configurations in a tumour growth model, we demonstrated that the differences in relational structure are able to effectively produce the correct expected outcomes in terms of the tissue growth, as a direct consequence of the message passing process.

Unlike the reference model [204], by keeping the simulation data in a non-Euclidean form, our method was able to track cell lineage throughout the simulation with $O(1)$ additional work as opposed to $O(|E| \ln |V|)$ for the reference model in which this form of analysis was not directly possible.

Additional contributions made in Chapter 5 include a novel form a computation useful for simulating biological and complex systems, and analysis of the evolutionary paths of the arising aneuploid cell configurations in tumour growth, contributing to the open debate on how such cells arise.

6.2 CRITICAL EVALUATION

6.2.1 *PiNet*

One of the key limitations of our work on PiNet is that the benchmark dataset problems are not sufficiently difficult to effectively distinguish the methods tested, making it hard to draw conclusions about which method(s) is/are better. Also, with many of the datasets including limited vertex features and the use of random train/val/test splits, the datasets are limited in how realistic they are for real-world settings.

The datasets we used were the best available at the time and used as standard in many works [21, 110, 207]. To help us distinguish the methods, when creating the dataset for the isomorphism test, we fixed the degree distribution across each unique generated graph in order to increase the symmetries between the different graph classes and make the task especially difficult. This worked well, in that we saw a much greater range between the best and worst methods in this task, making it much easier to distinguish the stronger and weaker methods.

Since completing this research, addressing the same issue of insufficient challenge for benchmark data, [112] *et al*. proposed Open Graph Benchmark (OGB) as a collection of realistic datasets with consistent data splits. Future work could investigate PiNet with this data.

6.2.2 *UVStyle-Net*

Despite the fact that B-Reps are the standard representation used in industrial and product design, there is very limited existing research on applying neural networks to them [127]. Consequently, aside from having a strong motivation to inventing methods which operate on B-Reps, our contribution is especially novel in addressing this gap.

Working with B-Reps, however, posed additional challenges that we would not have faced for meshes or point clouds. First, there is limited publicly available data, and at the time of starting this research there were no publicly available B-Rep datasets with reliable labels for style. Second, B-Rep data structures are more complex to work with and there is not the ecosystem of libraries for processing them that there is for mesh and point cloud etc.

We tackled the issue of data in three ways. To start, we adapted SolidMNIST [127], a generated dataset of extruded fonts, to better suit style metric evaluation. By removing sources of randomness within font classes in the test set, we strengthened the style labels allowing us to draw more meaningful conclusions from our results. This choice of data was ideal for validating our design decisions, as it was formed of small simple solids which were easy to debug if needed, and the generation process reflects the most common modelling process of a CAD designer — sketching in 2D then extruding.

Next, to enable us to work with real world data and perform quantitative comparisons, we manually labelled three small subsets of the public ABC dataset [89]. This was effective in allowing us to compare quantitatively against existing state of the art, as well as providing a benchmark to enable future research to build upon our contritions. However, to provide the ideal realistic setting, data containing style labels with solids such as car bodies, or other commercial products would be desirable.

Finally, our recognition of the challenges in training models without access to public labelled data is precisely what motivated our unsupervised few-shot approach. End-users such as designers may only have a few examples of their target style. For example, a car manufacturer may have only 10 or so cars in their range, and would therefore be unable to train a deep neural network from scratch in a supervised setting. Yet, with UVStyle-Net pre-training is possible on large publicly available datasets such as ABC, with the few-shot fine-tuning to tailor the style metric to their domain.

For the few-shot user optimization, we started with the simplest possible approach, to learn a weighting of each layer's style embedding through a convex linear combination. Using a convex loss function allowed us to guarantee finding a global minimum, and therefore strengthen our conclusions and ease debugging in the case of unexpected results. While in many cases we observed that all the weight would be applied to a single layer, our approach showed statistically significant improvement over the baseline.

Working without any existing libraries for machine learning with B-Reps was challenging as we had to create pipelines involving different languages, and any visualisation methods for debugging had to be created from scratch. We hope that by sharing our code in the public domain, we will help reduce the barrier to entry for research with

B-Reps, since our work has demonstrated the advantages in computational efficiency and the ability to better capture stylistic detail.

6.2.3 *IHDNs*

Our work on IHDNs posed a different problem setup to the rest of this thesis, in that we were aiming to model a complex phenomenom without labeled training data. To build upon existing work in cancer tumour growth and contribute to key questions around cell lineage we devised a message passing based model of computation akin to agent-based modelling that enables powerful graph-based queries to be performed on the complete simulation history.

Working directly with a graph database was effective in efficiently storing the graphs, as well as enabling powerful queries such as shortest paths with highly optimized and efficient implementations built in to the database.

We demonstrated that IHDN computational method is effective, and the implementation we provided is highly parallelisable. At the time, general message passing frameworks such as [23] (discussed in Section 1.3) were not available, thus we designed the computational process from scratch. Future work could combine the IHDN method with a now standard message passing framework.

6.3 FUTURE WORK

For continuing the research presented in this thesis, we now propose in detail several specific areas for further work. First we consider how PiNet may be adapted to enable hierarchical pooling (graph coarsening), then we consider regularization of our few-shot user defined optimization method, and an extension to the UVStyle-Net architecture which takes into account further relational information. Finally, we propose two general directions for further investigation into our aim beyond the specific applications considered so far.

6.3.1 *PiNet: Hard Masking & Hierarchical Pooling*

The success of conventional 2D CNNs is in a large part due to their hierarchical composition [134], in which multiple pooling layers provide a relational inductive bias allowing long range interactions between local information in the input signal [13]. Consequently, we hypothesise that introducing hierarchical pooling into our PiNet architecture will improve the performance. Thus, we now propose how our soft pooling process could be replaced with hard pooling in order to enable hierarchical composition of pooling layers.

As detailed in Chapter 3, graph level pooling in PiNet is achieved via learned vertex attention coefficients ϕ_A which are then passed through a softmax function to give the weighting of each node in the final representation.

Let $\alpha \in \mathbb{R}^{N \times F'}$ represent these weights,

$$\alpha = \sigma_S([\phi_A(A, X)]^\top) \tag{6.1}$$

where σ_S is the softmax function, $\phi_A : (\mathbb{R}^{N \times N}, \mathbb{R}^{N \times F}) \rightarrow \mathbb{R}^{N \times F'}$ is any message passing network, $A \in \mathbb{R}^{N \times N}$ is the adjacency matrix, and $X \in \mathbb{R}^{N \times F}$ is the corresponding F-dimensional vertex features. Then an alternative approach would be to convert these weights to a mask $M \in \mathbb{R}^{N \times F'}$,

$$M_{ij} = \begin{cases} 1, & \text{if } \alpha_{ij} > t, \\ 0, & \text{otherwise,} \end{cases} \tag{6.2}$$

where $0 \leq t \leq 1$ is a threshold hyperparameter, and to then replace Equation 3.3 with

$$z(G) = \sigma_S \left[g \left(M \cdot \phi_X(A, X) \right) W_D \right] \in \mathbb{R}^C. \tag{6.3}$$

The threshold t could be be tuned via a validation set, or alternatively computed per graph (as a function of N) to allow a fixed proportion of vertices to be kept (similar to the top-k stragety used in [208]).

We hypothesize that by removing vertices with small α and fully including vertices with large α, we could improve performance for classification tasks. Moreover, as discussed above, by using a 'hard' mask, vertices not selected for the mask could be removed from the following layer to enable hierarchical pooling (graph coarsening), whereby the pooling process is repeated several times before finally the global aggregation is applied.

6.3.2 *UVStyle-Net: Feature Distribution Measure*

As identified in Chapter 4 as a limitation of our approach, during few-shot optimization of our user defined loss we observed that often all weight is applied to a single layer (*e.g.* Figure 4.3). We can consider this as a form of over-fitting, therefore, we hypothesize that our style metric could be improved by regularizing the few-shot optimization.

Based on our understanding that the Gram matrices of adjacent layers capture more similar stylistic features than distance layers [148] (*i.e.* there is a smooth transition from low level surface details in the lowest layers to higher level patterns in the highest layers), one possible approach is to constrain the weights to fit a Gaussian bell curve. This will ensure that the weight is shared among adjacent layers with similar amounts.

