University College London

Doctoral Thesis

Overcoming data bottlenecks in T cell receptor specificity prediction with effective machine learning

Author:
Yuta Nagano

Supervisors:
Prof. Benny Chain
Prof. Sergio Quezada
Dr. Andreas Tiffeau-Mayer

Thesis Committee:
Prof. John Shawe-Taylor
Prof. Sergio Quezada
Prof. Trevor Graham
Dr. Andreas Tiffeau-Mayer

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

July 21, 2024

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

Contents

Acknowledgements

To Benny. Thank you very much for all of your guidance, and the belief that you put in me. Without you, the past three years that I have truly enjoyed and learned so much from would not have been possible. There were many times that your belief in me helped me to keep going. Even from before the start of my PhD candidacy when I was struggling to find funding, you were there to help me. During the early days of my candidacy when I was ready to give up on my own ideas, you encouraged me to persevere. And every time I came to you with ideas for a side-project or a new avenue of investigation, your response was *always* to nurture and guide me with your knowledge and wisdom on how I could best address those new interests – never to shut them down, or force a change in direction. Through your mentorship and example, I have learned a lot about how to ask good, tractable scientific questions, and also a bit on how to work towards answring these questions as part of a larger team. I feel that my time during this PhD candidacy has deeply impacted my growth both as a scientist and person, and it has been an honour and a privilege. One day I hope to become as kind, intelligent and wise of a supervisor or teacher to someone else as you have been to me.

To Andreas. Thank you for being such an open and approachable mentor and role model. I've been very lucky to have somebody from whom I can learn how to approach difficult problems using quantitative tools, while also enjoying the occasional blitz chess game after work at the pub, with odds not (yet) in my favour. The fact that you – with your incredibly sharp intuition and proficiency around mathematical methods – always fully included me in your group's quantitative discussions despite the fact that I am neither classically trained in mathematics nor physics, has massively helped me build up my self-confidence around discussing and learning mathematics, and I am very grateful for that. Also, thank you for always being patient with me, even though I can have excessively strong

opinions about what clean code should look like, or which text editor is the best one and why everyone should use it. I look forward to the day I can win against you in a blitz game without you handicapping yourself with alcohol.

家族へ。　パパ、ママ、いつも僕の事を応援してくれてありがとう。パパとママの応援があるから、大変な日も頑張れるし、楽しい日を安心して楽しむ事が出来ます。小さい頃から僕に沢山の刺激、経験や機会を与えてくれたから、科学や宇宙への興味を育んでくれたから、インターナショナルスクールに通わせて英語で教育を受けさせてくれたから、イギリスに一緒に連れて行ってくれたから、ロンドンの大学に送ってくれたから、沢山の興味を持たせてくれたから、こうやって自分の興味のある研究に励み、博士課程をする事が出来ました。また、いいちゃまやああちゃま、出雲のおじいちゃんやおばあちゃん、そして他の家族のみんなも、いつも応援してくれてありがとう。これからも世のため人のために頑張り続けるので、引き続き宜しくお願いします！

To Nora. Thank you for always supporting me and being a source of positivity, happiness, and energy in my life. Seeing you work hard to achieve your own goals every day always motivated me to do the same, and knowing that you believe in me really helped me enjoy working hard on my projects during my PhD candidacy. On those days when I was exhausted and needed a break from my work, you were always there to cheer me up and make me laugh. On other days when I was in the zone and was eager to push on with my research, you were always there to give me support and encouragement. From the bottom of my heart, thank you, muchas gracias, og tusen takk!

To John, Sergio, and Trevor. Thank you all very much for the mentorship, guidance and encouragement you've given me both inside and outside our thesis committee meetings. I'm very grateful to have been surrounded by, and have received academic and career advice from leading professors like you. It has been an honour and privilege.

To Prof. Noursadeghi (Maddy), Dr. Hasford, Carolyn, Prof. Burns, and Prof. Motallebzadeh. Thank you all for making the UCL MBPhD programme possible. Maddy and Dr. Hasford, thank you for always being there for the 8AM MBPhD journal clubs, and drilling into us students the habit and importance of critical medical research literacy. Carolyn, thank you so much for being the point of contact for us students, consistently

connecting us to teaching opportunities, helping us through university bureaucracy, and always supporting us whenever we needed help. Prof. Burns and Prof. Motallebzadeh, thank you for leading the MBPhD programme, and giving us students opportunities to get clinical exposure and teaching during the PhD. This programme has provided me with such an important opportunity for me to step into research, and I'm very grateful to have been part of it.

To the members of the Innate2Adaptive and QImmuno groups. Thank you all for having been the best friends and colleagues that one could have asked for. You all made my PhD experience fun and full of happy (and sometimes quite funny) memories. I want to say a special thanks to Martina, for showing me the ropes when I first joined the group, and always being there whenever I needed help or advice. They say that people/colleagues make or break a job – you guys all honestly made my PhD experience *amazing*.

To Iris. Thank you for being a great postgraduate tutor for our CRUK MBPhD cohort. You guidance and support helped me through my transition from being a medical student to a postgraduate research student, and the way that you regularly checked up on us helped me feel reassured that you would be there if anything went wrong.

Funding. Finally, thank you to **Cancer Research UK**, the **UCL Overseas Research Scholarship** and the **Masason Foundation** for their financial support throughout my PhD candidacy.

UCL research paper declaration form: referencing the doctoral candidate's own published work(s) (1 of 2)

1. **For a research manuscript that has already been published** (if not yet published, please skip to section 2)**:**
   (a) **What is the title of the manuscript?**
   tidytcells: standardizer for TR/MH nomenclature
   (b) **Please include a link to or doi for the work.**
   `https://doi.org/10.3389/fimmu.2023.1276106`
   (c) **Where was the work published?**
   Frontiers in Immunology
   (d) **Who published the work?**
   Frontiers
   (e) **When was the work published?**
   25th October 2023
   (f) **List the manuscript's authors in the order they appear on the publication.**
   Yuta Nagano, Benny Chain
   (g) **Was the work peer reviewd?**
   Yes
   (h) **Have you retained the copyright?**
   Yes
   (i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)? If 'Yes', please give a link or doi.**
   Yes: `https://www.biorxiv.org/content/10.1101/2023.09.21.`

558778v1

If 'No', please seek permission from the relevant publisher and check the box next to the below statement:

☐ *I acknowledge permission of the publisher named under 1d to include in this thesis portions of the publication named as included in 1c.*

2. **For a research manuscript prepared for publication but that has not yet been published** (if already published, please skip to section 3)**:**
   (a) **What is the current title of the manuscript?**
   (b) **Has the manuscript been uploaded to a preprint server 'e.g. medRxiv'? If 'Yes', please please give a link or doi.**
   (c) **Where is the work intended to be published?**
   (d) **List the manuscript's authors in the intended authorship order.**
   (e) **What is the stage of publication?**
3. **For multi-authored work, please give a statement of contribution covering all authors** (if single-author, please skip to section 4)**.**
   YN: Conceptualisation, Methodology, Software, Writing original draft, Writing – review & editing. BC: Supervision, Writing – review & editing.
4. **In which chapter(s) of your thesis can this material be found?**
   Chapter 3

e-Signatures confirming that the information above is accurate (this form should be co-signed by the supervisor/ senior author unless this is not appropriate, e.g. if the paper was a single-author work)**:**

Candidate: Yuta Nagano
Date: 18th July, 2024

Supervisor/Senior Author signature (where appropriate)**:** Benny Chain
Date: 19th July, 2024

UCL research paper declaration form: referencing the doctoral candidate's own published work(s) (2 of 2)

1. **For a research manuscript that has already been published** (if not yet published, please skip to section 2)**:**
   (a) **What is the title of the manuscript?**
   (b) **Please include a link to or doi for the work.**
   (c) **Where was the work published?**
   (d) **Who published the work?**
   (e) **When was the work published?**
   (f) **List the manuscript's authors in the order they appear on the publication.**
   (g) **Was the work peer reviewd?**
   (h) **Have you retained the copyright?**
   (i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)? If 'Yes', please give a link or doi.**

   If 'No', please seek permission from the relevant publisher and check the box next to the below statement:

   ☐ *I acknowledge permission of the publisher named under 1d to include in this thesis portions of the publication named as included in 1c.*

2. **For a research manuscript prepared for publication but that has not yet been published** (if already published, please skip to section 3)**:**

(a) **What is the current title of the manuscript?**
Contrastive learning of T cell receptor representations

(b) **Has the manuscript been uploaded to a preprint server 'e.g. medRxiv'? If 'Yes', please please give a link or doi.**
Yes: `https://arxiv.org/abs/2406.06397`

(c) **Where is the work intended to be published?**
Cell Systems

(d) **List the manuscript's authors in the intended authorship order.**
Yuta Nagano, Andrew Pyo, Martina Milighetti, James Henderson, John Shawe-Taylor, Benny Chain, Andreas Tiffeau-Mayer

(e) **What is the stage of publication?**
Under review

3. **For multi-authored work, please give a statement of contribution covering all authors** (if single-author, please skip to section 4)**.**
YN: Conceptualisation, Methodology, Analysis, Interpretation, Writing original draft, Writing – review & editing. AP: Methodology, Interpretation, Writing – review & editing. MM: Interpretation, Writing – review & editing. JH: Interpretation, Writing – review & editing. JST: Writing - review & editing. BC: Conceptualisation, Supervision, Interpretation, Writing – review & editing. AM: Conceptualisation, Supervision, Interpretation, Writing – review & editing.

4. **In which chapter(s) of your thesis can this material be found?**
Chapter 4

e-Signatures confirming that the information above is accurate (this form should be co-signed by the supervisor/ senior author unless this is not appropriate, e.g. if the paper was a single-author work)**:**

Candidate: Yuta Nagano
Date: 18th July, 2024

Supervisor/Senior Author signature (where appropriate)**:** Andreas Tiffeau-Mayer
Date: 19th July, 2024

Abstract

T cells are an integral part of the adaptive immune system. They detect host cells that have been compromised due to infection or mutation, and eliminate them through direct attack or recruitment of other effector immune cell types. This is achieved through T cell receptors (TCRs) expressed on their surface, which allows T cells to bind to peptide-major histocompatibility complexes (pMHCs) in a target-specific manner.

Uncovering the rules of TCR-pMHC specificity has the potential for profound and positive impact in biomedicine. However, this remains an unsolved challenge made particularly difficult by the immense numbers of possible TCRs ($\sim 10^{60}$) and pMHCs ($\sim 10^{15}$). While it remains implausible to empirically map out anywhere near a majority of the possible binders, an application of machine learning may help us better understand the binding rules by extrapolating from existing data.

Recent advances in the natural language processing (NLP) field have demonstrated the impressive ability of transformer language models to learn from unsupervised objectives using large corpora of unlabelled text. Since TCRs, like other proteins, can naturally be represented as a sequence of amino acids, there has been growing interest in applying language modelling technologies to the TCR domain. However, how to most effectively design unsupervised training objectives to optimise language models for downstream TCR-pMHC specificity prediction remains an open question.

The core theme of this thesis is the investigation of contrastive learning as a method of training transformer-based TCR representation models. In this regard, I show that combining unsupervised contrastive learning (autocontrastive learning) with the traditional masked-language modelling (MLM) objective is a highly effective way of pre-training a TCR representation model. In addition to the above, the thesis presents related work that I have conducted during my PhD candidacy around automated TCR data standardisation as well as a statistical framework for calibrating TCR distance metrics to probabilities of TCR co-specificity.

Impact statement

From an academic point of view, my work presented in this thesis conveys useful ideas for furthering our progress towards tacking the unsolved grand challenge of predicting TCR-pMHC binding using machine learning technologies. This includes 1) how to focus machine learning models on useful information about TCR specificity contained in their amino acid sequences, 2) ways we can address the need for robust uncertainty quantification in TCR specificity prediction, and 3) what the most effective strategies for TCR data generation may be from the perspective of training computational models of specificity.

From a more practical point of view, my work provides the community with useful, free and open-source tools to expedite and improve pipelines for quantitative and computational TCR analysis. As a result of my efforts presented in chapter 3, I maintain an open-source software library named tidytcells (`https://pypi.org/project/tidytcells`) that helps scientists around the globe prepare standardised and computer-readable TCR/major histocompatibility complex (MHC) datasets – a prerequisite for effective TCR data analysis. I have also made publicly available the state-of-the-art TCR representation model we call Simple contrastive embedding of the primary sequence of T cell receptors (SCEPTR), which resulted from my work in chapter 4 (`https://pypi.org/project/sceptr`). SCEPTR provides the community with an easy-to-use tool to generate high-quality vector embeddings of TCRs for downstream specificity prediction and alignment-free TCR clustering. While my work in chapter 5 on Bayesian nearest neighbour association (BANANAS) – a statistical framework for converting TCR distance measures into well-calibrated predictions of TCR specificity – has not yet been shared with the wider public, my collaborators and I plan to distribute an open-source implementation of the methodology once it is sufficiently polished and fully investigated.

There are a multitude of potentially high-impact applications of the presented technologies. For example, using SCEPTR's TCR embeddings,

it may be possible to train algorithms to detect diseases based on the distribution of TCR sequences in patients' peripheral blood samples. This could provide minimally invasive screening against numerous conditions and save lives while potentially reducing healthcare costs by preventing further disease progression. Alternatively, SCEPTR's measures of TCR similarity and its conversion to calibrated probabilities of specificity with BANANAS may be useful for detecting clusters of similar TCRs that proliferate in response to antigens of interest. This could better inform and expedite the identification of candidate TCRs for T cell therapies, as well as reverse epitope discovery for vaccine design, and enable more cost-effective development of both curative and preventative therapeutics. In general, tidytcells, SCEPTR, and BANANAS together provide the community with the necessary components to build a fully open-source pipeline for high-throughput computational screening of TCR specificity: tidytcells for the collation and pre-processing of specificity-annotated reference TCR data, SCEPTR for detecting query TCRs that are similar to those in the reference set, and BANANAS for the generation of well-calibrated TCR specificity predictions based on SCEPTR's TCR similarity measures. Such a pipeline can be used for the purposes of any of the applications mentioned above, and more.

Finally, while all technologies presented in this thesis have been or plan to be released in a free and open-source manner, improved versions of the tools – either through further technical innovation or through the provision of high-quality and high-volume proprietary TCR data – have the potential to be effectively monetised for commercial purposes.

Publications and preprints

Below are a list of the publications and preprints that have come out as a result of my work during my PhD candidacy. Some of them are featured as chapters of this thesis.

tidytcells: standardizer for TR/MH nomenclature Yuta Nagano, Benjamin Chain, Frontiers in Immunology. Link: `https://doi.org/10.3389/fimmu.2023.1276106`

Limits on Inferring T-cell Specificity from Partial Information James Henderson, Yuta Nagano, Martina Milighetti, Andreas Tiffeau-Mayer, arxiv.org. Link: `https://arxiv.org/abs/2404.12565`

Intra- and inter-chain contacts determine TCR specificity: applying protein co-evolution methods to TCR$\alpha\beta$ pairing Martina Milighetti, Yuta Nagano, James Henderson, Uri Hershberg, Andreas Tiffeau-Mayer, Anne-Florence Bitbol, Benny Chain, biorxiv.org. Link: `https://www.biorxiv.org/content/10.1101/2024.05.24.595718v1`

Contrastive learning of T cell receptor representations Yuta Nagano, Andrew Pyo, Martina Milighetti, James Henderson, John Shawe-Taylor, Benny Chain, Andreas Tiffeau-Mayer, arxiv.org. Link: `https://arxiv.org/abs/2406.06397`

Chapter 1

Overview

The purpose of this chapter is to give the reader an executive overview of the contents of the following chapters, and provide a narrative of how the various pieces of work have come together and influenced one another.