Recall that in our proposed method the optimal user-defined weights $\mathbf{w}^\star$ are found

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \sum_{l=1}^{L} w_l \cdot E_l \tag{6.4}$$

such that

$$E_l = c_1 \cdot \sum_{\substack{\mathbf{t}_i, \mathbf{t}_j \in T \\ i \neq j}} D_l(\mathbf{t}_i, \mathbf{t}_j) - c_2 \cdot \sum_{(\mathbf{t}, \mathbf{t}') \in T \times T'} D_l(\mathbf{t}, \mathbf{t}') \tag{6.5}$$

where $D_l(\cdot,\cdot)$ is the cosine distance of the normalised activation Gram matrices at layer l, T and T' are the set of positive and negative user selected examples respectively, and c_1 and c_2 are normalisation constants.

Thus, we could instead optimize a single parameter μ

$$\mathbf{w}^\star = \arg\min_{\mu} \sum_{l=1}^{L} w_l \cdot E_l \tag{6.6}$$

such that

$$w_l = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{l-\mu}{\sigma}\right)^2} \tag{6.7}$$

where $\mu \in [0, L]$, L is the number of layers, and $\sigma > 0$ is the amount of regularization.

Based on this reparametrisation of our few-shot loss, setting σ close to 0 would result in all the weight applied to the single layer in which the feature distributions of the user selected positive/negative examples match/differ the most, with increasing σ causing the weight to be shared among the adjacent layers with decreasing amounts.

6.3.3 *UVStyle-Net: Further Relational Structure*

As shown in Figure 4.4, the UV-Net encoder [127] used in our proposed method is comprised of an MPNN over the face-adjacency graph of a B-Rep, where each face is first encoded with a conventional 3 layer 2D CNN over the masked 10 by 10 grid of UV sampled points. This architecture is able to leverage the relational structure of the face-adjacency graph as well as the groupings of each sampled point into the face boundaries. However, by processing each face entirely separately prior to entering the MPNN, this architecture does not capture information regarding the edges between faces, or the relative orientations of adjacent faces, until entering the MPNN at layer 5.

As many of the stylistic details associated with style are best captured in the lowest layers of the network (see Figure 4.7), we hypothesize that incorporating information regarding the edges at an earlier stage would be beneficial to the learned style representations.

Since conducting the research presented in Chapter 4, the authors of [127] have extended the architecture to include edge information by encoding each edge curve with a 1D conventional CNN over 10 points sampled from the UV domain of the edge [209]. These edge representations are then added as edge features to the face-adjacency graph before applying an MPNN as before, but this time considering edge properties as well as vertex features.

While this extension to UV-Net considers edge curve information at earlier layers, it does still not take into account the relative orientations of adjacent faces (*i.e.* the angles between faces), since the curve features only include the absolute *xyz* positions, tangent normals and a trimming mask. Thus, we propose also including the estimated angle between the adjacent faces at each of the sampled points as an additional feature.

To provide fine control over the impact of the relative orientations of faces within our style embeddings, we propose that the angle features should be fed to a separate 1D CNN to the edge curve CNN and concatenated to the edge curve features when passed to the MPNN. Then, the Gram matrices for the faces G_l^F, edge curves G_l^E and edge angles G_l^A at each layer l can be computed (see Figure 6.1) and weighted separately to give the style distance:

$$\mathcal{D}_{style} = \mathcal{D}_{low} + \mathcal{D}_{high} \tag{6.8}$$

with

$$\begin{aligned} \mathcal{D}_{low}(\mathbf{a}, \mathbf{b}) = \sum_{l=1}^{4} \Big(& w_l^F \cdot D(G_l^F(\mathbf{a}), G_l^F(\mathbf{b})) \\ & + w_l^E \cdot D(G_l^E(\mathbf{a}), G_l^E(\mathbf{b})) \\ & + w_l^A \cdot D(G_l^A(\mathbf{a}), G_l^A(\mathbf{b})) \Big) \end{aligned} \tag{6.9}$$

and

$$\mathcal{D}_{high} = \sum_{l=5}^{L} w_{l-4}^M \cdot D(G_l(\mathbf{a}, \mathbf{b})) \tag{6.10}$$

where $D(\cdot, \cdot)$ is the cosine distance, $\mathbf{w}^F, \mathbf{w}^E, \mathbf{w}^A \in \mathbb{R}^4$ are weights vectors for the face, edge and angle Gram matrices respectively, $\mathbf{w}_M \in \mathbb{R}^{L-4}$ is a weights vector for the MPNN layer Grams, $\mathbf{w}^F, \mathbf{w}^E, \mathbf{w}^F, \mathbf{w}^M \succeq 0$, and

$$\sum_l w_l^F + \sum_l w_l^E + \sum_l w_l^A + \sum_l w_l^M = 1. \tag{6.11}$$

We note that the optimization of our user defined loss as described in Section 4.3 with the style distance defined in Equation 6.8 will remain a convex optimization problem, thus can be solved using Sequential Least Squares Quadratic Programming (SLSQP) [178] as before.

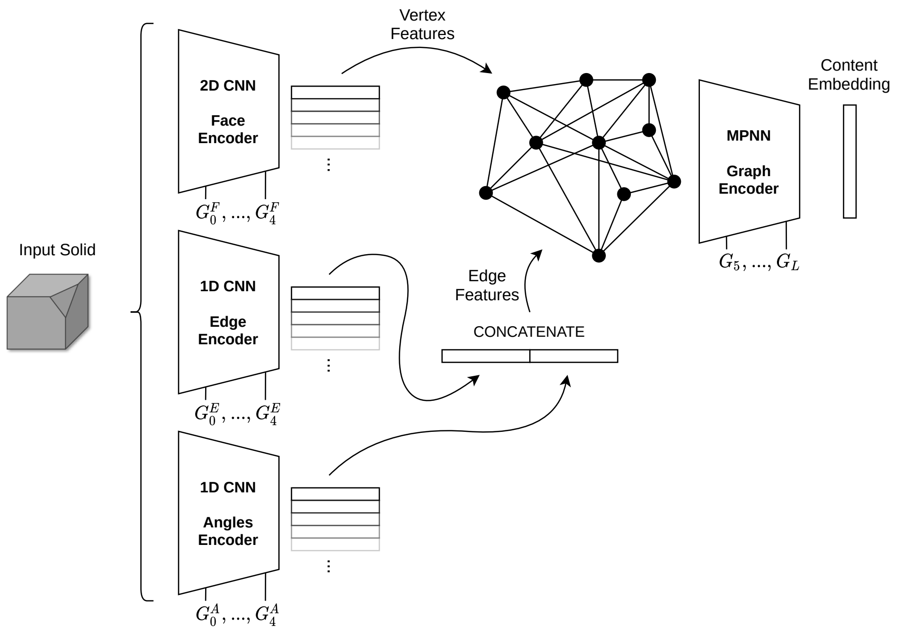


Figure 6.1: Proposed architecture to incorporate further relational information regarding edge curve features and adjacent face angles.

By factoring in the this extra relational structure of how faces are connected, our style measure should be better equipped to match solids with corresponding regions of similarly angled surfaces (*i.e.* cogs and gears, saw teeth etc.).

6.3.4 *Further Directions*

Thus far, we have investigated three diverse applications in order to realise our aim. To extend our work beyond these, we propose investigation into the lesser explored area of discriminative analysis on dynamic domains, *i.e.* time series graphs. Possible applications of this include predicting user behaviours in social networks, predicting levels of face-to-face proximity in dynamic human contact networks, and predicting consumer purchases based on dynamic consumer-product graphs.

With the applications investigated in the body of this thesis, we have shown the advantages that arise when leveraging relational structure over treating the inputs as sets (for example UVStyle-Net vs the set-based PSNet* in Chapter 4). Yet in many widely used methods on graphs, the relational structure of the order in which vertices and edges are created is ignored. For example, with collaborative filtering (a widely-used industry choice for recommender systems [210]), static snapshots of the dynamic consumer-product graphs are taken and processed in order to provide recommended next purchases, thus not taking into account the order in which previous purchases were made.

In recognition of the utility in this temporal relational information, some works have utilized RNNs in order to capture sequential patterns [211, 212]; however, RNNs are very complex models with many parameters, and as such their use is dependant on large amounts of training data. Therefore, it seems a logical step to investigate the use of MPNNs on graphs in which the temporal relations are made explicit, especially given the current success of MPNNs as recommenders due to their ability to capture extra external relational information in the form of knowledge graphs and social relations among users [213].

Another important area for further investigation is that of graph generation, with possible applications in drug design, computer program synthesis, and 3D design. While there have been many recent works addressing the graph generation problem (*i.e.* [70, 71]), at the time of writing there is not yet a definitive approach. As with discriminative graph models, again we hypothesise that modelling temporal relational information explicitly could be valuable and is an important area for investigation.

A CHANGES

Changes as requested in examiner's report:

- Check all the important claims you made (clearly temporality matters). — see pages 23, 79
- Chapter 1, p. 23: generalized aggregation operator in eq 1.1 was previously used by Wang et al. in DGCNN. — see page 23
- Chapter 2: add a discussion on how your WL results compare with the results in Bevilacqua ...Maron 2021 (`https://arxiv.org/pdf/2110.02910.pdf`). — see page 53–54
- Chapter 2: discuss in a more extensive way the expressive power of positional encoding. — see page 39–40, 54
- Chapter 3 discuss how the methodology could be extended to the use of information on geometric coordinates of atoms. Also discuss in a more clear way the expressivity of PiNet with respect to 1-WL/MPNN. — see page 63–64, 65/67, 69, 79
- Chapter 4: update info on accepted publications and discuss reviewers comments. — see page 6, 59, 80, 115
- Chapter 5: describe in a more extensive way the data used and add further relevant citations. — see page 117, 132–133

Description of changes by page number:

- p6: update list of publications based on publication of UVStyle-Net
- p23: credit Wang et al for generalization of aggregation operator
- p39–40: discuss positional encodings more extensively
- p44: clarify claim based on the results of Bevilacqua et al proposing a method for using 1-WL methods with more power than 1-WL test
- p53–54: discussion of Bevilacqua et al result with respect to WL test
- p54: further discussion of how positional information can increase expressivity beyond 1-WL test
- p59: discuss peer review feedback for PiNet
- p63–64: discuss use of geometric features in PiNet for molcules
- p65/67: discuss isomorphism test graphs distinguishability with 1-WL test
- p69: discussion of PiNet with relation to 1-WL test and provide possible interpretation of the difference between its performance compared to the other methods tested
- p79: clarify/reduce the claim made that PiNet is more expressive than the methods tested
- p80: discuss peer review feedback on UVStyle-Net
- p115: discuss peer review feedback on IHDNs
- p117 (and others): update reference to published paper (Araujo et al) rather than thesis
- p132–133: discuss the data from the reference model and how to obtain it

REFERENCES

[1] Peter Meltzer, Hooman Shayani, Amir Khasahmadi, Pradeep Kumar Jayaraman, Aditya Sanghi, and Joseph Lambourne. "UVStyle-Net: Unsupervised Few-Shot Learning of 3D Style Similarity Measure for B-Reps." In: *International Conference on Computer Vision (ICCV)* (2021).

[2] Joseph Lambourne, Karl D. D. Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. "BRep-Net: A Topological Message Passing System for Solid Models." In: *Computer Vision and Pattern Recognition (CVPR)* (2021).

[3] Peter Meltzer, Marcelo Daniel Gutierrez Mallea, and Peter J. Bentley. "PiNet: Attention Pooling for Graph Classification." In: *Neural Information Processing Systems (NeurIPS): Graph Representation Learning Workshop*. 2019.

[4] Peter Meltzer and Peter J. Bentley. "Interacting Hierarchical Dynamic Networks." In: *The 2018 Conference on Artificial Life: A Hybrid of the European Conference on Artificial Life (ECAL) and the International Conference on the Synthesis and Simulation of Living Systems (ALIFE)* (2018), pp. 582–589. DOI: 10.1162/isal_a_00108.

[5] Peter Meltzer, M.D.G. Marcelo Daniel Gutierrez Mallea, and Peter J. Bentley. "PiNet: A Permutation Invariant Graph Neural Network for Graph Classification." In: *arXiv* (2019).

[6] Marcelo Daniel Gutierrez Mallea, Peter Meltzer, and Peter J. Bentley. "Capsule Neural Networks for Graph Classification Using Explicit Tensorial Graph Representations." In: *arXiv* (2019).

[7] Robin Milner, Joachim Parrow, and David Walker. "A Calculus of Mobile Processes, I." In: *Information and Computation* 100.1 (Sept. 1992), pp. 1–40. DOI: 10.1016/0890-5401(92)90008-4.

[8] A H Taub. "The General and Logical Theory of Automata." In: *Design of Computers, Theory of Automata and Numerical Analysis* V (1951), pp. 1–41.

[9] Judea Pearl. "Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach." In: *Proceedings of the National Conference on Artificial Intelligence*. 1982, pp. 133–136. ISBN: 0-86576-043-8.

[10] Jin H Kim and Judea Pearl. "A Computational Model for Causal and Diagnostic Reasoning in Inference Systems." In: vol. 1. 1983, pp. 190–193. ISBN: 0-86576-064-0.

[11] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. "The PageRank Citation Ranking: Bringing Order to the Web." In: *World Wide Web Internet And Web Information Systems* 54.1999-66 (1998), pp. 1–17. ISSN: 9781424433803.

[12] William L Hamilton, Rex Ying, and Jure Leskovec. "Representation Learning on Graphs: Methods and Applications." In: *arXiv* (2017).

[13] Peter W Battaglia et al. "Relational Inductive Biases, Deep Learning, and Graph Networks." In: *arXiv* (2018).

[14] Benjamin R Mitchell. "The Spatial Inductive Bias of Deep Learning." PhD thesis. 2017.

[15] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. "Graph Neural Networks: A Review of Methods and Applications." In: *arXiv* (2018), pp. 1–20.

[16] Ziwei Zhang, Peng Cui, and Wenwu Zhu. *Deep Learning on Graphs: A Survey*. 2018. DOI: 10.1109/tkde.2020.2981333.

[17] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks." In: *International Conference on Learning Representations (ICLR)*. Sept. 2016, pp. 1–14. ISBN: 978-1-61197-068-5. DOI: 10.1051/0004-6361/201527329.

[18] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. "A Comprehensive Survey on Graph Neural Networks." In: *arXiv* (2019). DOI: 10.1109/tnnls.2020.2978386.

[19] Uwe Schöning. "Graph Isomorphism Is in the Low Hierarchy." In: *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 247 LNCS. 1987, pp. 114–124. ISBN: 978-3-540-17219-2. DOI: 10.1007/BFb0039599.

[20] Stephen A. Cook. "The Complexity of Theorem-Proving Procedures." In: *Proceedings of the Third Annual ACM Symposium on Theory of Computing*. STOC '71. New York, NY, USA: ACM Press, 1971, pp. 151–158. DOI: 10.1145/800157.805047.

[21] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. "Learning Convolutional Neural Networks for Graphs." In: 1 (2016). ISSN: 9781510829008.

[22] Alberto Sanfeliu and King-Sun Fu. "A Distance Measure between Attributed Relational Graphs for Pattern Recognition." In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13.3 (May 1983), pp. 353–362. DOI: 10.1109/TSMC.1983.6313167.

[23] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. "Neural Message Passing for Quantum Chemistry." In: *International Conference on Machine Learning (ICML)*. 2017. ISBN: 978-1-5108-5514-4. DOI: 10.1021/acs.jmedchem.7b01484.

[24] Yue Wang, Michael M Bronstein, Justin M Solomon, Yongbin Sun, Ziwei Liu, and Sanjay E Sarma. "Dynamic Graph CNN for Learning on Point Clouds." In: *ACM Transactions on Graphics* 38.5 (Nov. 2019), pp. 1–12. DOI: 10.1145/3326362.

[25] Arnold Neumaier. "Mathematical Model Building." In: *Modeling Languages in Mathematical Optimization*. Springer, 2004, pp. 37–43.

[26] Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. "Gated Graph Sequence Neural Networks." In: *International Conference on Learning Representations (ICLR)*. 2016. DOI: 10.1103/PhysRevLett.116.082003.

[27] William L Hamilton, Rex Ying, Jure Leskovec, Zhitao Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs." In: *Neural Information Processing Systems (NeurIPS)*. Ed. by I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett. Curran Associates, Inc., 2017, pp. 1024–1034.