Chapter 2 provides the reader with some background knowledge on the key topics that come up later in this thesis. I first provide a high-level overview of T cell biology and the importance and challenge of studying the interaction between T cell receptors (TCRs) and peptide-major histocompatibility complexes (pMHCs). This is followed by an introduction to the study of TCR coincidence statistics: a framework through which to quantitatively examine the relationship between measures of TCR similarity and their probabilities of sharing pMHC specificity. I dedicate the final sections of this chapter to relevant machine learning concepts including transformer neural networks, commonly used measures of model performance, and a brief review of existing literature in the space of TCR machine learning.

In chapter 3, I go over tidytcells, a free an open-source TCR/pMHC data standardisation software that I developed during the early part of my PhD candidacy. The first order of business when applying machine learning to the TCR field is to prepare clean, consistent and machine-readable TCR data. When surveying available TCR/pMHC data from both publicly available as well as internal sources, I found that TCR and major histocompatibility complex (MHC) nomenclature was not standardised across different datasets. This was an obstacle when trying to design pipelines for data analysis and machine learning, as my data parsing logic often had to be modified depending on the dataset I was looking at. Furthermore, combining independent datasets became difficult as I often had to resolve nomenclature between them. I wrote tidytcells to automate the nomenclature standardisation process, and to promote

the usage of International immunogenetics information system (IMGT)-compliant TCR and MHC nomenclature.

Chapter 4 covers my work investigating the application of language modelling technologies to TCR machine learning. This has been the main focus of my PhD candidacy. Currently, efforts to apply machine learning to computational TCR specificity predictions suffer from the limited availability of specificity-labelled TCR data. One way to alleviate this data bottleneck is to exploit the larger volume of available unlabelled TCR data for pre-training an unsupervised representation model. TCRs, like other proteins, can naturally be represented as a sequence of amino acid residues. This fact, coupled with the recent advances in natural language processing (NLP) and the advent of pre-trained transformers, has made the application of the latter technology to the field an area of active interest. However, how to best go about such an application currently remains an open question. To address this gap in our knowledge, I investigated the use of unsupervised contrastive learning (autocontrastive learning) as a pre-training strategy for a transformer-based TCR representation model. The resulting model which we call Simple contrastive embedding of the primary sequence of T cell receptors (SCEPTR), shows superior downstream TCR specificity prediction performance compared to existing protein language models (PLMs) that are only trained through the more traditional objective of masked-language modelling (MLM). I also provide evidence that supervised contrastive learning can be used as a fine-tuning strategy to specialise TCR language models on discriminating between a number of predetermined pMHC specificities of interest.

Something else that I demonstrate in chapter 4 is that when the number of known binders for a pMHC is small, simple nearest-neighbour prediction based on embedding space distances remains competitive with more complex algorithmic approaches. Now, one of the end goals of quantitative TCR analysis is its application in the biomedical field, where the cost of incorrect inference is often high. For this reason, it is of utmost importance to consider how computational models of TCR specificity can provide robust uncertainty quantification. To this end, I have been jointly leading a collaborative effort on Bayesian nearest neighbour association (BANANAS): a principled framework through which such TCR distance measurements can be rescaled to generate well-calibrated estimations of the posterior probability of TCR specificity given the distance values that are observed between a query and reference TCRs. I present my contributions to our work in progress in chapter 5. While I focus on the application of BANANAS to distances as measured by SCEPTR, the framework

is general and could theoretically be used to calibrate any arbitrary measure of distances between TCRs.

I conclude my thesis with an executive discussion in chapter 6. There, I provide the reader with an executive review of the work presented, its contributions to the field, its limitations, and finally speculate on avenues for future research.

Chapter 2

Background

2.1 A brief introduction to T cells

The body's immune system can largely be divided into two parts: the innate arm and the adaptive arm [1]. Cells of the innate immune system are more ubiquitous across the body, and can respond more rapidly to incoming invaders. However, they lack the ability to mount focused, target-specific responses, and thus are usually unable to completely clear the source of immune stimulus (such as an infectious agent) on their own. This is why an initial innate immune encounter against an invader is designed to then invoke a subsequent adaptive immune response, which can mount highly specific responses against the target in question. T cells are a central cell type of the adaptive immune system, whose wide range of roles include the orchestration, regulation, and execution of the adaptive immune response [2].

2.1.1 The T cell receptor

T cells express T cell receptors (TCRs) on their cell surface, which allow them to bind to specific peptide antigens – also referred to as *epitopes* – presented on other host cells via the major histocompatibility complex (MHC) [4] (Fig. 2.1). A unique TCR can only bind to a narrow range of possible peptide-major histocompatibility complexes (pMHCs), which are often referred to as the TCR's "cognate" targets. This specific binding between TCRs and pMHCs are what give T cells their target-specific abilities. Epitopes can be presented on either class I or class II MHCs, and which ones a TCR can interact with is determined by the phenotype of the T cell on which it is expressed [1] (see section 2.1.2).

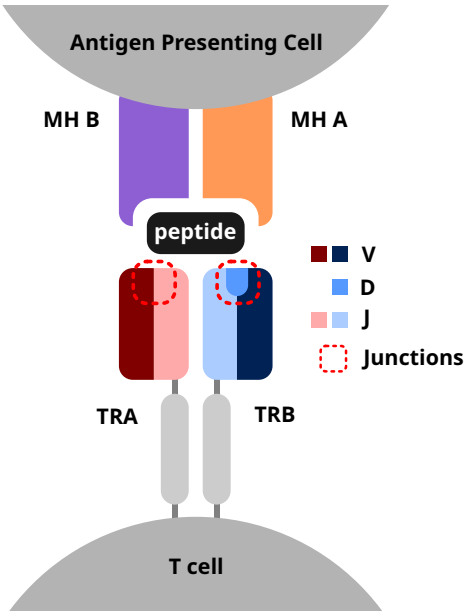


Figure 2.1: **A schematic of a TCR binding to a pMHC.** The T cell receptor (TCR) is a heterodimer, comprised of either an α and a β chain, or a γ and δ chain. The schematic above shows an $\alpha\beta$ TCR with its α chain marked as "TRA" and beta chain as "TRB". Each chain is comprised of a constant membrane region (bottom region shown in grey), which is spliced together with a variable binding domain (shown in red and blue). The binding domain itself is a result of splicing together a set of gene regions, called the V (variable), D (diversity, only in β and δ chains), and J (joining) regions. The boundary where the V, (D) and J regions join together is called the CDR3 or junction region (marked with a red dotted circle). The CDR3 region is structurally what tends to most directly contact the peptide epitope when a TCR engages with a pMHC. Adapted from my own publication [3] (licensed CCBY).

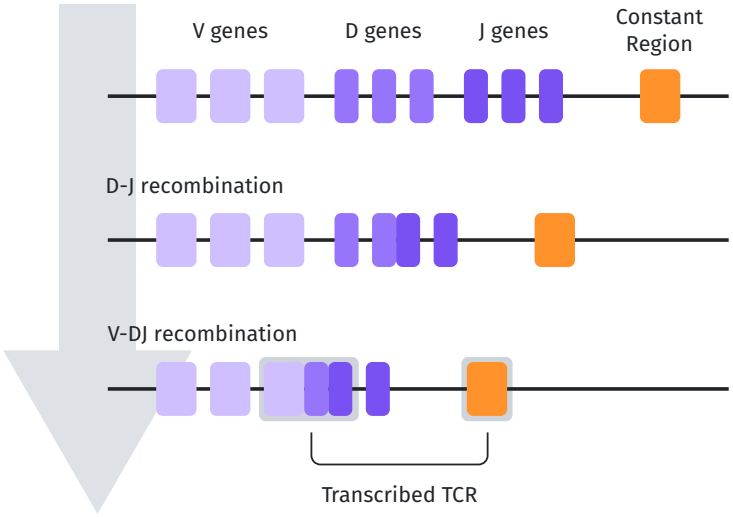


Figure 2.2: **A schematic of VDJ recombination.** The mammalian genome encodes a number of possible V (variable), D (diversity, only for β and δ chains) and J (joining) gene segments that can comprise a TCR chain. During VDJ recombination, one of each gene segments are selected stochastically and spliced together to form a complete TCR chain. The D and J segments are recombined first, although this only happens for the β and δ chains (the α and γ chains do not have a D region). This is followed by the recombination of the V and (D)J segments. The recombined VDJ exon together with a constant region is transcribed which leads to the expression of a full TCR chain.

TCRs are heterodimeric receptors [5]. They can either be composed of an α and β chain, or a γ and δ chain, although the vast majority of the body's T cells express $\alpha\beta$ TCRs. Unlike other proteins, the underlying nucleotide sequence of the TCR is not germline-encoded, and is instead produced through a stochastic process of somatic gene rearrangement called VDJ recombination (Fig. 2.2) [5, 6]. VDJ recombination occurs during the early stages of development for every new T cell generated by the body, which means that in general, the TCR amino acid sequence and hence target specificity is unique between different T cell clones.

During VDJ recombination, one each out of multiple available C (constant), V (variable), D (diversity, only for β and δ chains) and J (joining) gene segments are spliced together to generate both chains of the TCR [1, 7] (Fig. 2.2). In addition to the combinatorial variation resulting from the stochastic selection of the individual gene segments, the imprecise nature of the splicing process introduces somatic mutations at the junction re-

gions where different gene segments are glued together. This results in a staggering number of possible TCR amino acid sequences that can be produced, with estimates as high as $\sim 10^{60}$ [8]. In this way, the mammalian adaptive immune system has evolved to cover a broad range of possible antigens that can be presented via the pMHC.

There are three sites within the amino acid sequence of the TCR that are particularly variable across different receptors [1, 5]. These are referred to as the complementarity determining regions (CDRs). Each TCR chain has three CDRs, where the first two lie within the germline-encoded V gene, and the third is defined to be the junction region between the V, (D) and J regions. Due to its junctional nature, the third CDR is the most diverse. Structurally, the third CDRs of both TCR chains are also the regions of the receptor that most directly contact the target epitope when interacting with a pMHC (Fig. 2.1) [9].

2.1.2 T cell development and thymic selection

T cells are generated in the bone marrow [1]. However, they do not become fully functional until they are trafficked through the circulation to the thymus, where only a subset survive what is known as thymic selection. At a high level, this process selects for T cells with TCRs that fulfil the following criteria:

1. They must be functional (i.e. able to engage productively with pMHCs)

2. They must not be auto-reactive (i.e. not interact strongly with pMHCs that are presenting self-antigens)

This is achieved through the unique ability of thymic epithelial cells to express a wide variety of host proteins – even those reserved for other tissue types – and present the resulting self-peptides on their pMHCs [2]. T cells that do not interact sufficiently with these pMHCs (non-functional) are left to die, while those that interact too strongly (auto-reactive) are killed.

Also during thymic selection, T cells commit to one of two families of differentiation pathways by choosing to express either CD4 or CD8 molecules on their surface [1]. T cells expressing CD4 (sometimes referred to generally as "helper" T cells) are able to interact with class II pMHCs expressed on the surfaces of so-called professional antigen-presenting cells (APCs) like dendritic cells. These APCs act as hubs where many immune cells types can convene, and thus CD4+ T cells' interaction

with class II pMHCs allow them to act as facilitators and directors of the overall immune response. T cells expressing CD8 (sometimes referred to generally as "cytotoxic" T cells) are able to interact with class I pMHCs, which are expressed on all nucleated cells in the body. Most CD8+ T cells assume an effector role, and can detect and kill compromised host cells through cytotoxic attack. While their detailed exploration is outside the scope of this short review, it should be noted that the CD4+ and CD8+ T cell lineages can be subdivided into finer pathways of differentiation, and that there are some exceptions to the general model outlined above [1, 2] (e.g. CD4+/CD8+ regulatory T cells, natural killer T cells, $\gamma\delta$T cells, etc.).

2.1.3 The T cell receptor repertoire

The multiset of all TCRs that exist within a host's T cell population at a particular point in time is referred to as the host's *TCR repertoire* [2]. One useful and common way of thinking about the TCR repertoire is as a series of samples from some underlying probability distribution over the set of possible TCR sequences.

A widely accepted model for this underlying distribution is one where there are two major influencing factors [1, 2, 8, 10]. The first is the generative distribution associated with VDJ recombination, which is predictable and non-uniform over TCR sequence space [11]. The statistics of this process are largely universal with only minor variation between individuals [12]. The second factor is the clonal expansion of T cells in response to immune stimuli. That is, when there is an immune signal – for example due to an infection – T cells with TCRs that recognise the invading pathogen through its epitopes will become activated, causing them to proliferate and thus increasing their observed frequencies.

In contrast to TCR generation probabilities, the effects of immune exposures to the TCR repertoire is dynamic. The binding specificity of TCRs to pMHCs mean that the set of TCRs whose frequencies increase as a result of an immune stimulus is a function of the epitopes produced by the source of the stimulus. Therefore, how this process shapes a host's TCR repertoire depends on *what* stimulates the immune system and *when*. Another nuanced but important thing to note here is that the gene loci coding for the proteins comprising the MHC are highly polymorphic between individuals [1, 6]. This means that in general, different individuals will produce slightly different MHC molecules with distinct binding pockets, and thus the same antigens will produce different pMHCs. Therefore, how immune stimuli shape a host's repertoire is also a function of what

MHC alleles they carry, or their *MHC restriction*.

While being temporally dynamic, the effects of immune stimuli on the repertoire are not entirely transient [1]. When the source of an immune stimulus is successfully cleared from the host, most of the expanded T cells die. However, a subset of the responding cells – called *memory* T cells – persist in the repertoire. Therefore, a host's TCR repertoire is a mixture of TCRs on *naïve* T cells that have not yet encountered their cognate targets, potentially those on currently active T cells responding to an ongoing stimulus, and finally those on memory T cells that remain from previous exposures.

This type of model of the TCR repertoire have led to hypotheses that sequencing TCRs from the peripheral blood of patients can provide a minimally invasive method to profile the current state of the patients' immune system, and even detect disease [13–18]. For example, if one can develop a general understanding of what lung cancer-responsive TCRs look like, their presence in a patient's peripheral blood can be used as a biomarker for disease. An alternative approach to hunting for specific TCRs of interest might be to use more global measures of TCR repertoire state, via diversity indices or by quantifying the degree of clonal TCR expansion seen [19]. There is some speculation that the exponential nature of the initial T cell response against immune stimuli may make the TCR repertoire a particularly sensitive biomarker for early detection of disease. However, it is not yet clear to what extent the effects of initial T cell responses, which are usually tissue-restricted to the site of infection or cancer, are visible in the peripheral blood.

2.2 The TCR-pMHC problem

The rules governing which TCRs can bind which pMHCs is an unsolved grand challenge in systems immunology, which if we can crack, has significant potential for positive impact in healthcare [20]. From the therapeutic standpoint, it may lead to better-informed designs of vaccines (particularly for those that must induce cellular responses) as well as cellular immunotherapies. From the diagnostic standpoint, it may provide a minimally invasive medium through which to rapidly screen for a wide variety of diseases with immune correlates, such as infections, autoimmune conditions, and cancer. However, several factors make this a challenging task.

Firstly, TCR-pMHC binding is promiscuous. The number of unique TCRs that can exist in a human host at one moment in time ($\sim 10^6 -$

10^{10}) [20, 21] is dwarfed by the number of possible pMHCs that the immune system must be prepared to respond against ($\sim 10^{15}$) [22]. Thus, the physiological dynamics of TCR-pMHC interaction must be that a unique TCR can respond to more than one cognate pMHC. Similarly, the size of a human TCR repertoire is also miniscule compared to the universe of possible TCR sequences ($\sim 10^{60}$) [8], which makes seeing the same TCR in the repertoires of two distinct hosts extremely rare. Therefore, it is likely that multiple TCRs are able to respond to the same pMHC. Indeed, we see empirically that many different TCRs are able to interact strongly with the same pMHC, and vice versa [23–25]. This suggests a fuzzy and complex many-to-many mapping between the set of possible TCRs and pMHCs.

The second and related issue is the vast size of the problem space ($\sim 10^{75}$ receptor-ligand pairs). Although advances in high throughput assays of T cell function have accelerated data generation of interacting TCR-pMHC pairs, an exhaustive empirical survey of all possible pairs remains implausible. For this reason, there has been growing interest in applying machine learning to gain inroads into the problem [20, 26]. A brief review of existing machine learning work on TCR specificity prediction can be found in section 2.6.

2.3 T cell receptor coincidence statistics

As discussed above, one major factor which makes learning the TCR-pMHC binding rules challenging is the vastness of the problem space. In particular, only a minute proportion of the total possible diversity of pMHCs is explored in the specificity-annotated TCR data that is currently available. This poses a significant challenge, as the primary interest of the field is to discover binding rules that generalise across different pMHCs. One way to mitigate this problem as proposed by Mayer and Callan is to focus on the two-point statistics of TCRs [22]. By two-point statistics, I refer to the statistics of drawing *pairs* of TCRs from a distribution, as opposed to drawing individual TCR instances. I introduce the reader to this framework here, as it has played a significant role in motivating and shaping my own PhD work presented in the later chapters, and it will be helpful to understand it beforehand.

2.3.1 Intuition and motivation

What exactly do we mean by two-point statistics, and why would we be interested in it for quantitative TCR analysis? To make this idea clearer, let us build a mental model around TCR specificity inference from the statistical point of view. Let $\mathcal{T}$ and $\mathcal{P}$ be the set of possible TCRs and pMHCs respectively. Let $p_B(\tau), \tau \in \mathcal{T}$ be the background distribution of TCR sequences from VDJ recombination. Finally, let $p_{S|\Pi}(\tau|\pi), \tau \in \mathcal{T}, \pi \in \mathcal{P}$ be the distribution of specific binder TCRs against some pMHC π.

One of the end goals of quantitative TCR analysis is to be able to accurately predict TCR specificity against any arbitrary pMHC. At a high level, this means accurately approximating $p_{S|\Pi}(\cdot|\pi)$ for arbitrary π. Now, we see from empirical evidence that different pMHCs in general select for different-looking TCRs. That is, the locations of high density of $p_{S|\Pi}(\cdot|\pi)$ in $\mathcal{T}$ vary with different π. Therefore, to get a generalisable approximation of $p_{S|\Pi}$, one would need specificity-annotated TCR data such that:

1. A reasonable variety of different pMHCs specificities are covered
2. Each such pMHC has a sufficient number of binder TCRs sampled

Unfortunately, there is a general agreement that current limitations in data availability mean that these requirements are not yet met.

Now, let us turn to two-point statistics. Let d be some metric over $\mathcal{T}$ that measures a distance between any two TCRs– Mayer and Callan choose to investigate the Levenshtein distance between their amino acid sequences. Mayer and Callan propose a quantity they call the "near-coincidence probability" $p_C[\cdot](\delta)$, which is a functional of some given probability distribution p over TCRs:

$$p_C[p](\delta) \coloneqq \sum_{\tau \in \mathcal{T}} \sum_{\tau' \in \mathcal{T}} p(\tau) p(\tau') I_{d(\tau,\tau')=\delta} \tag{2.1}$$

Here, $I_{d(\tau,\tau')=\delta}$ is an indicator function that evaluates to one when $d(\tau, \tau') = \delta$, and zero otherwise[1]. $p_C[p](\delta)$ quantifies the probability that two independent draws from p result in a pair of TCRs that are a given distance δ away from one another according to d. Put simply, $p_c[p](\delta)$ is the distribution of *distances* between TCRs sampled from p.

[1]Mayer and Callan specifically investigate Levenshtein distance, which is a discrete distance measure. This makes $d(\tau, \tau') = \delta$ a well-defined event. However, the eagle-eyed reader may notice that this setup no longer works for continuous metrics. In such cases, one may consider a definition of near-coincidence based on the inequality $d(\tau, \tau') < \delta$.

Since co-specific TCRs tend to have similar sequences, we might expect $p_C[p_{S|\Pi}](\delta)$ to skew towards smaller values compared to $p_C[p_B](\delta)$. If we could accurately estimate these two distance distributions, we would be able to quantify such a skew, and thus have a principled framework through which to make inferences about the probability that two TCRs are co-specific given that they are at some distance away from one another.

By turning to the two-point distance statistics, the objective has shifted from the challenge of estimating $p_{S|\Pi}(\tau|\pi)$ for all $\mathcal{T} \times \mathcal{P}$, to the easier task of estimating $p_C[p_B](\delta)$ and $p_C[p_{S|\Pi}](\delta)$ for δ in the range of d ($\mathbb{Z}^+$ for Levenshtein distance). There are a couple of reasons why I claim the latter estimations are easier. Firstly, the pairwise nature of the problem means that from N samples of TCRs, you can generate $\binom{N}{2} \sim O(N^2)$ TCR pairs, giving you more empirical data points to make estimations from. Secondly, we made the observation earlier that the *locations* of high density for $p_{S|\Pi}$ vary when considering different pMHC specificities. However, as long as the *shapes* of $p_{S|\Pi}$ stay roughly similar – that is, the local co-specificity rules stay consistent – across pMHCs, $p_C[p_{S|\Pi}(\cdot|\pi)]$ should also behave similarly between different π. If so, we depend less on having specificity-annotated TCR data against many different pMHCs in order to learn something useful about local co-specificity rules, and indeed to estimate a general:

$$p_C[p_{S|\Pi}] \coloneqq \mathop{\mathbb{E}}_{\pi \in \mathcal{P}} \Big\{ p_C[p_{S|\Pi}(\cdot|\pi)] \Big\} \tag{2.2}$$

2.3.2 Hints of generalisable co-specificity rules

Mayer and Callan used publicly available TCR data to estimate $p_C[p_B](\delta)$ and $p_C[p_{S|\Pi}](\delta)$ and revealed that there is a large degree of enrichment for both exact and near coincidences in epitope-specific repertoires compared to background [22]. They show across different pMHCs and datasets that the profiles of the enrichment ratio $p_C[p_{S|\Pi}](\delta)/p_C[p_B](\delta)$ consistently display an exponential falloff to some plateau value with a decay rate (with respect to Levenshtein distance on the CDR3 sequences) around 10 (Fig. 2.3). In summary, the study of TCR coincidence statistics provides a principled framework through which to quantify the notion that TCR pairs with similar sequences are likely to recognise similar sets of pMHCs. Mayer and Callan's results provide evidence that while we may not yet have enough data to predict the locations of binding solutions against

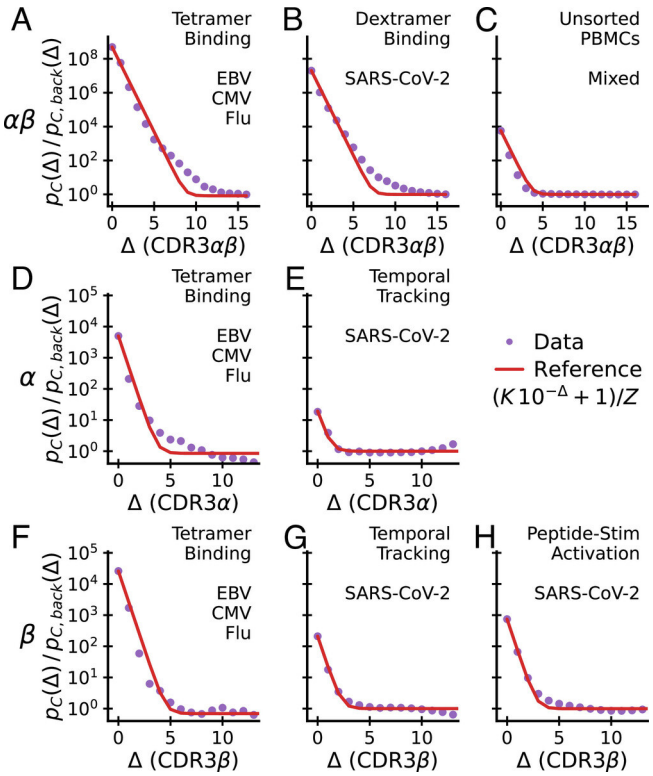


Figure 2.3: **Near-coincidence enrichment ratios between various specificity-restricted repertoires and background consistently show exponential decay to a plateau.** The y axes of all subplots show the near-coincidence probability enrichment ratio $p_C[p_{S|II}](\delta)/p_C[p_B](\delta)$ in log base 10 scale. The distance between the TCR pairs vary along the x axes. The top row shows the ratio calculated where distances are calculated between the CDR3 amino acid sequences of both α and β chains. The middle and bottom rows show similar plots only measuring distance between the α and β CDR3 sequences respectively. Each subplot is labelled in its top right hand corner with the assay method used to obtain the specificity-restricted repertoire data, and beneath this text is a list of species which would have produced the epitopes against which the TCRs in the specificity-restricted repertoire are responding. The purple points show the empirical estimates, overlaid with a red line showing a model fit of $(K10^{-\Delta}+1)/Z$. Such a model with an exponential decay rate of 10 with respect to Levenshtein edit distance seem to fit well with all the profiles shown. Reprinted with permission from [22] (licensed CCBY).

arbitrary pMHCs, the local co-specificity rules active around such binding solutions are learnable and generalisable across pMHCs.

2.4 Transformer neural networks

Later in chapter 4, I discuss my work on a TCR language model with a transformer neural network architecture. Here I provide the reader with an introduction to the architecture, its original motivation, a high-level overview of how it works, and some historical context.

2.4.1 How does it work?

The transformer is a neural network architecture originally developed for machine translation in the field of natural language processing (NLP) [27]. It is composed of an *encoder network*, which takes a piece of text in the model's input language and maps it to a series of numerical vectors that represent its semantic content, and a *decoder network*, which uses these numerical vectors to generate a corresponding text in the model's target language that best emulates the meaning of the original input text. An overview schematic from the original transformer publication [27] is shown in figure 2.4. Focused schematics for the encoder and decoder components are shown in figures 2.5 and 2.7 respectively.

The encoder

The transformer encoder's job is to take a piece of text in the model's input language, and map it to a set of vectors that encode its semantic content [27] (Fig. 2.5). Let us go over step by step how this mapping occurs.

Any input text provided to the transformer will be in its raw string form, which to the computer is just a sequence of individual characters. This must first be processed by a module within the encoder called the *tokeniser*, which splits the string into short and meaningful segments called tokens. A token is usually a word or a word-piece (e.g. common prefixes or suffixes). Earlier tokeniser modules used simple rules such as splitting on whitespace characters, but this approach has been made obsolete by more intelligent algorithms that can learn to tokenise languages that do not use white space [28].

Once the input text is tokenised, an *embedding* module is used to map each token to a vector representation of itself. One can think of the em-

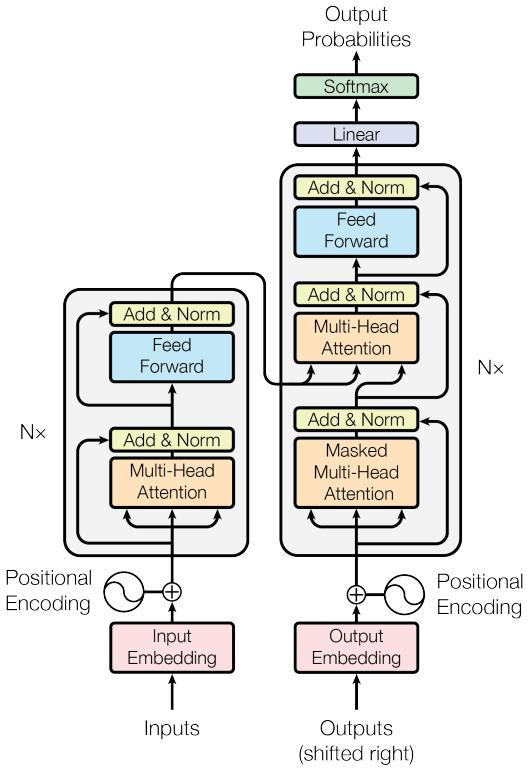


Figure 2.4: **A schematic of the transformer encoder-decoder neural network architecture as originally proposed by Vaswani et al.** The original transformer architecture was designed for machine translation, and is comprised of an *encoder network* (left) and a *decoder network* (right). The input to the transformer network is text in the model's input language, which is split into tokens (usually words, or word-pieces like common prefixes or suffixes) by the model's *tokeniser module* (Fig. 2.5). Then, these tokens are passed to the input language *embedding module*, which maps each token to a learned vector representation that capture the semantic information contained in the token. After being infused with an explicit encoding of their position within the input sequence, the embedded token vectors are passed to a stack of multi-head attention networks (Fig. 2.6) that adjust the input token embeddings such that their representation of the tokens' semantic content takes into account the rest of the input text as context. The decoder network then uses these improved input token embeddings as its guiding context to autoregressively generate a token sequence in the model's target language, completing the translation (Fig. 2.7). Taken from [27] with permission from Google.