[28] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. "CayleyNets: Graph Convolutional Neural Networks with Complex Rational Spectral Filters." In: (2017). ISSN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.576.

[29] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. "Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks." In: (2018).

[30] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lì, and Yoshua Bengio. "Graph Attention Networks." In: *International Conference on Learning Representations (ICLR)*. 2015. 2018, pp. 1–11. ISBN: 1710.10903v3. DOI: 10.4271/821240.

[31] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-Ichi Ichi Kawarabayashi, and Stefanie Jegelka. "Representation Learning on Graphs with Jumping Knowledge Networks." In: *International Conference on Machine Learning (ICML)*. Vol. 12. 2018, pp. 8676–8685. ISBN: 978-1-5108-6796-3.

[32] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. "Principal Neighbourhood Aggregation for Graph Nets." In: *Neural Information Processing Systems (NeurIPS)* (2020).

[33] Rémy Brossard, Oriel Frigo, and David Dehaene. "Graph Convolutions That Can Finally Model Local Structure." In: (2020).

[34] Dominique Beaini, Saro Passaro, Vincent Létourneau, William L Hamilton, Gabriele Corso, and Pietro Liò. “Directional Graph Networks.” In: *arXiv* (Oct. 2020).

[35] Andreas Loukas. “What Graph Neural Networks Cannot Learn: Depth vs Width.” In: *arXiv* (2019).

[36] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. “Convolutional Networks on Graphs for Learning Molecular Fingerprints.” In: *Advances in Neural Information Processing Systems 28*. Ed. by C Cortes, N D Lawrence, D D Lee, M Sugiyama, and R Garnett. Curran Associates, Inc., 2015, pp. 2224–2232.

[37] P. J. Bentley. “Methods for Improving Simulations of Biological Systems: Systemic Computation and Fractal Proteins.” In: *Journal of The Royal Society Interface* 6.Suppl_4 (2009), S451–S466. ISSN: 1742-5689. DOI: 10.1098/rsif.2008.0505.focus.

[38] Arturo Araujo, Buzz Baum, and Peter Bentley. “The Role of Chromosome Missegregation in Cancer Development: A Theoretical Approach Using Agent-Based Modelling.” In: *PLoS ONE* 8.8 (2013). DOI: 10.1371/journal.pone.0072206.

[39] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. “Geometric Deep Learning: Going beyond Euclidean Data.” In: *IEEE Signal Processing Magazine* 34.4 (July 2017), pp. 18–42. DOI: 10.1109/MSP.2017.2693418.

[40] Sang Hun Lee, Kunwoo Lee, Sang Hun Lee, and Kunwoo Lee. “Partial Entity Structure: A Compact Non-Manifold Boundary Representation Based on Partial Topological Entities.” In: *Proceedings of the Symposium on Solid Modeling and Applications* 1.4 (2001), pp. 159–170. ISSN: 1581133669. DOI: 10.1115/1.1433486.

[41] Daixin Wang, Yuan Qi, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, and Shuang Yang. “A Semi-Supervised Graph Attentive Network for Financial Fraud Detection.” In: *Proceedings - IEEE International Conference on Data Mining, ICDM*. Vol. 2019-Novem. Institute of Electrical and Electronics Engineers Inc., Nov. 2019, pp. 598–607. ISBN: 978-1-72814-603-4. DOI: 10.1109/ICDM.2019.00070.

[42] Aditya Grover. “Node2vec : Scalable Feature Learning for Networks.” In: (2016), pp. 855–864. ISSN: 9781450342322.

[43] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. “Knowledge Graph Embedding Based Question Answering.” In: *WSDM 2019 - Proceedings of the 12th ACM International Conference on Web Search and Data Mining*. Vol. 19. 2019, pp. 105–113. ISBN: 978-1-4503-5940-5. DOI: 10.1145/3289600.3290956.

[44] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. "Computational Capabilities of Graph Neural Networks." In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 81–102. DOI: 10.1109/TNN.2008.2005141.

[45] Blake Anderson, Daniel Quist, Joshua Neil, Curtis Storlie, and Terran Lane. "Graph-Based Malware Detection Using Dynamic Analysis." In: *Journal in Computer Virology* (2011). DOI: 10.1007/s11416-011-0152-x.

[46] Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael Bronstein. "Learning Shape Correspondence with Anisotropic Convolutional Neural Networks." In: *Advances in Neural Information Processing Systems*. 2016, pp. 3197–3205.

[47] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. "Protein Interface Prediction Using Graph Convolutional Networks." In: *Advances in Neural Information Processing Systems*. Vol. 2017-Decem. 2017, pp. 6531–6540.

[48] Jason Hartford, Devon R. Graham, Kevin Leyton-Brown, and Siamak Ravanbakhsh. "Deep Models of Interactions across Sets." In: *35th International Conference on Machine Learning (ICML)*. 2018. ISBN: 978-1-5108-6796-3.

[49] Belinda A Chiera and Belinda A Chiera. "On the Detection of Hidden Terrorist Cells Immersed in Peer to Peer Networks." In: August (2011), pp. 1–2.

[50] Nino Shervashidze and Karsten M Borgwardt. "Fast Subtree Kernels on Graphs." In: *Neural Information Processing Systems (NeurIPS)*. 2009, pp. 1660–1668. ISBN: 978-1-61567-911-9.

[51] Brian D. Ripley. *Pattern Recognition and Neural Networks*. 2014. ISBN: 978-0-511-81265-1. DOI: 10.1017/CBO9780511812651.

[52] K Sreedhar Reddy. "Enlargement of Image Based Upon Interpolation Techniques." In: 2.12 (2013), p. 10.

[53] Kunihiko Fukushima. "Neocognition: A Self." In: *Biol. Cybernetics*. 1980. ISBN: 0340-1200.

[54] Nadav Cohen and Amnon Shashua. "Inductive Bias of Deep Convolutional Networks through Pooling Geometry." In: *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*. 2017.

[55] Manzil Zaheer, Satwik Kottur, Siamak Ravanbhakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J Smola. "Deep Sets." In: *Neural Information Processing Systems (NeurIPS): Graph Representation Learning Workshop*. Vol. 2017-Decem. 2017, pp. 3392–3402.

[56] Davide Chicco. "Siamese Neural Networks: An Overview." In: *Methods in Molecular Biology*. 2021. DOI: 10.1007/978-1-0716-0826-5_3.

[57] Ryan L Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. "Janossy Pooling: Learning Deep Permutation-Invariant Functions for Variable-Size Inputs." In: *International Conference on Learning Representations (ICLR)*. 2019.

[58] Haggai Maron, Or Litany, Gal Chechik, and Ethan Fetaya. "On Learning Sets of Symmetric Elements." In: (Feb. 2020).

[59] Frank Lin and William W Cohen. "Power Iteration Clustering." In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (2010), pp. 655–662. ISSN: 9781605589077.

[60] A Y Ng, M Jordan, and Y Weiss. "On Spectral Clustering: Analysis and an Algorithm: In Advances in Neural Information Processing Systems 14." In: (2001).

[61] Michel Neuhaus, Kaspar Riesen, and Horst Bunke. "Novel Kernels for Error-Tolerant Graph Classification." In: *Spatial Vision* 22.5 (Sept. 2009), pp. 425–441. ISSN: 0169-1015. DOI: 10.1163/156856809789476119.

[62] John Lafferty and Risi Imre Kondor. "Diffusion Kernels on Graphs and Other Discrete Input Spaces." In: *ICML '02 Proceedings of the Nineteenth International Conference on Machine Learning* (2002), pp. 315–322. ISSN: 1558608737.

[63] Kai Chen and Qiang Huo. *Scalable Training of Deep Learning Machines by Incremental Block Training with Intra-Block Parallel Optimization and Blockwise Model-Update Filtering*. ISBN: 978-1-4799-9988-0.

[64] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. "Spectral Networks and Locally Connected Networks on Graphs." In: *International Conference on Learning Representations (ICLR)*. 2014, pp. 1–14.

[65] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering." In: *Neural Information Processing Systems (NeurIPS)* (2016). ISSN: 978-1-5108-3881-9.

[66] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. "DeepWalk: Online Learning of Social Representations." In: (2014). ISSN: 978-1-4503-2956-9. DOI: 10.1145/2623330.2623732.