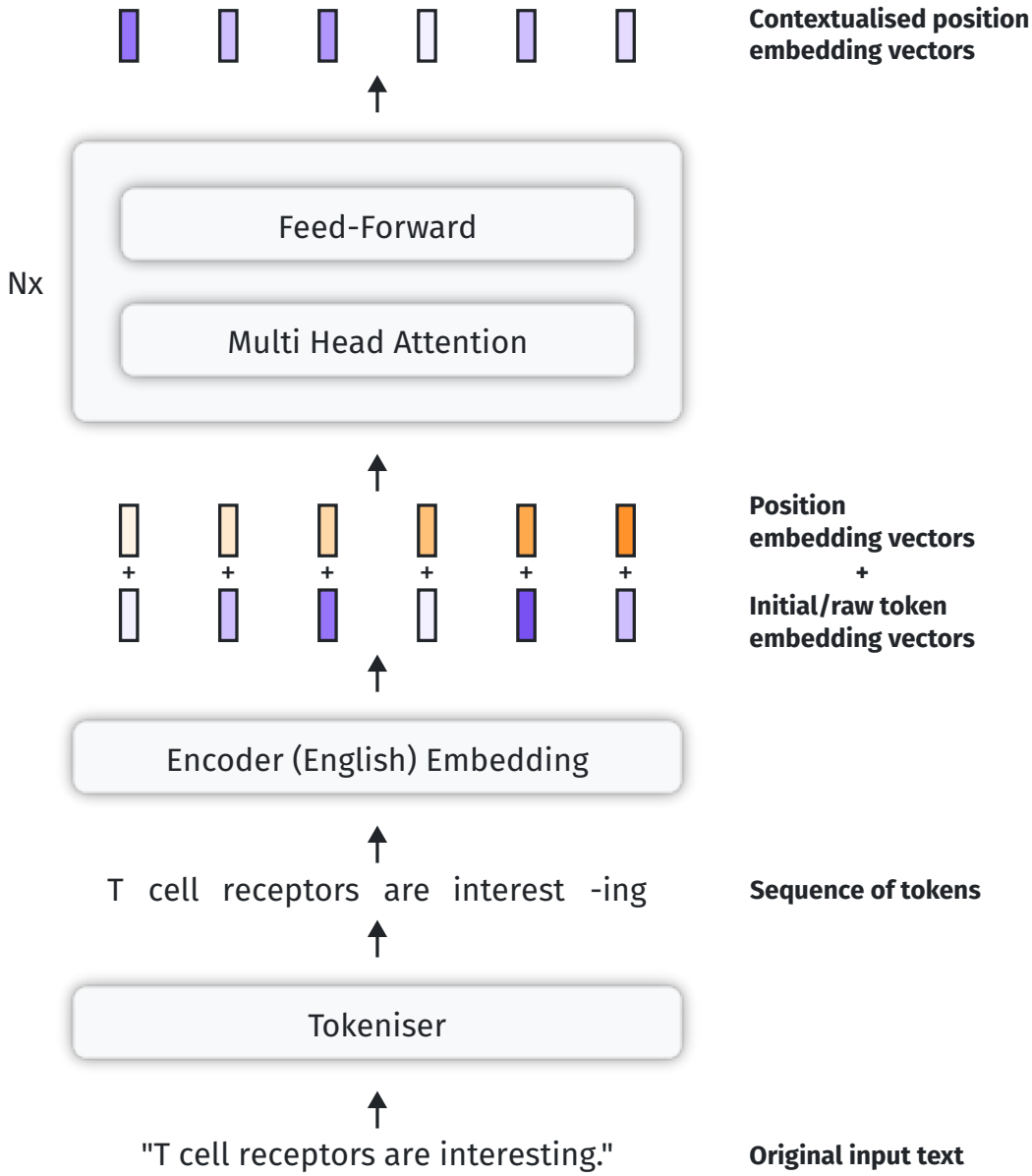


Figure 2.5: **A simplified schematic of the transformer encoder.** Input text is first segmented into meaningful subunits by the tokeniser module. Then, each token is mapped to an initial non-contextualised vector representation through the embedding module. These are infused with positional information through a summation with positional vector embeddings, and then passed through a stack of MHA and feed-forward layers. The job of the MHA layers are to intelligently adjust the vector embeddings of the various tokens such that they reflect their semantic content, in context of the rest of the input sequence.

bedding module as a large look-up table that stores a vector representation for every possible token. Of course, the embedding module has a finite capacity, and so the set of tokens to be accounted for must be set beforehand. A common practice has been to tabulate all possible tokens seen in the training corpus for the model's input language, and to use this for setting the embedding module's capacity. Sometimes, a special `<unk>` (for "unknown") token is also registered in case an unseen word is encountered at deployment time[2]. The vector representations for each token is randomly initialised at first, and are learnt together with the rest of the network parameters through gradient descent. The idea is that these learned vector representations, or embeddings, should converge to encode the typical "meanings" of the tokens.

The token embeddings are then summed with *position embeddings*, which are vectors that encode information on the position of a token within the sequence. Such explicit encodings of token positions are necessary when using transformers, since the transformer architecture is such that its output will be token order-invariant without them (as you will see in a moment). In their original publication, Vaswani et al. generated position embeddings by computing a vector whose components evaluated to sinusoidal functions of the token's position index, where the period of the sinusoid increased geometrically with each subsequent component.

Finally, these token embeddings are operated on by a stack of multi-head attention (MHA) layers. The design of these MHA networks is the primary innovation of the transformer. They model the intuitive idea that the function or meaning of a token in a natural language sequence is often influenced by its surrounding context. For instance, consider the word "cell" in the example sentence used in figure 2.5. Without any context, its meaning is ambiguous – "cell" could refer to a confined room, or perhaps a battery. It is only after considering the rest of the sentence that it becomes clear: here, "cell" should represent the idea of the smallest unit of a living system. MHA networks can learn to adaptively and intelligently adjust the values of the token embeddings, such that they better convey their *contextualised* meanings. These contextualised embeddings are the final output of the transformer encoder network.

How do the MHA networks compute such contextualised embeddings? Let us start with the high-level intuition. First, they use a learned rule-set to estimate the interaction strength (often referred to as "atten-

[2]While use of an `<unk>` token was common in the past, modern tokenisation methods [28] have made this largely unnecessary since they reserve each character/alphabet in their input language as a token and thus guarantees that any word be tokenisable.

tion" in the NLP literature) between every token pair in the input sequence. Then, for each token, the distribution of attention scores between it and all input tokens (including itself) is used to compute a weighted sum over the original token embeddings. This weighted sum becomes the token's new contextualised representation. In transformer models, layers of MHA networks are often chained in sequence such that the output of one feeds into the next, and thus the process of token contextualisation is done in increments. As an additional comment, the MHA layers in the transformer encoder are also referred to as *self-attention* networks, because they compute attention scores *within* tokens of the input sequence.

Now let us examine how each MHA layer implements this computation. Vaswani et al. propose a mechanism called *scaled dot-product attention*, where the attention between two tokens is computed as the dot product between (transformations of) their representations. Let $\mathbf{X} \in \mathbb{R}^{d \times M} = [\mathbf{x}_i]_{i=1}^{M}$ be the input to an MHA layer, which we represent as a matrix whose column vectors are the d-dimensional token embeddings of an input sequence with M tokens. In the simplest case, an MHA layer implements three learnable affine transformations[3]:

$$\begin{aligned} q &: \mathbb{R}^d \to \mathbb{R}^{d_k} && \text{(q for query)} \\ k &: \mathbb{R}^d \to \mathbb{R}^{d_k} && \text{(k for key)} \\ v &: \mathbb{R}^d \to \mathbb{R}^{d_v} && \text{(v for value)} \end{aligned}$$

Let $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ represent the matrices whose ith column vectors are $q(\mathbf{x}_i)$, $k(\mathbf{x}_i)$ and $v(\mathbf{x}_i)$ respectively. Then, the output of this MHA layer can be described as follows:

$$\text{MHA}(\mathbf{X}) := \mathbf{V} \times \text{softmax}\left(\frac{\mathbf{Q}^\top \times \mathbf{K}}{\sqrt{d_k}}\right)^\top \tag{2.3}$$

where:

$$\text{softmax}(\mathbf{A})_{ij} := \frac{\exp \mathbf{A}_{ij}}{\sum_{k=1}^{n} \exp \mathbf{A}_{ik}}, \quad \mathbf{A}, \text{softmax}(\mathbf{A}) \in \mathbb{R}^{m \times n} \tag{2.4}$$

Can we interpret what is happening in equation 2.3? The purple portion corresponds to the MHA layer using the dot product $q(\mathbf{x}_i)^\top k(\mathbf{x}_j)$ to compute the attention score between the tokens at indices i and j (note that the presence of the q and k transformations make this computation

[3] An affine transformation is that of the form $f(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$, where the weight matrix $\mathbf{W}$ and bias vector $\mathbf{b}$ are learnable.

asymmetric and thus give attention scores directionality). These attention scores are weighted by the square root of d_k to promote gradient stability when training. The orange portion corresponds to how all the attention scores from the same token are normalised with respect to each other (in this matrix formulation, this is row-wise) using the softmax function to produce a probability distribution over tokens. Finally, the green portion corresponds to computing the weighted sum of the v transformations of the token embeddings according to the probability distributions calculated in the previous step, to produce an updated embedding for each token.

I mentioned earlier that my description above represents the simplest case. This is because in reality, what equation 2.3 describes is the computation of one *attention head*. As its name suggests, a *multi-head attention* layer can have more than one attention head, each with its own learnt q, k and v projections. In an MHA layer with N_h heads, the outputs of all attention heads are computed in parallel and then concatenated to produce the overall output of the MHA layer: $\mathbf{X}^* \in \mathbb{R}^{N_h d_v \times M}$. It is common to set the dimensionality d_v of the attention head outputs to d/N_h, such that the dimensionality of the layer's final output embeddings remains the same as that of the input.

Before the output embeddings of one MHA layer reach the next MHA layer in the stack, they are passed through a *feed-forward* layer, which is simply a composition of two affine transformations: the first from $\mathbb{R}^d$ to a higher-dimensional space $\mathbb{R}^{d_{\text{ff}}}, d_{\text{ff}} > d$, and the second back to $\mathbb{R}^d$. A rectified linear unit (ReLU) activation function is put in between the two affine projections, such that the overall computation of a feed-forward layer is:

$$FF(\mathbf{X}) \coloneqq \mathbf{W}_2 \text{ReLU}(\mathbf{W}_1 \mathbf{X} + \mathbf{b}_1) + \mathbf{b}_2 \tag{2.5}$$

where:

$$\text{ReLU}(\mathbf{x})_i \coloneqq \max(0, \mathbf{x}_i) \tag{2.6}$$

$$\mathbf{W}_1 \in \mathbb{R}^{d_{\text{ff}} \times d}, \mathbf{W}_2 \in \mathbb{R}^{d \times d_{\text{ff}}}, \mathbf{b}_1 \in \mathbb{R}^{d_{\text{ff}}}, \mathbf{b}_2 \in \mathbb{R}^d \tag{2.7}$$

The presence of these feed-forward networks provide the transformer with additional flexibility.

The decoder

The responsibility of a transformer decoder is to take the numerical representation of the input text generated by the encoder, and translate it to a sequence of tokens (and hence text) in the target language [27] (Fig. 2.7).

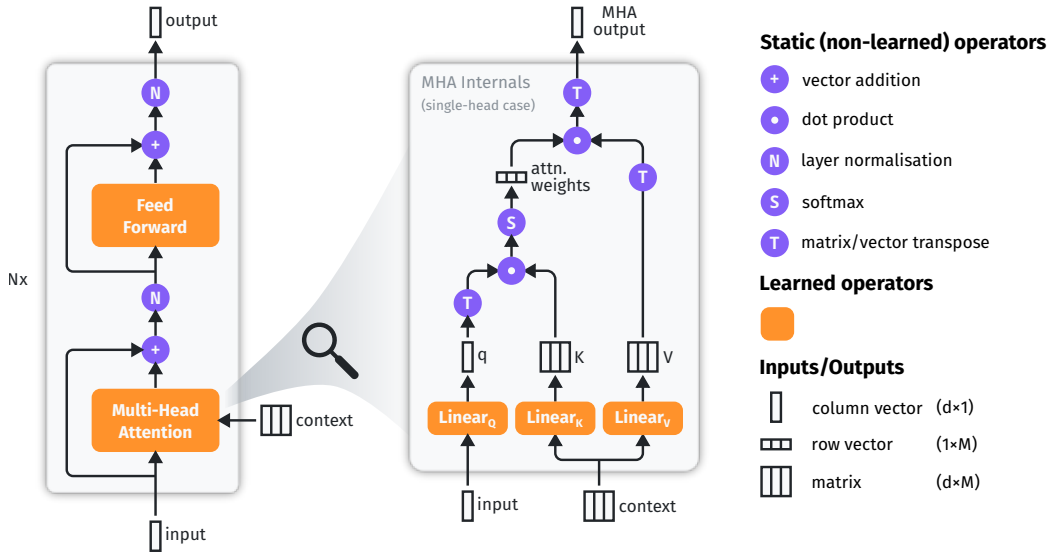


Figure 2.6: **A simplified schematic depicting the internals of the transformer multi-head attention stack.** The schematic is accurate for the case of a single attention head per layer. In the more general case of N_h attention heads, each MHA block will have N_h parallel q, k and v linear projections, each from dimensionality d to dimensionality d/N_h. Each parallel set of q, k and v vectors/matrices undergo the series of operations shown in the schematic. Finally, the final output vector (of shape $d/N_h \times 1$) from each parallel branch are concatenated together to produce the output of the MHA block (of shape $d \times 1$). Taken from my own preprint, [29] (licensed CCBY).

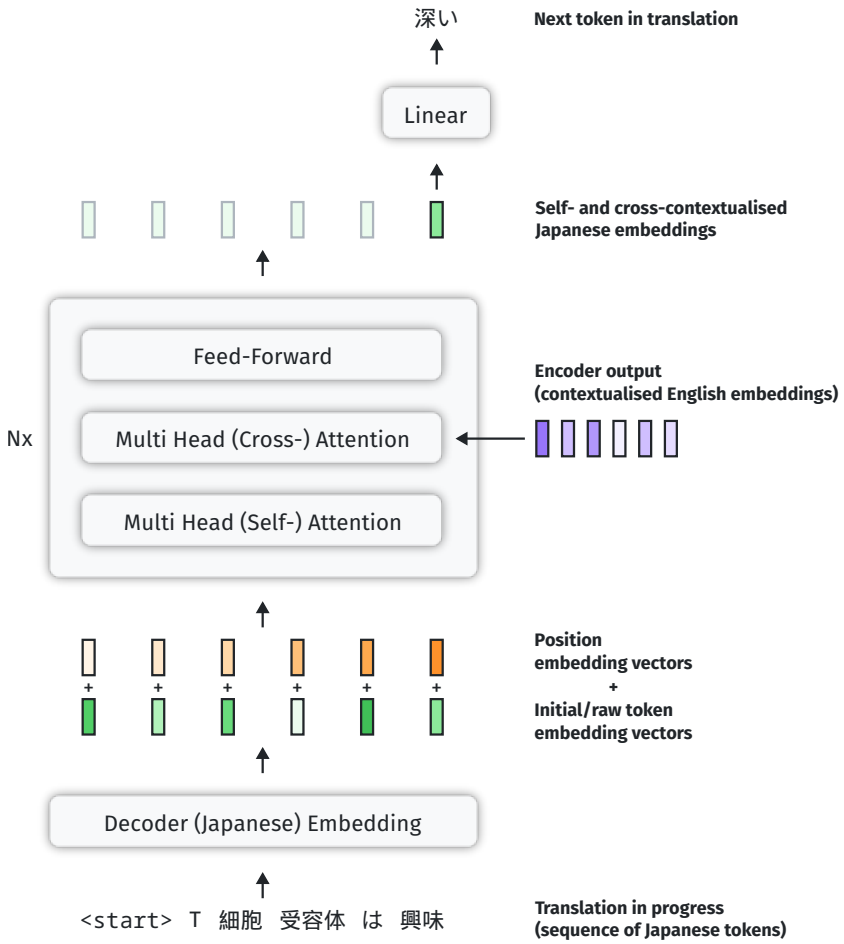


Figure 2.7: **A simplified schematic of the transformer decoder.** The decoder network generates a translation of the input text in an auto-regressive manner. Given an incomplete translation, the model predicts the next token. All model translations are started with the `<start>` token, which allows the model to predict the first real token of the translation. The model is applied recursively to iteratively construct the full translation. This is halted when the model predicts the `<end>` token, which signifies that the translation is complete. The decoder works by first mapping the tokens of the current translation in progress to a set of embeddings, then infusing the embeddings with positional information using position embedding vectors, and passing the resulting initial embeddings through a stack of MHA and feed-forward layers that contextualise the token embeddings not only with respect to themselves, but also to the output embeddings of the encoder. Finally, the contextualised embedding for the final token in the current translation sequence is pushed through a linear layer with softmax activation to predict the next token.

Like the encoder, the decoder also has an embedding module. This time, the embedding module contains mappings from tokens in the target language to their initial embedding vectors. It also registers additional special tokens called <start> and <end> which represent the beginning and end of a translated sequence, respectively. Why these are necessary will become clear in a moment.

The primary portion of the decoder is also a stack of MHA layers and feed-forward layers, but they function in a slightly different way. We will again first start with the intuition. Unlike the encoder whose job is to process incoming data, the decoder must *generate* a new sequence. This generative process is modelled by training the decoder to predict the next token in an unfinished translation. This framework, combined with the assertion that all translations must begin with the <start> token, allows the transformer decoder to use the starter-sequence [<start>] to predict the first token of the translated sequence, append its own prediction to the end of the current translation sequence, then use this to predict the second token, and so on – slowly building up the translation one token at a time. This process of next-token-prediction continues until the predicted token is the <end> token, which signals that the translation is over.

Now, let us follow step-by-step the computations of the decoder network. First, the decoder embedding module is used to map the current translation sequence into a sequence of embedding vectors (in the beginning, this will be of length one, and only contain the <start> token). This sequence of vectors is then passed to a stack of MHA and feed-forward layers.

Unlike the encoder, each repeating unit of the transformer decoder has three layers. The first is a self-attention MHA network, and it functions much like those discussed previously for the encoder. That is, its job is to contextualise the token embeddings of the translation sequence with respect to one another.

The second second layer is also an MHA network – but in contrast to the self-attention MHA layers discussed until now, this one uses the token embeddings from the current state of the translation to generate the $\mathbf{Q}$ matrix, while using the *output of the encoder network* to generate the $\mathbf{K}$ and $\mathbf{V}$ matrices. Put into words, the second MHA layer updates the embeddings of the tokens in the translation sequence by computing a weighted sum over the embeddings of the model's input in the *original language*. Since this second MHA layer computes attention across the input sequence and the generated output translation sequence, it is often called the *cross-attention* layer. The idea is that the cross-attention layer helps the model align its translation to the semantic content of its input.

Finally, the output of the cross-attention layer is passed through a feed-forward layer. This concludes the computation of one repeatable element of the decoder stack.

From the output of the final layer of the decoder stack, the column vector corresponding to the final token in the current translation sequence is fed through an affine projection to $\mathbb{R}^{N_t}$ followed by a softmax activation, where N_t is the number of tokens in the target language. This generates a probability distribution over possible tokens which can be used to predict the identity of the next token in the translation. If the predicted identity of the next token is `<end>`, the translation is halted and the resulting token sequence in the target language is used as the model's output translation. With this, we have gone over all computations inside of the transformer architecture. The curious reader is referred to the original publication for further details of the model architecture [27].

2.4.2 Unsupervised pre-training of transformers

In Vaswani et al.'s original publication, the transformer encoder-decoder stack needed to be trained end-to-end using labelled data with paired translations between the target languages. However the volume of available human-translated paired sentence datasets quickly became a bottleneck for training. Later works addressed this challenge by proposing methods of unsupervised pre-training, both for the encoder [30] as well as the decoder [31].

On the encoder side, Devlin et al. proposed Bidirectional Encoder Representations from Transformers (BERT) [30]. Devlin et al.'s key innovation was an unsupervised pre-training strategy called masked-language modelling (MLM), where a transformer encoder would be given a real token sequence with the identity of a fixed proportion of tokens hidden, or "masked". The objective of the model would be to predict back the correct identities of the masked tokens, based on visible portion of the input as context. At a high level, this objective encourages the model to learn any contextual interactions that may exist between tokens of the input language, while also precluding the requirement for any human-translated text. In this way, MLM allowed Devlin et al. to train BERT, a large transformer encoder network, in a completely unsupervised manner. BERT and its derivative models have been able to achieve high levels of performance on various downstream textual tasks with relatively little supervised fine-tuning on labelled data. Furthermore, the encoder-only nature of the BERT architecture has made it useful for text processing

tasks outside of machine translation, most notably for text representation and classification. There is growing interest in applying BERT-like transformer encoders to protein analysis, where they are pre-trained on polypeptide sequences instead of natural language. My work in chapter 4 is one example of this. Other works applying this technology to TCR analysis are mentioned in section 2.6.

On the generative decoder transformer side, Radford et al. proposed the generative pre-trained transformer (GPT) [31]. They demonstrated that an unsupervised pre-training strategy where the model is fed the beginning of a text and trained to maximise the conditional probability of generating the correct subsequent token sequence results in a generative language model that is highly capable of intelligent behaviour, particularly with interactive question answering. As with BERT, fine-tuned GPTs have proven to be generally useful for interactive text generation outside of machine translation, and now form the backbone of the latest language-model-based AI assistant services including OpenAI's ChatGPT [32], Anthropic's Claude [33], and Google's Gemini [34].

2.5 Measures of model performance

In machine learning, the act of evaluating model performance is just as crucial as the act of training the models. In this section, I introduce the reader to some common measures of model performance often seen in the machine learning literature. I focus on measures of binary classification performance, both due to their popularity in use and their relevance to my own work shown in the later chapters of this thesis.

2.5.1 The confusion matrix

One simple way to summarise the statistics of the performance of a binary classifier is by building a 2×2 *confusion matrix* [35, 36] (Fig. 2.8). The two rows of the matrix correspond to the statistics of the ground truth label, with the top row assigned to the *positive* case, and the bottom row to the *negative* case. The two columns correspond to the statistics of labels as predicted by the binary classification model of interest, with the left column for positives, and the right column for negatives. We can now fill each cell of the matrix with the joint probability of obtaining the ground truth and prediction labels associated with it. For example, the top left cell corresponds to the joint probability of obtaining a true positive (TP) sample (i.e. ground truth positive, predicted positive). Going across from

True Label X \ Predicted Label Y	Pos.	Neg.
Pos.	$P(X{=}\text{Pos},Y{=}\text{Pos})$ **True Positives (TP)**	$P(X{=}\text{Pos},Y{=}\text{Neg})$ **False Negatives (FN)**
Neg.	$P(X{=}\text{Neg},Y{=}\text{Pos})$ **False Positives (FP)**	$P(X{=}\text{Neg},Y{=}\text{Neg})$ **True Negatives (TN)**

Figure 2.8: **The confusion matrix.** The confusion matrix is a table whose entries are the joint probabilities of different combinations of true and predicted binary labels. It is a useful tool through which to understand various performance metrics of binary classifiers.

the top left to the bottom right, we will refer to the cells in the matrix as TP, false positive (FP), true negative (TN), and false negative (FN).

Many useful metrics of binary classification performance can be calculated based on the values in such a confusion matrix. Here, I will introduce the four arguably most commonly used [35]:

1. Sensitivity
2. Specificity
3. Positive predictive value (PPV)
4. Negative predictive value (NPV)

Sensitivity and specificity

The *sensitivity* (also known as *recall*) of a binary classifier is the conditional probability of a positive model prediction given a positive ground truth label. Its complement is the *specificity*, which is the conditional probability of a negative prediction given a negative ground truth label. Their values can be respectively obtained from the cells in the confusion matrix as follows.

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$
$$\text{Specificity} = \frac{TN}{FP + TN}$$

Note that both sensitivity and specificity integrate out the statistics of the ground truth label through conditioning, and purely describe the behaviour of label predictions. This makes both of these measures an intrinsic property of a given binary classifier. For this reason, they are very popular and often used in machine learning and statistics.

Positive and negative predictive values

The *PPV* (also known as *precision*) of a binary classifier is the conditional probability of a positive ground truth label given a positive prediction. Its complement, the *NPV*, is the conditional probability of a negative ground truth label given a negative prediction. They can be obtained from the confusion matrix as follows.

$$\text{PPV} = \frac{TP}{TP + FP}$$
$$\text{NPV} = \frac{TN}{FN + TN}$$

Note how in contrast to sensitivity and specificity, PPV and NPV are dependent on the marginal probability of a positive ground truth label, sometimes referred to as the *prevalence* ($TP + FN$). To see this more clearly, we can for example re-express PPV as the following:

$$\begin{aligned}\text{PPV} &= \frac{\frac{TP}{TP+FN}(TP + FN)}{\frac{TP}{TP+FN}(TP + FN) + \frac{FP}{FP+TN}(FP + TN)} \\ &= \frac{\text{Sensitivity} \times \text{Prevalence}}{\text{Sensitivity} \times \text{Prevalence} + (1 - \text{Specificity}) \times (1 - \text{Prevalence})}\end{aligned}$$

A similar rearrangement can be done for the NPV as well. The consequence is that even the same model with the same sensitivity and specificity can produce very different values for PPV and NPV under different prevalences [35] (Fig. 2.9). That being said, in cases where the prevalence is known, the PPV and NPV are directly useful measures of how one's beliefs about the potential label of an unknown sample should update depending on an outcome of the model prediction. For this reason, these values are also commonly used.

2.5.2 Performance curves

In many cases, the raw output of a model underlying a binary predictor is a scalar value. Binary predictions are then obtained from such models by

picking a threshold value such that whenever the model output is above this threshold, the model prediction is interpreted as being positive, and vice versa. But in such situations, any of the four performance measures discussed in the previous section may change depending on the choice of the threshold. This becomes particularly clear when we consider the most extreme thresholds. For example, if the threshold were set to the lowest possible value for the model output, the resulting model prediction would always be positive. Thus the model's sensitivity would be one, and the specificity would be zero. A threshold on the other extreme would in contrast lead to a sensitivity of zero, and specificity of one.

Threshold in between interpolate between such extremes, and in fact, the nature of this interpolation contains valuable information about the "goodness" of a model. This makes intuitive sense – if one were to slowly increase the threshold from its lowest extreme, an ideal model's specificity should quickly shoot up to one, while the sensitivity should remain high and close to one as well. The nature of such trade-offs can be visualised and quantified by plotting how these performance measures change as the threshold is varied from one extreme to the other.

Two very commonly used examples of such performance curves are the receiver operating characteristic (ROC) and precision-recall (PR) curves [36, 37]. The ROC curve plots the trade-off between sensitivity (usually on the y axis) and specificity as a model's threshold is varied, while the PR curve plots the change in PPV (precision, usually on the y axis) against sensitivity (recall). Examples of ROC and PR curves are shown in figure 2.9.

The reader may have noticed that both axes of the ROC curve track one of the two previously discussed performance measures that are intrinsic properties of the model. Indeed, this means that the behaviour of the ROC curve should not depend on the prevalence. On one hand, this property of the ROC curve makes it a robust measure of performance that requires no assumptions about the prevalence [36]. However, ROC curves must be interpreted with caution in situations where it is known that the model will have to operate in regimes of extremely high or low prevalence. This is because in such situations, it becomes necessary to guarantee either a very high sensitivity or specificity. As such, the regions of importance of the ROC curve become limited to their corresponding extremities. In such cases, looking at the entire ROC curve may create a misguided impression of good performance when this may not be the case when operating at the regimes of interest.

In contrast, the variables tracked by the PR curve include the PPV, which is explicitly affected by the prevalence. As such, a PR curve is useful in situations where there is a known typical prevalence – one which

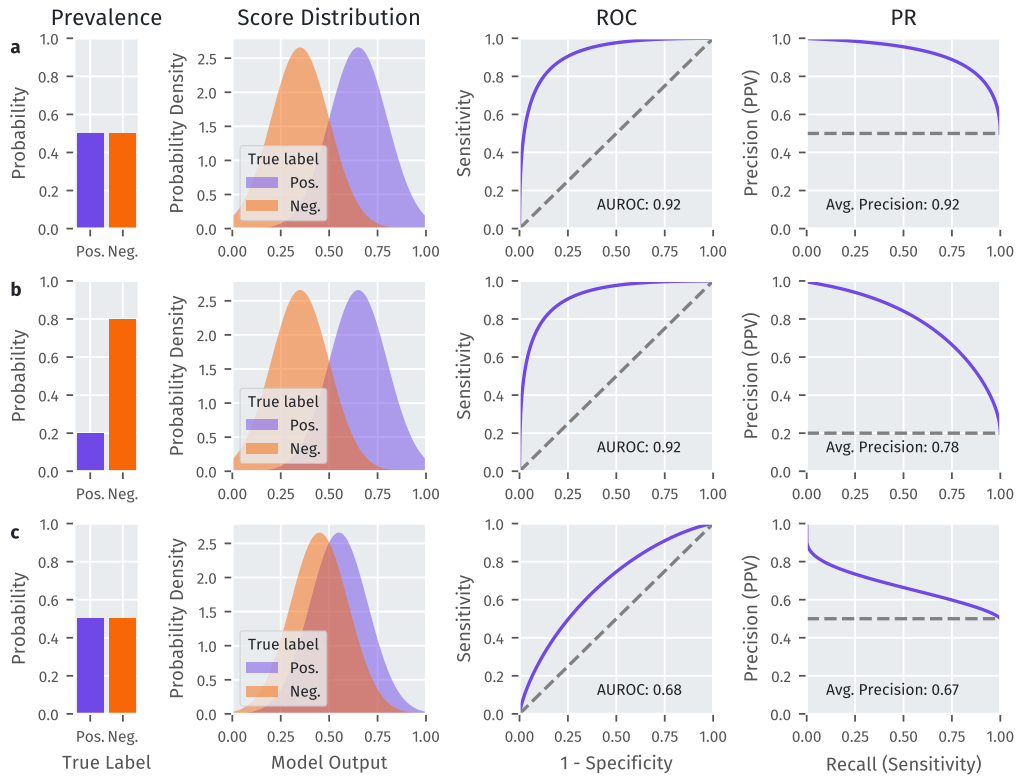


Figure 2.9: **An illustration of how the ROC and PR curves change with respect to changing classifier behaviour as well as the prevalence.** It is common for the model underlying a binary classifier to have a scalar output that is converted into a binary classification by defining a threshold output above which the model's prediction is set to be positive. The ROC curve plots the trade-off between sensitivity and specificity with varying thresholds, while the PR curve plots the trade-off between precision (PPV) and recall (sensitivity). **a)** Say the prevalence of some condition is 50% (leftmost plot). Then, a simulated binary classifier with the conditional output distribution shown in the 2nd plot from the left will produce the ROC and PR curve shown in the 3rd and 4th plots from the left respectively. **b)** A change in the prevalence does not affect the ROC curve, but significantly affects the PR curve. **c)** A change in the conditional model output distribution affects both the ROC and PR curves.

the model should be trained for, and will operate at [36]. On the other hand, a PR curve is less appropriate when the prevalence is either unknown or can change depending on the situation.

For both ROC and PR curves, a commonly reported summary statistic is the area under the curve (AUC). Since an AUC is effectively an integration over the x axis, it can be interpreted as the mean sensitivity across specificities (ROC) or the mean PPV across sensitivities (PR). This summary statistic is commonly called the area under the receiver operating characteristic (AUROC) for the ROC curve, and average precision for the PR curve [36].

2.6 Machine learning on T cell receptors

While the rest of this thesis presents my own efforts on machine learning strategies for TCR analysis, it is important to put this in the context of others' work as well. I conclude this chapter with a brief review of existing work in the TCR machine-learning field, broadly categorised by their formulations of the machine learning problem. In the following text, let f denote a machine-learning model, $\mathcal{T}$ the set of all possible TCRs, $\mathcal{P}$ the set of all pMHCs. Let $\tau \in \mathcal{T}$ and $\pi \in \mathcal{P}$ denote individual TCRs and pMHCs, respectively. In the vast majority of contexts, $\mathcal{T}$ and $\mathcal{P}$ are considered to be and modelled as spaces, so I will be referring to them as such.