[67] Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space." In: *Proceedings of the International Conference on Learning Representations (ICLR 2013)* (2013), pp. 1–12. ISSN: 1532-4435. DOI: 10.1162/153244303322533223.

[68] Leonardo F. R. Ribeiro, Pedro H. P. Saverese, and Daniel R. Figueiredo. "Struc2vec: Learning Node Representations from Structural Identity." In: (2017), pp. 385–394. ISSN: 9781450335423. DOI: 10.1145/3097983.3098061.

[69] John Boaz Lee, Ryan Rossi, and Xiangnan Kong. "Graph Classification Using Structural Attention." In: *KDD* (2018), pp. 1666–1674. ISSN: 9781450355520. DOI: 10.1145/3219819.3219980.

[70] Jiaxuan You, Rex Ying, Xiang Ren, William L Hamilton, and Jure Leskovec. "GraphRNN: Generating Realistic Graphs with Deep Auto-Regressive Models." In: *35th International Conference on Machine Learning* (*ICML*). Vol. 13. 2018, pp. 9072–9081. ISBN: 978-1-5108-6796-3.

[71] Tony Duan and Juho Lee. "Graph Embedding VAE: A Permutation Invariant Model of Graph Structure." In: (2019).

[72] Ashish Vaswani. "Attention Is All You Need arXiv:1706.03762v5." In: *Neural Information Processing Systems* (*NeurIPS*). 2017. DOI: 10.1017/S0952523813000308.

[73] Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding." In: *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*. 2019. ISBN: 978-1-950737-13-0.

[74] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. *Graph Transformer Networks*. Tech. rep.

[75] Vijay Prakash Dwivedi and Xavier Bresson. "A Generalization of Transformer Networks to Graphs." In: (2020).

[76] Dai Quoc Nguyen, Tu Dinh Nguyen, and Dinh Phung. "Universal Graph Transformer Self-Attention Networks." In: (2019). ISSN: 1909.11855v9.

[77] Charlie Nash, Yaroslav Ganin, S. M.Ali Ali Eslami, and Peter W Battaglia. "PolyGen: An Autoregressive Generative Model of 3D Meshes." In: *arXiv* (2020).

[78] Chaitanya K. Joshi. "Transformers Are Graph Neural Networks." In: *The Gradient* (2020).

[79] S. V.N. Vishwanathan, Karsten M. Borgwardt, and Nicol N. Schraudolph. "Fast Computation of Graph Kernels." In: *Advances in Neural Information Processing Systems*. 2007. ISBN: 978-0-262-19568-3. DOI: 10.7551/mitpress/7503.003.0186.

[80] Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. "GraphiT: Encoding Graph Structure in Transformers." June 2021.

[81] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. *Benchmarking Graph Neural Networks*. 2020.

[82] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. "Self-Attention with Relative Position Representations." In: *arXiv:1803.02155 [cs]* (Apr. 2018). arXiv: 1803.02155 [cs].

[83] Jiaxuan You, Rex Ying, and Jure Leskovec. "Position-Aware Graph Neural Networks." In: *arXiv:1906.04817 [cs, stat]* (June 2019). arXiv: 1906.04817 [cs, stat].

[84] Jack W Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P Lillicrap. "Compressive Transformers for Long-Range Sequence Modelling." In: *arXiv* (2019).

[85] Geoffrey Hinton, Sara Sabour, and Nicholas Frosst. "Matrix Capsules With EM Routing." In: *International Conference on Learning Representations (ICLR)*. 2018, pp. 1–15.

[86] Saurabh Verma and Zhi-Li Zhang. "Graph Capsule Convolutional Neural Networks." In: (2018).

[87] Loris Nanni, Alessandra Lumini, Federica Pasquali, and Sheryl Brahnam. "iProStruct2D: Identifying Protein Structural Classes by Deep Learning via 2D Representations." In: *arXiv* (2019).

[88] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. "Multi-View Convolutional Neural Networks for 3D Shape Recognition." In: *2015 IEEE International Conference on Computer Vision (ICCV)*. Vol. 2015 Inter. IEEE, Dec. 2015, pp. 945–953. ISBN: 978-1-4673-8391-2. DOI: 10.1109/ICCV.2015.114.

[89] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. "ABC: A Big Cad Model Dataset for Geometric Deep Learning." In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 2019-June. 2019, pp. 9593–9603. ISBN: 978-1-72813-293-8. DOI: 10.1109/CVPR.2019.00983.

[90] Ryoma Sato. "A Survey on The Expressive Power of Graph Neural Networks." In: *arXiv* (2020).

[91] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. "How Powerful Are Graph Neural Networks?" In: *International Conference on Learning Representations (ICLR)*. 2019. ISBN: 1810.00826v3.

[92] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. "Provably Powerful Graph Networks." In: *arXiv* (2019).

[93] B Yu Weisfeiler and A. A. Lehman. "Reduction of a Graph to a Canonical Form and an Algebra Which Appears in the Process." In: *Nauchno-Technicheskaya Informatsiya, Ser.* 2 9 (1968), pp. 12–12.

[94] László Babai, Paul Erdő˝s, and Stanley M Selkow. "Random Graph Isomorphism." In: *SIAM Journal on Computing* 9.3 (1980), pp. 628–635. DOI: 10.1137/0209047.

[95] Bronstein Michael. *Expressive Power of Graph Neural Networks and the Weisfeiler-Lehman Test*. 2020.

[96] Laszlo Babai and Eugene M. Luks. "Canonical Labeling of Graphs." In: *Conference Proceedings of the Annual ACM Symposium on Theory of Computing*. 1983. ISBN: 0-89791-099-0. DOI: 10.1145/800061.808746.

[97] Nino Shervashidze, S V N Vishwanathan, Tobias H Petri, Kurt Mehlhorn, and Karsten M Borgwardt. "Efficient Graphlet Kernels for Large Graph Comparison." In: *Proceedings of the 12th International Confe- Rence on Artificial Intelligence and Statistics (AISTATS)*. 2009.

[98] Karsten M. Borgwardt and Hans Peter Kriegel. "Shortest-Path Kernels on Graphs." In: *Proceedings - IEEE International Conference on Data Mining, ICDM*. 2005. ISBN: 0-7695-2278-5. DOI: 10.1109/ICDM.2005.132.

[99] Zhao Min Chen, Xiu Shen Wei, Peng Wang, and Yanwen Guo. "Multi-Label Image Recognition with Graph Convolutional Networks." In: *arXiv* (2019).

[100] Kaveh Hassani and Amir Hosein Khasahmadi. "Contrastive Multi-View Representation Learning on Graphs." In: *International Conference on Machine Learning (ICML)* (June 2020).

[101] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations Using RNN Encoder–Decoder for Statistical Machine Translation." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2014, pp. 1724–1734. ISBN: 978-1-937284-96-1. DOI: 10.3115/v1/D14-1179.

[102] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. "Deep Graph Infomax." In: *7th International Conference on Learning Representations (ICLR)* (2019).

[103] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. "Learning Deep Representations by Mutual Information Estimation and Maximization." In: *International Conference on Learning Representations (ICLR)*. 2018.

[104] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. "InfoGraph: Unsupervised and Semi-Supervised Graph-Level Representation Learning via Mutual Information Maximization." In: *International Conference on Learning Representations (ICLR)* (July 2019).

[105] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate." In: *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. 2015.

[106] Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. "A Convolutional Encoder Model for Neural Machine Translation." In: *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*. Vol. 1. 2017, pp. 123–135. ISBN: 978-1-945626-75-3. DOI: 10.18653/v1/P17-1012.

[107] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit Yan Yeung. "GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs." In: *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*. 2018. ISBN: 978-1-5108-7160-1.

[108] Yao Ma, Suhang Wang, Charu C Aggarwal, and Jiliang Tang. "Graph Convolutional Networks with EigenPooling." In: *Association for Computing Machinery* (Apr. 2019). DOI: 10.1145/nnnnnnn.nnnnnnn.

[109] Ziwei Zhang, Peng Cui, and Wenwu Zhu. "Deep Learning on Graphs: A Survey." In: *arXiv* (2018). DOI: 10.1109/tkde.2020.2981333.

[110] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. "Hierarchical Graph Representation Learning with Differentiable Pooling." In: (2018), p. 11.

[111] Junying Li, Deng Cai, and Xiaofei He. "Learning Graph-Level Representation for Drug Discovery." In: *arXiv* (2017).