2.6.1 pMHC-specific models

One straightforward approach taken by several existing works is to train pMHC-specific models $f_\pi : \mathcal{T} \to \mathbb{R}$ that takes an arbitrary TCR and predicts binding to some predetermined pMHC. Since these models are only concerned with $\mathcal{T}$, a key benefit is their ability converge to reasonably accurate solutions as long as the binder TCRs against the pMHC of interest are sufficiently well-sampled. However, an inherent downside is that a brand-new model must be trained for every new target pMHC. To be more precise, where there is no model of the pMHC space, it is difficult to find well-defined ways to extrapolate knowledge gained from one pMHC to another.

The earliest examples of this approach used decision trees, where input TCRs were featurised according to their β chain CDR3 amino acid sequences [38, 39]. Another work proposed training Gaussian process classifiers on a TCR representation space defined by featurising the α

and β chain V genes and CDR3 sequences using BLOSUM62 vectors [40]. More recently, deep neural networks of various architectures have been applied to this approach, including fully connected networks [41], convolutional neural networks (CNNs) [42, 43], and transformers [43–45], with all of them using some combination of V gene usage and CDR3 sequence data of both the α and β chains.

Independent benchmarking studies have demonstrated the success of these models in predicting specificities against pMHCs for which we have many known TCR binders [46], although prediction accuracy for pMHCs with few known binders remain limited [47]. Currently, no particular model architecture nor approach seem to show significantly superior performance over the others. However, it is now widely recognised that models with full $\alpha\beta$ chain inputs outperform α or β chain-only models, and this pattern continues to be seen for models in all categories discussed below [46].

2.6.2 General models of TCR-pMHC interaction

A more ambitious approach is to explicitly model both the TCR and pMHC spaces and learn some function $f : \mathcal{T} \times \mathcal{P} \to \mathbb{R}$ that can predict binding between any arbitrary TCR/pMHC pair. Due to their theoretical ability to generalise their predictions to all possible TCR/pMHC combinations, many consider learning an accurate function of this form to be the holy grail of computational TCR analysis. That being said, only a minute fraction of the diversity of the $\mathcal{T} \times \mathcal{P}$ product space is covered by the available specificity-annotated TCR data – especially with respect to $\mathcal{P}$ – which currently presents a key challenge to this approach [20].

The vast majority of work under this approach implement their models as deep neural networks, with the most popular architectures split between CNNs [48–54], recurrent neural networks [55–57], and transformers [58–61]. Other approaches have included decision trees and random forests [62, 63], fine-tuned protein structure predictors [64], physical models of receptor-ligand interaction [65], and neural network ensemble models [66]. All models cited above take TCR amino acid sequences as input, although only some [49, 56, 60, 61, 63–66] use both the α and β chains, with the rest using only β chain information. Three of the models [54, 64, 65] explicitly account for the structure of the input TCRs and pMHCs.

Evidence from independent benchmarking studies show that these more general models can produce accurate binding predictions for pMHCs

with many known TCR binders seen during training, where they show similar performance to those of the pMHC-specific models mentioned above [46]. Unfortunately, generalisable performance to unseen pMHCs currently remains elusive, most likely due to the limited coverage of the $\mathcal{T} \times \mathcal{P}$ space [20, 67]. There have been various attempts to ameliorate this issue of data limitation through innovations including transfer learning using general protein-ligand interaction data [51], pre-training on unlabelled TCR data [60, 61], leveraging pre-trained protein multimer structure predictors [64], and meta-learning [59], but none have convincingly been able to overcome the current data bottleneck.

2.6.3 Models of TCR co-specificity

Another approach is to shift to two-point statistics and re-frame the problem into TCR co-specificity prediction. Here, the objective is to learn a model $f : \mathcal{T} \times \mathcal{T} \to \mathbb{R}$ that takes arbitrary TCR pairs and predicts their likelihood of sharing specificity. This can be thought of as learning a similarity measure or distance on TCR space. As discussed in section 2.3, a benefit of shifting to two-point statistics is that we explicitly put our focus on pair-wise patterns that generalise across pMHCs. Also mentioned was a benefit from the machine-learning point of view that sampling pairwise data points is cheaper, with N TCRs generating $\binom{N}{2} \sim O(N^2)$ pairs. Finally, the pMHC-agnostic nature of the co-specificity formulation makes models of this kind particularly suitable for downstream analyses on data that only involve TCRs. A popular example of such analyses is TCR clustering. Co-specificity models can still be applied to conventional specificity prediction by making a series of co-specificity inferences between a query TCR and a set of known binders to the target pMHC. However, as with the pMHC-specific models, there exist no obvious ways of making predictions to unseen pMHCs with no known binders.

While models of the previous categories have been dominated by neural networks, TCR co-specificity models to date have been based on more traditional bioinformatics methods, such as sequence alignment [68–71] and string k-mer kernels [72]. When applied to TCR specificity prediction, evidence from benchmarking studies suggest that co-specificity models are similarly competitive to those of the previous two categories [46].

Despite the continued and widespread popularity of TCR clustering as a method of TCR repertoire analysis, the TCR co-specificity prediction paradigm seems to be understudied in comparison to the previous categories of models. In fact, only one of the works cited above [70] is a

true machine learning method trained on TCR data – the others all have model parameters that are inherited from existing work on general protein sequence statistics (e.g. the BLOSUM substitution matrices [73]), or hand-engineered by the authors. Some groups have proposed methods of mapping TCRs to numerical representation spaces, within which TCR co-specificity inferences can be made [74, 75]. However, these too are static, non-trained models whose main goals were to approximate existing sequence-based models but have faster computation times by being alignment-free.

2.6.4 TCR repertoire classifiers

Zooming out from individual TCRs, some works directly optimise models for classifying TCR repertoires. Here, the objective is to learn a model $f : \{\tau_i \in \mathcal{T}\}_{i=1}^{N} \rightarrow \mathbb{R}$ that takes an arbitrary set of TCRs (a TCR repertoire) and predicts some global property of the repertoire's host (typically disease state, like cancer vs non-cancer). A strength of this approach is that it directly addresses a key downstream application of computational TCR analysis. Furthermore, there are some practical arguments to be made for the relative scalability of data generation with repertoire-level labels, compared to functional assays at the individual TCR level needed to generate specificity-annotated TCR data.

As with other multiple instance learning problems, a key challenge for repertoire classification is that in most cases only a small fraction of the TCRs in a given repertoire is likely to be predictive of its disease state label. For example, a sample from a cancer patient's TCR repertoire may contain some cancer-reactive TCRs, but the vast majority of TCRs are likely to be bystanders. Additionally, there are many potential confounders when dealing with repertoire-level data, including repertoire host age and MHC restriction, repertoire location (e.g. peripheral blood vs. tissue), and statistical biases due to differences in sequencing assays. Therefore, care must be taken to ensure repertoire classification models do not over-fit to such false signals.

Existing work in this category include models trained to detect infectious diseases such as HIV and SARS-CoV-2 [13–15], autoimmune conditions like systemic lupus erythematosus [14], and early-stage cancers of various types [16–18]. While many of these works report impressive AUROC values of greater than 0.9, they still remain in a proof-of-concept stage, due to a current lack of validation on large scale, unbiased, and prospective patient data.

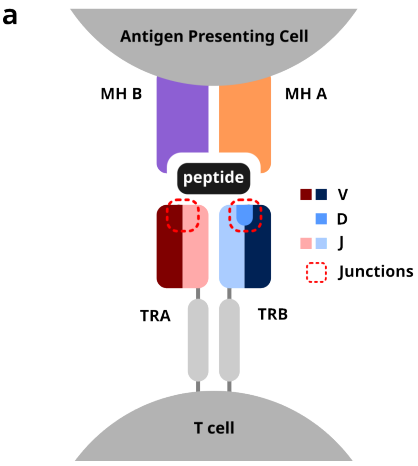
Chapter 3

Automating TCR and MHC data standardisation

TCRs underpin the diversity and specificity of T cell activity. As such, TCR repertoire data is valuable both as an adaptive immune biomarker, and as a way to identify candidate therapeutic TCR. Analysis of TCR repertoires relies heavily on computational analysis, and therefore it is of vital importance that the data is standardized and computer-readable. However in practice, the usage of different abbreviations and non-standard nomenclature in different datasets makes this data pre-processing non-trivial. `tidytcells` is a lightweight, platform-independent Python package that provides easy-to-use standardization tools specifically designed for TCR nomenclature. The software is open-sourced under the MIT license and is available to install from the Python package index (PyPI). All work presented in this chapter has been published prior to the submission of this thesis under my authorship and copyright [3] (licensed CCBY). This work was led by and written by myself, under the supervision of Benny Chain.

3.1 Introduction

T cells are an important immune cell population that helps orchestrate the vertebrate adaptive immune system. They express TCRs on their cell surface (Fig. 3.1a), which allows them to recognize and respond to antigens presented on the surfaces of other cells via the MHC proteins [5, 6]. Each T cell clone has a specific antigenic stimulus that it can respond to, often termed a T cell's "cognate antigen". The great range of target specificity is made possible by the fact that each new T cell clone generates its



b **Messy data**

v_call	junction_aa	j_call	peptide	mhc_a	mhc_b
TRBV11-2	ASQY	TRBJ2	VMAPRTLIL	A2	B2M
bv18	unknown	bj1.2	eeylkawtf	dqa1*6.2	dqb1*6.2

```
tidytcells.tr.standardize
tidytcells.mh.standardize
tidytcells.junction.standardize
tidytcells.aa.standardize
```

Standardized data

v_call	junction_aa	j_call	epitope	mhc_a	mhc_b
TRBV11-2	CASQYF	TRBJ2-1	VMAPRTLIL	HLA-A*02	B2M
TRBV18		TRBJ1-2	EEYLKAWTF	HLA-DQA1*06:02	HLA-DQB1*06:02

Figure 3.1: **a)** A diagram of a TCR interacting with a pMHC. The V, D and J genes comprising each TCR chain are shown by colour. The red dotted lines point out the junction sequences of both TCR chains. **b)** An illustration of how `tidytcells` can help clean TCR data. By using `tidytcells`, non-standard nomenclature in the "messy data" is corrected, and any invalid values are filtered out. Reprinted from my own publication [3] (licensed CCBY).

own unique TCR via a stochastic process of somatic gene rearrangement termed VDJ recombination.

Advances in high-throughput parallel sequencing allow large numbers of T cells isolated from blood or tissue to be sequenced. This gives us a snapshot of which T cells exist in the immune system of an individual at a given time, together with their frequency in the population, which is referred to as the individual's TCR repertoire. Because T cell clones proliferate after recognising their cognate antigen, TCR repertoire data has proven useful as an adaptive immune biomarker in various contexts, from cancer [16, 19, 76, 77] to SARS-CoV-2 infection [78–81], and there is growing excitement that TCR repertoire data can be used as a sensitive yet minimally invasive diagnostic biomarker for many other transmissible and non-transmissible diseases [14]. TCR repertoire data may also be exploited for therapeutic purposes, for example in the context of cellular therapies by contributing to the identification of TCR with reactivities against clinically relevant targets [82, 83].

The TCR is a heterodimer, created by imprecise somatic recombination of one of a set of V and J genes (α and γ chains), or V, J and D genes (β and δ chains) (Fig. 3.1a). The current convention is to represent TCR

sequence data by specifying, for each of the two chains that comprise it, which variable (V) and joining (J) genes are used, and what the amino acid sequence is of the *junction* region (also known as the CDR3) between the V and J genes (Fig. 3.1b). Because of junctional imprecision, this sequence is not template-driven, and cannot be aligned to the germline sequence. In many cases, for example where populations of cells are lysed before sequencing, α/γ to β/δ chain pairing is unresolved, and indeed only one chain (typically the TCR β) may be sequenced. In many studies, TCR sequences are further annotated by their cognate peptide/MHC (Fig. 3.1b).

Although the immunology community has generally converged to a common format of TCR data representation and has developed an international standardised nomenclature [84], TCR data in practice still contain variation due to several issues. These include (Fig. 3.1b):

- The use of non-standard TCR/MHC gene symbols
- The inclusion of non-functional TCR genes when one is only interested in data for functional TCR
- Differing levels of TCR/MHC gene resolution- for example, some data may resolve TCR genes to the level of the allele (TRAV1-1*01) while others only to the level of the gene (TRAV1-1)

This variation is particularly problematic when trying to compile large sets of machine-readable TCR data for downstream computational analysis. For example, algorithms may not easily recognise that TRAV1-1 and TRAV1-1*01 are in fact the same TRAV. Similarly, HLA-A*01 may not be understood as semantically identical to the abbreviated symbol A1.

`tidytcells` is a lightweight python package that addresses this issue by providing simple-to-use utilities to standardize TCR nomenclature. Its primary content is a set of functions that can convert non-standard TCR/MHC gene symbols into their International immunogenetics information system (IMGT)-standard versions [85]. Additionally, it provides simple functions to standardize junction and epitope amino acid sequences, as well as some other extra utilities.

`tidytcells` is available on the PyPI at `https://pypi.org/project/tidytcells/`. The source code is available at `https://github.com/yutanagano/tidytcells` under the MIT license. For more details such as the API reference, please see the documentation at `https://tidytcells.readthedocs.io/en/latest/`.

3.2 Method (Software Features)

I provide a high-level overview of `tidytcells`' features below. For more detailed instructions on use, including API references for each function, please refer to the documentation page. The API references for each of the standardization functions also include an outline of their decision logic, so that users can make informed decisions about their scope and limitations.

3.2.1 TCR gene symbol standardisation

`tidytcells` provides the function `tr.standardize`, which takes as input a string representing a potentially non-standard TCR gene symbol, and outputs the corresponding IMGT-standardized symbol.

By default, if the input string cannot be resolved to a known TCR gene, the function outputs `None`. The function attempts to standardize to human TCR genes by default, but *Mus musculus* genes are also supported. Further options can be specified to exclude non-functional TCR genes, or limit the resolution of the symbols to the level of the gene (as opposed to allele). Below is a code block demonstrating the use of `tr.standardize`.

```
>>> import tidytcells as tt
>>> tt.tr.standardize("aj1")
"TRAJ1"
>>> tt.tr.standardize("TRBV6-4*01", precision="gene")
"TRBV6-4"
>>> result = tt.tr.standardize("TRBV1", enforce_functional=
    True)
UserWarning: Failed to standardize "TRBV1" for species
    homosapiens: gene has no functional alleles.
>>> print(result)
None
>>> tt.tr.standardize("TCRBV22S1A2N1T", species="musmusculus"
    )
"TRBV2"
```

3.2.2 MHC gene symbol standardisation

A similar function `mh.standardize` is available for standardizing MHC gene symbols. Its function signature and behaviour is essentially equivalent to its TCR counterpart.

```
>>> tt.mh.standardize("HLA-A*01:01:01", precision="protein")
```

```
"HLA-A*01:01"
>>> tt.mh.standardize("CRW2", species="musmusculus")
"MH1-M5"
```

3.2.3 Junction/epitope sequence standardisation

`aa.standardize` and `junction.standardize` provide standardization utilities for amino acid sequences. `aa.standardize` can be used to clean generic amino acid sequence data, including epitopes, while `junction.standardize` provides TCR junction sequence-specific logic.

```
>>> tt.junction.standardize("sadaf")
"CSADAFF"
>>> result = tt.junction.standardize("sadaf", strict=True)
UserWarning: Input sadaf was rejected as it is not a valid
   junction sequence.
>>> print(result)
None
```

3.2.4 Extra utilities

A brief list of additional features provided by `tidytcells` is shown in table 3.1.