[112] Weihua Hu et al. "Open Graph Benchmark: Datasets for Machine Learning on Graphs." In: *arXiv* (2020).

[113] Floris Geerts, Filip Mazowiecki, and Guillermo A Pérez. "Let's Agree to Degree: Comparing Graph Convolutional Networks in the Message-Passing Framework." In: *arXiv* (2020). ISSN: 23318422.

[114] Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M. Bronstein, and Haggai Maron. "Equivariant Subgraph Aggregation Networks." In: *arXiv:2110.02910 [cs, stat]* (Oct. 2021). arXiv: 2110.02910 [cs, stat].

[115] Johannes Klicpera, Janek Groß, and Stephan Günnemann. "DIRECTIONAL MESSAGE PASSING FOR MOLECULAR GRAPHS." In: (2020), p. 13.

[116] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. "DeepGCNs: Can GCNs Go as Deep as CNNs?" In: *Proceedings of the IEEE International Conference on Computer Vision*. Vol. 2019-Octob. 2019, pp. 9266–9275. ISBN: 978-1-72814-803-8. DOI: 10.1109/ICCV.2019.00936.

[117] Lingxiao Zhao and Leman Akoglu. "Pairnorm: Tackling Oversmoothing in GNNS." In: *International Conference on Learning Representations (ICLR)*. 2020.

[118] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. "DeeperGCN: All You Need to Train Deeper GCNs." In: *arXiv* (2020).

[119] Kezhi Kong, Guohao Li, Mucong Ding, Zuxuan Wu, Chen Zhu, Bernard Ghanem, Gavin Taylor, and Tom Goldstein. "FLAG: Adversarial Data Augmentation for Graph Neural Networks." In: (2020).

[120] Yue Xu, Hao Chen, Zengde Deng, Junxiong Zhu, Yanghua Li, Peng He, Wenyao Gao, and Wenjun Xu. "Single-Layer Graph Convolutional Networks For Recommendation." In: *arXiv*. 2020.

[121] Zhengdao Chen, Lisha Li, and Joan Bruna. "Supervised Community Detection with Line Graph Neural Networks." In: *International Conference on Learning Representations (ICLR)*. 2017.

[122] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. "Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions." In: *Machine Learning-International Workshop Then Conference-* 20.2 (2003), pp. 912–912. ISSN: 9780262255899.

[123] Xin Li, Lu Wang, Yang Xin, Yixian Yang, and Yuling Chen. "Automated Vulnerability Detection in Source Code Using Minimum Intermediate Representation Learning." In: *Applied Sciences (Switzerland)* 10.5 (2020). DOI: 10.3390/app10051692.

[124] Philip Rathle. *Driving Innovation in Retail with Graph Technology*. Tech. rep.

[125] Shiyuan Jin, Ming Zhou, and Annie S Wu. "Sensor Network Optimization Using a Genetic Algorithm." In: *7th World Multiconference on Systemics, Cybernetics and Informatics* (2003), pp. 1–6. ISSN: 2251-6581 (Electronic)\r2251-6581 (Linking).

[126] Xiaodan Liang, Xiaohui Shen, Jiashi Feng, Liang Lin, and Shuicheng Yan. "Semantic Object Parsing with Graph LSTM." In: *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 9905 LNCS. 2016, pp. 125–143. ISBN: 978-3-319-46447-3. DOI: 10.1007/978-3-319-46448-0_8.

[127] Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph Lambourne, Thomas Davies, Hooman Shayani, and Nigel Morris. "UV-Net: Learning from Curve-Networks and Solids." In: (June 2020).

[128] Nicola De Cao and Thomas Kipf. "MolGAN: An Implicit Generative Model for Small Molecular Graphs." In: (2018).

[129] Marinka Zitnik and Jure Leskovec. "Predicting Multicellular Function through Multi-Layer Tissue Networks." In: *Bioinformatics*. Vol. 33. 2017, pp. i190–i198. DOI: 10.1093/bioinformatics/btx252.

[130] Zaïd Harchaoui and Francis Bach. "Image Classification with Segmentation Graph Kernels." In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2007. ISBN: 1-4244-1180-7. DOI: 10.1109/CVPR.2007.383049.

[131] Duen Horng Chau, Carey Nachenberg, Jeffrey Wilhelm, Adam Wright, and Christos Faloutsos. "Polonium: Tera-Scale Graph Mining and Inference for Malware Detection." In: *Proceedings of the 11th SIAM International Conference on Data Mining, SDM 2011*. 2011. ISBN: 978-0-89871-992-5.

[132] Benjamin Graham. "Fractional Max-Pooling." In: *arXiv* (2014).

[133] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. "SoundNet: Learning Sound Representations from Unlabeled Video." In: *Advances in Neural Information Processing Systems*. 2016, pp. 892–900.

[134] Yann LeCun and Yoshua Bengio. "The Handbook of Brain Theory and Neural Networks." In: ed. by Michael A Arbib. Cambridge, MA, USA: MIT Press, 1998, pp. 255–258. ISBN: 0-262-51102-9.

[135] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. "Recurrent Models of Visual Attention." In: *Advances in Neural Information Processing Systems*. 2014.

[136] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. "An End-to-End Deep Learning Architecture for Graph Classification." In: *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*. 2018.

[137] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. "Strategies for Pre-Training Graph Neural Networks." In: *arXiv:1905.12265 [cs, stat]* (Feb. 2020). arXiv: 1905.12265 [cs, stat].

[138] P Erdõs and A Rényi. "On Evolution of Random Graphs." In: *Publications of the Mathematical Institute of the Hungarian Academy of Sciences* 5 (1960), pp. 17–61.

[139] Brendan D. McKay and Adolfo Piperno. "Practical Graph Isomorphism, II." In: *Journal of Symbolic Computation* 60 (2014), pp. 94–112. ISSN: 0747-7171. DOI: 10.1016/j.jsc.2013.09.003.

[140] Ulrik Brandes. "A Faster Algorithm for Betweenness Centrality." In: *Journal of Mathematical Sociology* 25.2 (2001), pp. 163–177. ISSN: 0022-250X. DOI: 10.1080/0022250X.2001.9990249.

[141] John E McMurry. *Fundamentals of Organic Chemistry*. Seventh. Brooks/Cole CENGAGE Learning, 2011.

[142] JM Berg, JL Tymoczko, and L Stryer. "Mutations Involve Changes in the Base Sequence of DNA." In: *Biochemistry*. Vol. 27.6. New York: W H Freeman, 2002.

[143] "Steric Hindrance." In: (Aug. 2020).

[144] Jonathan Clayden, Nick Greeves, Stuart Warren, and Peter Wothers. "Organic Chemistry." In: (2001).

[145] Mojtaba Alipour and Faezeh Taravat. "Efficiency of Electrostatic and Steric Forces in Theoretical Appreciating Chemical Reactivity-Related Phenomena." In: *Molecular Physics* 117.2 (2019), pp. 136–142.

[146] Marcus D Hanwell, Donald E Curtis, David C Lonie, Tim Vandermeersch, Eva Zurek, and Geoffrey R Hutchison. "Avogadro: An Advanced Semantic Chemical Editor, Visualization, and Analysis Platform." In: *J. Cheminformatics* 4.1 (Dec. 2012), p. 17. DOI: 10.1186/1758-2946-4-17.

[147] Nikolay Golovanov. *Geometric Modeling: The Mathematics of Shapes*. Reprint edition (24 Dec. 2014). CreateSpace Independent Publishing Platform, 2011. ISBN: 1-4974-7319-5.

[148] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. "Image Style Transfer Using Convolutional Neural Networks." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2016-Decem. IEEE, June 2016, pp. 2414–2423. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.265.

[149] Zhaoliang Lun, Evangelos Kalogerakis, and Alla Sheffer. "Elements of Style: Learning Perceptual Shape Style Similarity." In: *ACM Transactions on Graphics*. Vol. 34. 2015. ISBN: 978-1-4503-3331-3. DOI: 10.1145/2766929.

[150] Tianqiang Liu, Aaron Hertzmann, Wilmot Li, and Thomas Funkhouser. "Style Compatibility for 3D Furniture Models." In: *ACM Transactions on Graphics* 34.4 (July 2015), pp. 1–9. DOI: 10.1145/2766898.

[151] Isaak Lim, Anne Gehre, and Leif Kobbelt. "Identifying Style of 3D Shapes Using Deep Metric Learning." In: *Computer Graphics Forum* 35.5 (Aug. 2016), pp. 207–215. DOI: 10.1111/cgf.12977.

[152] Tse-Yu Pan, Yi-Zhu Dai, Wan-Lun Tsai, and Min-Chun Hu. "Deep Model Style: Cross-Class Style Compatibility for 3D Furniture within a Scene." In: *2017 IEEE International Conference on Big Data (Big Data)*. Vol. 2018-Janua. IEEE, Dec. 2017, pp. 4307–4313. ISBN: 978-1-5386-2715-0. DOI: 10.1109/BigData.2017.8258459.

[153] Luisa F. Polania, Mauricio Flores, Yiran Li, and Matthew Nokleby. "Learning Furniture Compatibility with Graph Neural Networks." In: (2020).

[154] Xiang Pan, Jie Lu, and Fuchang Liu. "3D Patch-Based Sparse Learning for Style Feature Extraction." In: *IEEE Access* 7 (2019), pp. 172403–172412. DOI: 10.1109/ACCESS.2019.2954693.

[155] Kevin J Weiler. "Topological Structures for Geometric Modeling." In: (1986), pp. 340–340.

[156] Hiroshi Masuda, Kenji Shimada, Masayuki Numao, and Shinji Kawabe. "A Mathematical Theory and Applications of Non-Manifold Geometric Modelling." In: *International Symposium on Advanced Geometric Modelling for Engineering Applications*. 1989, pp. 89–103. ISBN: 978-1-62623-977-7.

[157] Zhangjie Cao, Qixing Huang, and Ramani Karthik. "3D Object Classification via Spherical Projections." In: *2017 International Conference on 3D Vision (3DV)*. IEEE, Oct. 2017, pp. 566–574. ISBN: 978-1-5386-2610-8. DOI: 10.1109/3DV.2017.00070.

[158] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. "3D ShapeNets: A Deep Representation for Volumetric Shapes." In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 07-12-June. IEEE, June 2015, pp. 1912–1920. ISBN: 978-1-4673-6964-0. DOI: 10.1109/CVPR.2015.7298801.

[159] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation." In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. Vol. 2017-Janua. 2017, pp. 77–85. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.16.

[160] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. "Deep Learning for 3D Point Clouds: A Survey." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), pp. 1–1. ISSN: 23201/20351. DOI: 10.1109/TPAMI.2020.3005434.

[161] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. "MeshCNN: A Network with an Edge." In: *ACM Transactions on Graphics* 38.4 (July 2019), pp. 1–12. ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3306346.3322959. arXiv: 1809.05910.

[162] Pim de Haan, Maurice Weiler, Taco Cohen, and Max Welling. "Gauge Equivariant Mesh CNNs: Anisotropic Convolutions on Geometric Graphs." In: (Mar. 2020). ISSN: 2003.05425v1.

[163] David Griffiths and Jan Boehm. "A Review on Deep Learning Techniques for 3D Sensed Data Classification." In: *Remote Sensing* 11.12 (June 2019), pp. 1499–1499. DOI: 10.3390/rs11121499.

[164] Eman Ahmed, Alexandre Saint, Abd El Rahman Shabayek, Kseniya Cherenkova, Rig Das, Gleb Gusev, Djamila Aouada, and Bjorn Ottersten. "A Survey on Deep Learning Advances on Different 3D Data Representations." In: *arXiv:1808.01462 [cs]* (Apr. 2019). arXiv: 1808.01462 [cs].

[165] Xun Huang and Serge Belongie. “Arbitrary Style Transfer in Real-Time with Adaptive Instance Normalization.” In: *2017 IEEE International Conference on Computer Vision (ICCV)*. Vol. 2017-Octob. IEEE, Oct. 2017, pp. 1510–1519. ISBN: 978-1-5386-1032-9. DOI: 10.1109/ICCV.2017.167.

[166] Mohammad Babaeizadeh and Golnaz Ghiasi. “Adjustable Real-Time Style Transfer.” In: *Deep Generative Models for Highly Structured Data, DGS@ICLR 2019 Workshop* (Nov. 2018). ISSN: 1811.08560v1.

[167] Tero Karras, Samuli Laine, and Timo Aila. “A Style-Based Generator Architecture for Generative Adversarial Networks.” In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Dec. 2019), pp. 4396–4405. ISSN: 978-1-7281-3293-8. DOI: 10.1109/CVPR.2019.00453.

[168] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. “Analyzing and Improving the Image Quality of StyleGAN.” In: (2019).

[169] Yulun Zhang, Chen Fang, Yilin Wang, Zhaowen Wang, Zhe Lin, Yun Fu, and Jimei Yang. “Multimodal Style Transfer via Graph Cuts.” In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Vol. 2019-Octob. IEEE, Oct. 2019, pp. 5942–5950. ISBN: 978-1-72814-803-8. DOI: 10.1109/ICCV.2019.00604.

[170] Liming Jiang, Changxu Zhang, Mingyang Huang, Chunxiao Liu, Jianping Shi, and Chen Change Loy. “TSIT: A Simple and Versatile Framework for Image-to-Image Translation.” In: *arXiv* (2020).

[171] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu. “Contrastive Learning for Unpaired Image-to-Image Translation.” In: *arXiv* (2020).

[172] Samaneh Azadi, Matthew Fisher, Vladimir Kim, Zhaowen Wang, Eli Shechtman, and Trevor Darrell. “Multi-Content GAN for Few-Shot Font Style Transfer.” In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7564–7573. ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00789.

[173] Hsueh-Ti Derek Liu, Vladimir G. Kim, Siddhartha Chaudhuri, Noam Aigerman, and Alec Jacobson. “Neural Subdivision.” In: *arXiv:2005.01819 [cs]* (May 2020). arXiv: 2005.01819 [cs].

[174] Xu Cao, Weimin Wang, Katashi Nagao, and Ryosuke Nakamura. “PSNet: A Style Transfer Network for Point Cloud Stylization on Geometry and Color.” In: *IEEE Winter Conference on Applications of Computer Vision (WACV)* (2020), pp. 3326–3334. ISSN: 9781728165530. DOI: 10.1109/WACV45572.2020.9093513.

[175] Angel X Chang et al. “ShapeNet: An Information-Rich 3D Model Repository.” In: *arXiv* (2015).

[176] Mattia Segu, Margarita Grinvald, Roland Siegwart, and Federico Tombari. "3DSNet: Unsupervised Shape-to-Shape 3D Style Transfer." In: *arXiv* (Nov. 2020).

[177] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Improved Texture Networks: Maximizing Quality and Diversity in Feed-Forward Stylization and Texture Synthesis." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2017-Janua. IEEE, July 2017, pp. 4105–4113. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.437.

[178] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." In: *Nature Methods* 17.3 (Mar. 2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[179] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. "EMNIST: An Extension of MNIST to Handwritten Letters." In: *arXiv* (Feb. 2017).

[180] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space." In: *Advances in Neural Information Processing Systems* 2017-Decem (June 2017), pp. 5100–5109.

[181] Guillaume Alain and Yoshua Bengio. "Understanding Intermediate Layers Using Linear Classifier Probes." In: (2016).

[182] Tom Lenaerts, Dominique Groß, and Richard Watson. "On the Modelling of Dynamical Hierarchies : Introduction to the Workshop." In: *Proceedings of the Alife VIII workshop* 31 (2002), pp. 37–37.

[183] Mileidy W. Gonzalez and Maricel G. Kann. "Chapter 4: Protein Interactions and Disease." In: *PLoS Computational Biology* 8.12 (2012). ISSN: 1553-7358 (Linking). DOI: 10.1371/journal.pcbi.1002819.

[184] Christos Sakellariou and Peter J Bentley. "Demonstrating the Performance , Flexibility and Programmability of the Hardware Architecture of Systemic Computation Modelling Cancer Growth." In: *Bio-Inspired Computation* 7.6 (2015).

[185] Nigel Gilbert. "Agent-Based Social Simulation: Dealing with Complexity." In: *The Complex Systems Network of Excellence* 9 (2004), pp. 1–14. ISSN: 9780203461730. DOI: 10.4114/ia.v9i25.771.