Table 3.1: A brief overview of extra utilities provided by `tidytcells`.

Function	Description
`mh.get_chain`	Given an MHC gene symbol, classify as α or β chain
`mh.get_class`	Given an MHC gene symbol, classify as MH1 or MH2
`mh.query`	Query the list of all known MHC genes/alleles
`tr.get_aa_sequence`	Obtain the underlying amino acid sequence of a TCR gene
`tr.query`	Query the list of all known TCR genes/alleles

3.3 Results (Application to real data)

As a test use case of `tidytcells`' functionality, I used it in combination with the `pandas` package to clean TCR and MHC data from the Immune Epitope Database (IEDB) [24]. Where species data was available on the database, it was used. For TCR or MHC samples missing species labels, the species *Homo sapiens* was assumed. Other settings were left at default

Table 3.2: Examples of standardisation successes.

Category	Species	Input	Output
TCR	*Homo sapiens*	TCRBV17S1	TRBV17
	Homo sapiens	TRAV15	TRAV15-1/DV6-1
	Homo sapiens	29/DV5*01	TRAV29/DV5*01
	Homo sapiens	TCRBV5-1*01 or TCRBV5-1*02	TRBV5-1*01
	Mus musculus	TCRAV14D-3/DV8*02	TRAV14D-3/DV8*02
MHC	*Homo sapiens*	HLA-A*02:01 W167A mutant	HLA-A*02:01
	Homo sapiens	DQB1*06:02	HLA-DQB1*06:02
	Homo sapiens	B2M	B2M
	Mus musculus	H2-Q9	MH1-Q9
	Mus musculus	H2-Db	MH2-D1
Junction	N/A	AASANSGTYQR	CAASANSGTYQRF
	N/A	CSVNRDTGAGGYTF	CSVNRDTGAGGYTF
Epitope	N/A	VMAPRTLIL	VMAPRTLIL

values, and no particular restrictions on gene functionality were imposed. Standardization was considered a success if the species associated with a particular value was supported, and the function managed to resolve the value to a recognised IMGT-compliant symbol.

Out of 2225 unique TCR gene symbol values found in the database, 2127 values (95.6%) were standardized successfully. Similarly, 173 of 284 MHC genes (60.9%), 301,554 of 301,670 junction sequences (99.9%) and 1825 of 1996 epitopes (91.4%) were standardized. Some examples of standardization successes and failures are shown in tables 3.2 and 3.3.

3.4 Discussion

As demonstrated, `tidytcells` in its current form can successfully standardise the majority of TCR/MHC data from public databases such as IEDB. However, there are still limitations to its standardization ability. Below I discuss current limitations that we as maintainers of `tidytcells` hope to address in the near future. The package is entirely open source and code contributions from the community are welcome.

Currently, when `tidytcells` encounters a string of the general form "A B" where A is a valid example of what it is attempting to standardize, it will ignore "B" and return "A" as the standardized form. This works well for cases like the first MHC success example, where the B string is a

Table 3.3: Examples of standardisation failures.

Category	Species	Input	Reason for failure
TCR	*Homo sapiens*	TCRAJ1-3	Nonexistent gene
	Homo sapiens	TRBV14DV4	Nonexistent gene
	Mus musculus	1	Insufficient information
	Mus musculus	12D-2	Insufficient information
	Mus musculus	TRVB13-1*02	Nonexistent gene
MHC	*Homo sapiens*	HLA class II	Insufficient information
	Homo sapiens	HLA-DQ	Insufficient information
	Homo sapiens	human MR1 K43A mutant	Non-classical HLAs not supported
	Mus musculus	M23I	Mutation specifier with no gene
	Mus musculus	HLA-DRB1*04:01	Incorrect species annotation
Junction	N/A	IVRVSHN*G#RDNYGQNFV	Ambiguity symbols not supported
Epitope	N/A	LLFGFPVYV + SCM(F5)	Peptide modification not supported
	N/A	diclofenac	Not a peptide

qualifier string that can be removed without fundamentally changing the underlying data. However, in cases like the fourth TCR success example where the string is of the form "A or B", the intuitively better representation of the underlying data is to standardize to the greatest common factor of A and B (i.e. TRBV5-1 in this case). Implementing separate logic to handle these cases would improve standardization quality.

For junction sequence standardization, the default behaviour when dealing with a valid amino acid sequence that does not start with a cysteine (C) and end with a phenylalanine (F) or tryptophan (W) is to append a "C" at the beginning and an "F" at the end, and return the resulting string. The logic is implemented this way because the most common reason for these missing residues is that some data sources encode the junction as the CDR3 sequence (without the starting C and ending F/W). However, this rudimentary logic always assumes that the junction terminates with an F rather than a W. A possible improvement would be to use prior knowledge of the amino acid sequences of J genes to better predict the terminal residue. It may also be useful to provide an option to perform the reverse procedure (i.e. remove the C and F/W residues).

Other areas of potential improvement include parsing amino acid ambiguity codes and peptide modification syntax, more optional standard-

ization constraints (e.g. specify a-priori that values should be resolved to TRAV genes/alleles as opposed to any TCR gene), support for non-classical MHC, allele imputation (if a gene has only one allele, resolve to that allele), and support for more species (only *Homo sapiens* and *Mus musculus* are currently supported).

3.5 Conclusion

`tidytcells` is a lightweight Python package that solves the issue of messy TCR/MHC data by providing easy-to-use utilities for standardizing TCR/MHC gene symbols, as well as general and TCR junction amino acid data. I believe this will prove to be a useful utility to the rapidly growing community of scientists who are studying the TCR repertoire.

Chapter 4

Contrastive learning of T cell receptor representations

Computational prediction of the interaction of TCRs and their ligands is a grand challenge in immunology. Despite advances in high-throughput assays, specificity-labelled TCR data remains sparse. In other domains, the pre-training of language models on unlabelled data has been successfully used to address data bottlenecks. However, it is unclear how to best pre-train protein language models for TCR specificity prediction. Here I introduce a TCR language model called Simple contrastive embedding of the primary sequence of T cell receptors (SCEPTR), capable of data-efficient transfer learning. Through this model, I introduce a novel pre-training strategy combining autocontrastive learning and masked-language modelling, which enables SCEPTR to achieve its state-of-the-art performance. In contrast, existing protein language models and a variant of SCEPTR pre-trained without autocontrastive learning are outperformed by sequence alignment-based methods. I anticipate that contrastive learning will be a useful paradigm to decode the rules of TCR specificity. All work presented in this chapter has been distributed publicly as a preprint prior to the submission of this thesis under my authorship and copyright [29] (licensed CCBY). This work was led by and written by myself, with the collaboration of Andrew Pyo, Martina Milighetti, James Henderson, and John Shawe-Taylor, and under the supervision of Benny Chain and Andreas Tiffeau-Mayer.

4.1 Introduction

Antigen-specific T cells play important protective and pathogenic roles in human disease [2]. The recognition of pMHCs by $\alpha\beta$TCRs determines the specificity of cellular immune responses [4]. Hyperdiverse $\alpha\beta$TCRs are generated during T cell development in the thymus by genetic recombination of germline-encoded V, D (for TCR β) and J gene segments with additional diversification through insertions and deletions of non-template nucleotides at gene segment junctions.

A major goal of systems immunology is to uncover the rules governing which TCRs interact with which pMHCs [20]. Advances in high-throughput functional assays of TCR specificity [68, 86, 87] have made the use of machine learning a promising prospect to discover such rules.

The most direct approach for applying machine learning to TCR specificity prediction has been to train pMHC-specific models that take an arbitrary TCR and predict binding [38, 40, 41, 44, 45, 49]. More ambitiously, model architectures have been proposed that can in principle generalise predictions to arbitrary pMHCs as well [51–53, 56–62, 65, 88]. Independent benchmarking studies have shown that both approaches are effective for predicting TCR binders against pMHCs for which many TCRs have been experimentally determined [46], but generalisation to pMHCs not seen during training has largely remained elusive [67] and prediction accuracy is limited for pMHCs with few known binders [47]. This severely limits the utility of current predictive tools given that only $\sim 10^3$ of the $> 10^{15}$ possible pMHCs are currently annotated with any TCRs in VDJdb [23], and given that for $>$ 95% of them less than 100 specific TCRs are known.

Meanwhile, there is abundant unlabelled TCR sequence data that may be exploited for unsupervised representation learning. A TCR representation model that compactly captures important features would provide embeddings useful for data-efficient training of downstream specificity predictors.

In NLP, unsupervised pre-trained transformers have demonstrated capacity for transfer learning to diverse downstream tasks [27, 30, 89]. This has spurred substantial work applying transformers to protein analysis. protein language models (PLMs) such as those of the ESM [90, 91] and ProtTrans [92] families have been successfully used in structure-prediction pipelines and for protein property prediction [93–95]. PLMs have also been applied to TCR-pMHC interaction prediction [44, 60, 61, 88], and the related problem of antibody-antigen interaction prediction [96, 97]. However, there has been limited systematic testing of how com-

petitive PLM embeddings are in the *few-shot* setting typical for most ligands – that is, where only few labelled data points are available for transfer learning.

To address this question, we benchmarked existing PLMs on a standardised few-shot specificity prediction task, and surprisingly found that they are inferior to state-of-the-art sequence alignment-based methods. This motivated us to develop SCEPTR, a novel TCR PLM which closes this gap. Our key innovation is a pre-training strategy involving an autocontrastive learning procedure adapted for $\alpha\beta$TCRs, which we show is the primary driver behind SCEPTR's improved performance.

4.2 Results

4.2.1 Benchmarking PLM embeddings on TCR specificity prediction

Given the scarcity of specificity-labelled TCR data, it is of practical importance to evaluate model performance where access to such data is limited. Therefore, we set up a benchmarking framework focused on few-shot TCR specificity prediction.

To conduct our benchmark, we curated a set of specificity-labelled $\alpha\beta$TCR data from VDJdb [23]. We only included human TCRs with full α and β chain information, and excluded data from an early 10x Genomics whitepaper [98], as there are known issues with data reliability in this study [99, 100]. This left us with a total of 7168 $\alpha\beta$TCRs annotated to 864 pMHCs. Of these, we used the six pMHCs with greater than 300 distinct binder TCRs for our benchmarking task.

We created a benchmarking task that allowed us to directly compare sequence alignment-based distance metrics such as the state-of-the-art TCRdist [68, 82] (Fig. 4.1a) to distances in PLM embedding spaces (Fig. 4.1b). For each pMHC, we tested models on their ability to distinguish binder TCRs from non-binders using embedding distances between a query TCR and its closest neighbour within a reference set (Fig. 4.1c). We call this *nearest-neighbour prediction*. This framework is simple and attractive for benchmarking models in the few-shot regime, since it remains well defined for as few as a single reference TCR and does not require model specific fine-tuning.

We conducted multiple benchmarks for each pMHC, varying the number of its cognate TCRs used as the reference set. In each case, we combined the remaining TCRs for the target with the rest of the filtered VD-

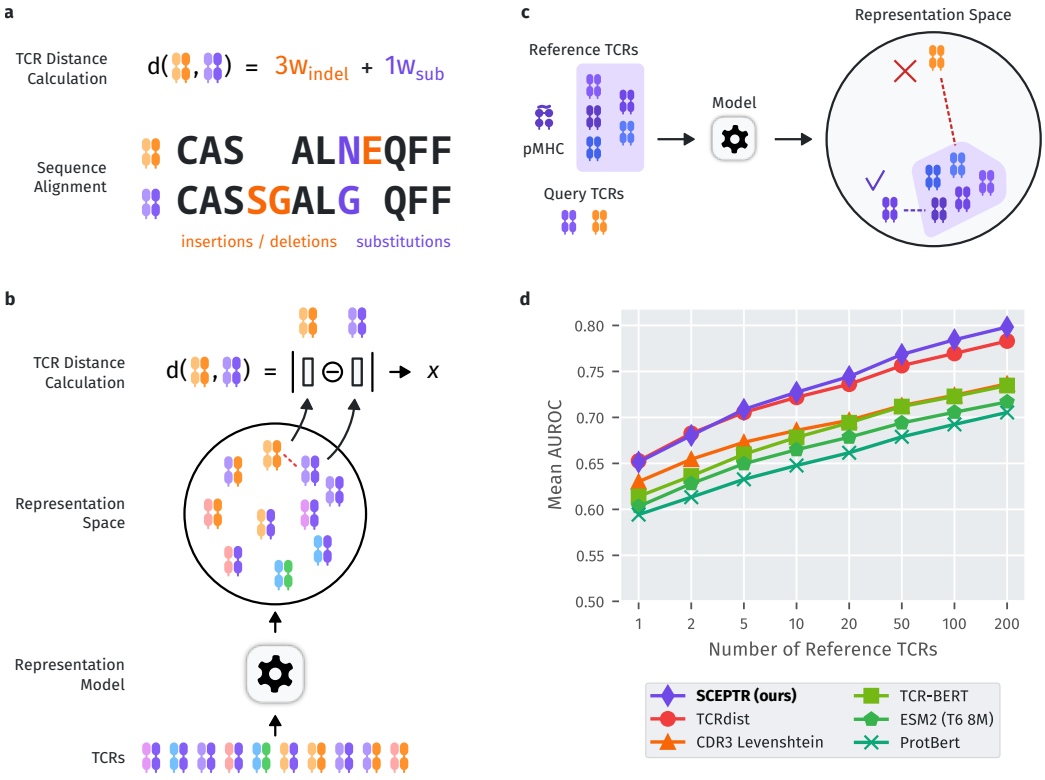


Figure 4.1: **Benchmarking TCR language models against sequence alignment-based approaches on few-shot TCR specificity prediction. a)** TCR similarity can be quantified using sequence-alignment by taking a (weighted) count of how many sequence edits turn one TCR into another. **b)** Learned sequence representations allow alignment-free sequence comparisons based on distances in the embedding feature space. **c)** Sketch of our standardized benchmarking approach to allow side-by-side comparison of sequence-alignment and embedding methods. Using a reference set of known TCR binders to a pMHC of interest, we propose nearest-neighbour prediction as a task for unbiased comparison of the quality of embeddings for specificity prediction. **d)** Performance of six different models on TCR specificity prediction as a function of the number of reference TCRs. Specificity predictions were made by the nearest neighbour method sketched in **c** against six different pMHCs and performance is reported as the AUROC averaged across the pMHCs. Reprinted from my own preprint [29] (licensed CCBY).

Jdb dataset (including TCRs annotated to pMHCs other than the six target pMHCs) to create a test set (see methods 4.4.1). By studying how performance depends on the size of the reference set, we are effectively probing representation alignment with TCR co-specificity prediction at different scales.

We benchmarked six models: two alignment-based TCR metrics (CDR3 Levenshtein distance and TCRdist [68]), two general-purpose PLMs (ProtBert [92] and ESM2 [91]), and two TCR domain-specific language models (TCR-BERT [44] and our own model SCEPTR). We report performance using the AUROC averaged over the tested pMHCs.