[186] Hiroki Sayama. "Generative Network Automata: A Generalized Framework for Modeling Complex Dynamical Systems with Autonomously Varying Topologies." In: *Proceedings of the 2007 IEEE Symposium on Artificial Life, CI-ALife 2007* (2007), pp. 214–221. ISSN: 142440701X. DOI: 10.1109/ALIFE.2007.367799.

[187] M. Batty, Yichun Xie, and Zhanli Sun. "Modeling Urban Dynamics through GIS-Based Cellular Automata." In: *Computers, Environment and Urban Systems* 23.3 (1999), pp. 205–233. ISSN: 0198-9715. DOI: 10.1016/S0198-9715(99)00015-0.

[188] M Webster and G Malcolm. "Hierarchical Components and Entity-Based Modelling in Artificial Life." In: *Artificial Life XI: Proceedings of the 11th International Conference on the Simulation and Synthesis of Living Systems, ALIFE 2008* 1991 (2008), pp. 678–685. ISSN: 9780262750172.

[189] Robert Axelrod. *The Complexity of Cooperation: Agent-Ased Models of Competition and Collaoration*. 1997, p. 248. ISBN: 0-691-01567-8. DOI: 10.1002/(SICI)1099-0526(199801/02)3:3<46::AID-CPLX6>3.3.CO;2-#.

[190] Joshua M. Epstein. "Agent-Based Computational Models and Generative Social Science." In: *Complexity* 4.5 (1999), pp. 41–60. ISSN: 9780387257464. DOI: 10.1002/(SICI)1099-0526(199905/06)4:5<41::AID-CPLX9>3.0.CO;2-F.

[191] Salman Ahmed, Mohd N. Karsiti, and Herman Agustiawan. "A Development Framework for Collaborative Robots Using Feedback Control." In: *arXiv* (2006).

[192] I. Arel, C. Liu, T. Urbanik, and A.G. Kohls. "Reinforcement Learning-Based Multi-Agent System for Network Traffic Signal Control." In: *IET Intelligent Transport Systems* 4.2 (2010), pp. 128–128. ISSN: 1751956X. DOI: 10.1049/iet-its.2009.0070.

[193] Lorenzo Bortot, Bernhard Auchmann, Idoia Cortes Garcia, Alejando M. Fernando Navarro, Michal Maciejewski, Matthias Mentink, Marco Prioli, Emmanuele Ravaioli, Sebastian Schops, and Arjan Verweij. "STEAM: A Hierarchical Co-Simulation Framework for Superconducting Accelerator Magnet Circuits." In: *IEEE Transactions on Applied Superconductivity* 28.3 (2017). DOI: 10.1109/TASC.2017.2787665.

[194] Yao Yao and Yves Van de Peer. "Simulating Biological Complexity through Artificial Evolution." In: *2017 3rd IEEE International Conference on Cybernetics (CYBCONF)*. IEEE, June 2017, pp. 1–8. ISBN: 978-1-5386-2201-8. DOI: 10.1109/CYBConf.2017.7985809.

[195] Thilo Gross, Carlos D'Lima, and Bernd Blasius. "Epidemic Dynamics on an Adaptive Network." In: *Physical Review Letters* 96.20 (2006), pp. 208701–208701. DOI: 10.1103/PhysRevLett.96.208701.

[196] Sebastian Funk, Marcel Salathé, and Vincent a a Jansen. "Modelling the Influence of Human Behaviour on the Spread of Infectious Diseases: A Review." In: *Journal of the Royal Society, Interface / the Royal Society* 7.50 (2010), pp. 1247–56. ISSN: 1742-5689. DOI: 10.1098/rsif.2010.0142.

[197] Barak A. Pearlmutter and Conor J. Houghton. "A New Hypothesis for Sleep: Tuning for Criticality." In: *Neural Computation* 21.6 (2009), pp. 1622–1641. ISSN: 0899-7667 (Print)\r0899-7667 (Linking). DOI: 10.1162/neco.2009.05-08-787.

[198] Hiroki Sayama, Irene Pestov, Jeffrey Schmidt, Benjamin James Bush, Chun Wong, Junichi Yamanoi, and Thilo Gross. "Modeling Complex Systems with Adaptive Networks." In: *Computers and Mathematics with Applications* 65.10 (2013), pp. 1645–1664. ISSN: 0898-1221. DOI: 10.1016/j.camwa.2012.12.005.

[199] Ian Robinson, Jim Webber, and Emil Eifrem. *Graph Databases*. O'Reilly, 2014, p. 46. ISBN: 978-0-12-407192-6. DOI: 10.1016/B978-0-12-407192-6.00003-0.

[200] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. "Cypher: An Evolving Query Language for Property Graphs." In: *Proceedings of the 2018 International Conference on Management of Data*. Houston TX USA: ACM, May 2018, pp. 1433–1445. ISBN: 978-1-4503-4703-7. DOI: 10.1145/3183713.3190657.

[201] Steven a. Frank. "Cell Lineage History." In: *Dynamics of Cancer. Incidence, Inheritance, and Evolution*. Princeton University Press, 2007, pp. 1–378. ISBN: ISBN-13: 978-0-691-13366-9. DOI: 10.1053/j.gastro.2010.01.058.

[202] Helen Bolton, Sarah J.L. Graham, Niels Van Der Aa, Parveen Kumar, Koen Theunis, Elia Fernandez Gallardo, Thierry Voet, and Magdalena Zernicka-Goetz. "Mouse Model of Chromosome Mosaicism Reveals Lineage-Specific Depletion of Aneuploid Cells and Normal Developmental Potential." In: *Obstetrical and Gynecological Survey* 71.11 (2016), pp. 665–666. ISSN: 2041-1723 (Electronic)\r2041-1723 (Linking). DOI: 10.1097/01.ogx.0000508248.22573.8b.

[203] T Cremer, C Cremer, Not at Dartmouth/Dhmclibraries, and request on interlibrary loan. "Chromosome Territories, Nuclear Architecture and Gene Regulation in Mammalian Cells." In: *Nature Reviews Genetics* 2.4 (2001), pp. 292–301. ISSN: 1471-0056.

[204] Arturo Araujo. "Modelling Chromosome Missegregation in Tumour Evolution." PhD thesis. 2013, p. 251.

[205] Mel Greaves and Carlo C. Maley. "Clonal Evolution in Cancer." In: *Nature* 481.7381 (2012), pp. 306–313. ISSN: 1476-4687 (Electronic)\n0028-0836 (Linking). DOI: 10.1038/nature10762.

[206] Andrea Sottoriva et al. "A Big Bang Model of Human Colorectal Tumor Growth." In: *Nature Genetics* 47.3 (2015), pp. 209–216. ISSN: 1546-1718 (Electronic) 1061-4036 (Linking). DOI: 10.1038/ng.3214.

[207] Pinar Yanardag and S.V.N. Vishwanathan. "Deep Graph Kernels." In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15*. 2015, pp. 1365–1374. ISBN: 978-1-4503-3664-2. DOI: 10.1145/2783258.2783417.

[208] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. "Self-Attention Graph Pooling." In: *International Conference on Machine Learning (ICML)* (2019).

[209] Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph G. Lambourne, Karl D. D. Willis, Thomas Davies, Hooman Shayani, and Nigel Morris. "UV-Net: Learning from Boundary Representations." In: *arXiv:2006.10211 [cs]* (Apr. 2021). arXiv: 2006.10211 [cs].

[210] John S. Breese, David Heckerman, and Carl Kadie. "Empirical Analysis of Predictive Algorithms for Collaborative Filtering." In: *arXiv:1301.7363 [cs]* (Jan. 2013). arXiv: 1301.7363 [cs].

[211] Balázs Hidasi and Alexandros Karatzoglou. "Recurrent Neural Networks with Top-k Gains for Session-Based Recommendations." In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (Oct. 2018), pp. 843–852. DOI: 10.1145/3269206.3271761. arXiv: 1706.03847.

[212] Yong Kiam Tan, Xinxing Xu, and Yong Liu. "Improved Recurrent Neural Networks for Session-Based Recommendations." In: *arXiv:1606.08117 [cs]* (Sept. 2016). arXiv: 1606.08117 [cs].

[213] Shiwen Wu, Fei Sun, Wentao Zhang, and Bin Cui. "Graph Neural Networks in Recommender Systems: A Survey." In: *arXiv:2011.02260 [cs]* (Apr. 2021). arXiv: 2011.02260 [cs].