To our surprise, we found that TCR-BERT, ESM2, and ProtBert all fail to outperform the baseline sequence alignment method (CDR3 Levenshtein) and are significantly inferior to TCRdist (Figs. 4.1d / A.1). A repeat of the benchmarking with a broader set of epitopes obtained by including post-processed 10x Genomics whitepaper data [100], recapitulated these results, demonstrating the robustness of our findings (Fig. A.2). In contrast to existing PLMs, SCEPTR performs on par with or better than TCRdist. For a reference set of size 200, SCEPTR performs best among all models for five out of six tested peptides (Table A.1).

We additionally compared models using the average distance between a query TCR and all references, instead of only the nearest neighbour (Fig. A.3). In this case, SCEPTR outperforms other models by an even wider margin. Interestingly, all models perform worse compared to their nearest neighbour counterpart. This finding might be explained mechanistically by the multiplicity of viable binding solutions with distinct sequence-level features, which are thought to make up pMHC-specific TCR repertoires [22, 101].

4.2.2 Autocontrastive learning as a pre-training strategy

We now briefly summarize SCEPTR's architecture and autocontrastive pre-training strategy (see Methods for full details). SCEPTR featurises an input TCR as the amino acid sequences of its six CDR loops. It uses a simple one-hot encoding system to embed the amino acid tokens, and uses a stack of three self-attention layers to generate a 64-dimensional representation vector of the input receptor (Fig. 4.2a,b). Unlike existing TCR language models, SCEPTR is jointly pre-trained using autocontrastive learning and MLM (Fig. 4.2c,d).

To motivate the considerations that have led us to adopt this training paradigm, some background on transformer architectures and their

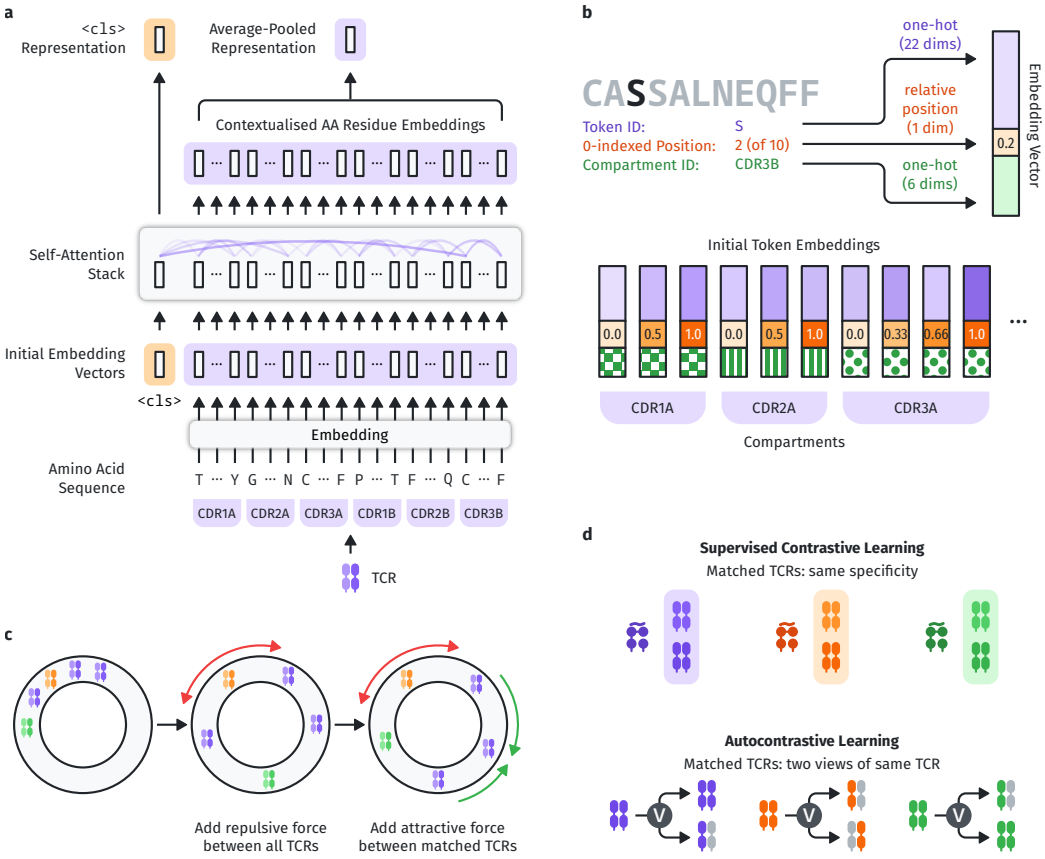


Figure 4.2: **A visual introduction to how SCEPTR works. a)** SCEPTR featurises an input TCR as the amino acid sequences of its six CDR loops. SCEPTR uses the contextualised embedding of the `<cls>` token as the overall TCR representation, in contrast to the average-pooling representations used by other models. **b)** SCEPTR's initial token embedding module uses a simple one-hot system to encode a token's amino acid identity and CDR loop number, and allocates one dimension to encode the token's relative position within its CDR loop as a single real-valued scalar. **c)** Contrastive learning allows us to explicitly optimise SCEPTR's representation mapping for TCR co-specificity prediction. **d)** With supervised contrastive learning, positive pairs are generated by sampling pairs of TCRs that are known to bind the same pMHC. In the unsupervised setting, positive pairs are generated by taking two independent "views" of the same TCR. We implement this by independently removing a proportion of input tokens and sometimes dropping the α or β chain entirely for every view (see methods 4.4.3). Reprinted from my own preprint [29] (licensed CCBY).

training by MLM needs to be introduced. The transformer is a neural network developed in NLP that uses dot-product attention to flexibly learn long-range dependencies in sequential data [27]. BERT is an encoder-only variation of the transformer useful for text analysis and processing [30]. BERT's innovation was its ability to be pre-trained in an unsupervised manner through MLM, where snippets of text are fed to the model with a certain proportion of tokens (e.g. words) masked, and the model must use the surrounding context to reconstruct the masked tokens. MLM allowed BERT and its derivative models to exploit large volumes of unlabelled data to learn grammar and syntax and achieve high performance on downstream textual tasks with comparatively little supervised fine-tuning.

While MLM-trained PLMs have been successful in some protein prediction tasks [90–92], they have been documented to struggle with others [95]. Our benchmarking results led us to believe that MLM pre-training may not be optimal for TCR-pMHC specificity prediction. Firstly, the majority of observed TCR sequence variation is attributable to the stochastic process of VDJ recombination. As such, MLM may not teach models much transferable knowledge for specificity prediction. Secondly, since the low volume of specificity-labelled TCR data provides limited opportunities for fine-tuning complex models, representation distances should ideally be directly predictive of co-specificity.

We were inspired to use contrastive learning to overcome these problems by the success of our previous work using statistical approaches to uncover patterns of sequence similarity characteristic of ligand-specific TCR repertoires [22, 101, 102]. Contrastive learning minimises distances between model representations of positive sample pairs while maximising distances between background pairs (Fig. 4.2c) through a loss function of the following form [103, 104]:

$$\mathcal{L}_{\text{contrastive}}(f) := \mathop{\mathbb{E}}_{\substack{(x,x^+)\sim p_{\text{pos}} \\ \{y_i\}_{i=1}^{N} \overset{\text{iid}}{\sim} p_{\text{data}}}} \left[-\log \frac{e^{f(x)^\top f(x^+)}}{e^{f(x)^\top f(x^+)} + \sum_i e^{f(x)^\top f(y_i)}} \right] \tag{4.1}$$

where $f : \mathcal{X} \to \mathcal{S}^{m-1}$ is a trainable embedding mapping from sample observation space $\mathcal{X}$ to points on the m-dimensional unit hypersphere $\mathcal{S}^{m-1} \subset \mathbb{R}^m$, p_{pos} is the joint distribution of positive pairs, p_{data} is the overall data distribution, and $N \in \mathbb{Z}_+$ is some fixed number of background samples.

There are several well-known variants of this learning approach. In supervised contrastive learning, positive pairs are generated by sampling

observations known to belong to the same class (Fig. 4.2d top). In the context of TCRs, we can define positive pairs to be TCRs annotated to interact with the same pMHC, in which case we can show that contrastive learning regresses distances between TCR pairs to their probabilities of co-specificity (see appendix B). Autocontrastive learning approximates such positive pairs through data augmentation by generating two independent views of the same observation by passing it twice through the model (Fig. 4.2d bottom).

Given the scarcity of available labelled data, we opted to use the autocontrastive approach for purely unsupervised PLM pre-training (see Sec. 4.2.5 for an application of supervised contrastive learning, more similar to other recent applications of contrastive learning to TCRs [105, 106]). We generate different "views" of a TCR by dropout noise as is standard in NLP [107], but additionally adopted a censoring strategy inspired by MLM that randomly removes a proportion of residues or even complete α or β chains. In contrast to the only other study known to us having explored the application of autocontrastive learning to TCRs [108], we trained SCEPTR on all six hypervariable loops of the full paired chain $\alpha\beta$ TCR, as all contribute to TCR-pMHC specificity [101]. That being said, our chain dropping procedure during censoring ensures that single chain data are also in distribution for the model, giving SCEPTR flexibility for downstream applications with bulk sequenced TCR repertoires.

We define SCEPTR's output representation vector to be a contextualised embedding of a special input token called `<cls>` (the naming convention for `<cls>` comes from the fact that the output of this vector is often used for downstream *classification* [30]), which is always appended to the tokenised representation of an input TCR (Fig. 4.2a). This allows SCEPTR to fully exploit the attention mechanism when generating the overall TCR representation. Such training of a sequence-level representation is uniquely made possible by having an objective – the contrastive loss (Eq. 4.1) – that directly acts on the representation output. In contrast, MLM-trained PLMs such as ProtBert, ESM2 and TCR-BERT generate sequence embeddings by average-pooling the contextualised embeddings of each input token at some layer: a destructive operation which risks diluting information [95].

4.2.3 Ablation studies

To understand which modelling choices drive the improved performance of SCEPTR, we trained variants of SCEPTR ablating a single component

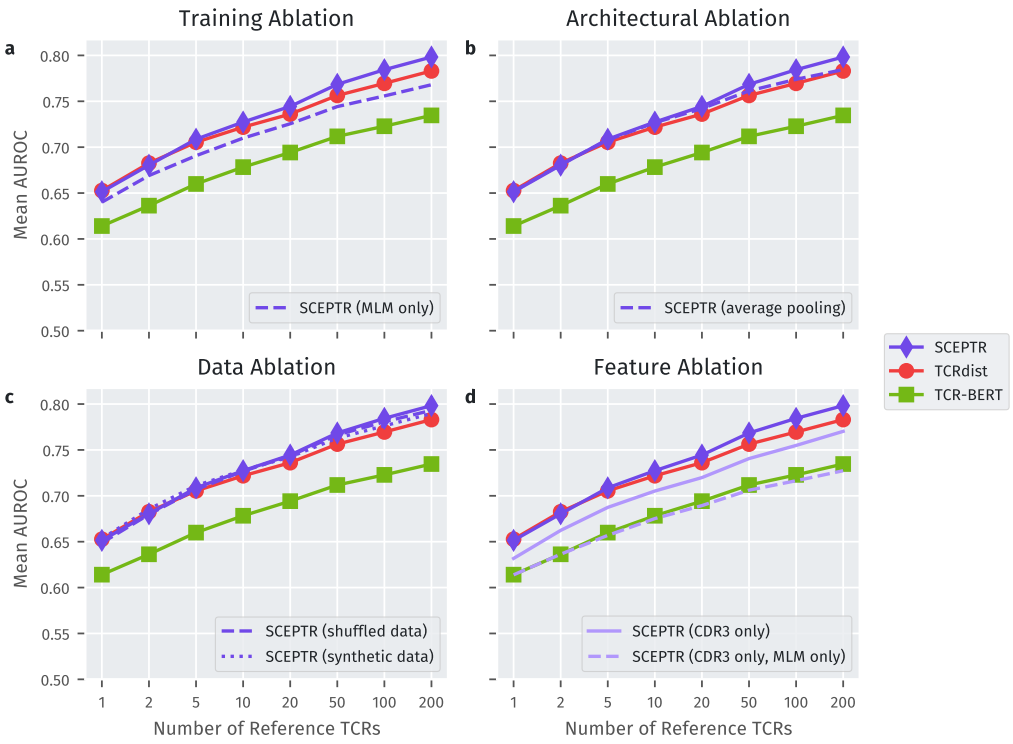


Figure 4.3: **Autocontrastive pre-training significantly improves SCEPTR's downstream performance.** The subplots show performance profiles of SCEPTR, TCRdist, TCR-BERT, and various ablation variants of SCEPTR on binary specificity prediction. **a)** Training SCEPTR solely on MLM results in worse specificity prediction performance. **b)** The baseline SCEPTR variant which uses the `<cls>` pooling method performs marginally better than the variant which uses the average-pooling method. However, the average-pooling variant still performs on par with TCRdist. **c)** Replacing SCEPTR's pre-training dataset with 1) the same dataset from Tanno et al., but with α/β chain pairing shuffled, and 2) synthetic data generated by OLGA both result in slightly inferior specificity prediction performance. **d)** Restricting SCEPTR's featurisation of input TCRs to the amino acids of the α and β CDR3 loops significantly worsen downstream performance. Additionally restricting training to only MLM further degrades performance, and produces a model with a near-equivalent performance profile to TCR-BERT. Reprinted from my own preprint [29] (licensed CCBY).

of either its architecture or training at a time, and benchmarked them using the framework described previously.

To establish the contribution of the autocontrastive learning to SCEPTR's performance, we trained the same model only using masked-language modelling:

SCEPTR (MLM only) This variant is trained only on MLM, without jointly optimising for autocontrastive learning. Following convention in the transformer field [109], TCR representation vectors are generated by average-pooling the contextualised vector embeddings of all constituent amino acid tokens produced by the penultimate self-attention layer, and $\ell 2$-normalising the result.

The MLM-only variant underperforms compared to both SCEPTR and TCRdist, demonstrating that autocontrastive learning is a necessary ingredient for the increased performance of SCEPTR in few-shot specificity prediction (Figs. 4.3a / A.4a).

We next sought to determine how much our pooling strategy and training dataset choice contributed to SCEPTR's performance gain. First, we asked whether autocontrastive learning also improves embeddings generated via token average-pooling:

SCEPTR (average pooling) This variant receives both autocontrastive learning and MLM, but uses the average-pooling method to generate TCR representations.

While SCEPTR's `<cls>` embeddings achieve the best results, the autocontrastive average-pooling variant still performs on par with TCRdist (Figs. 4.3b / A.4b).

Second, we determined how the performance of SCEPTR depends on the precise dataset used for pre-training. To answer this question we trained two variants of SCEPTR using size-matched datasets:

SCEPTR (synthetic data) This variant is trained on a size-matched set unlabelled $\alpha\beta$TCRs generated by OLGA [7], a probabilistic model of VDJ recombination.

SCEPTR (shuffled data) This variant is trained on the same set of $\alpha\beta$TCRs as the original model, but the α/β chain pairing is randomised.

We find that pairing information and the use of real post-selection repertoires slightly improves SCEPTR's performance, but variants trained on alternative datasets still match TCRdist performance (Figs. 4.3c / A.4c).

The synthetic data models the statistics of VDJ recombination and therefore estimates a pre-thymic TCR distribution. As such, SCEPTR's slight performance improvement using peripheral blood TCRs hints at a potential ability of language modelling to extract information from imprints left by antigen-driven selection in memory repertoires [22]. Similarly, the slight performance increase on paired TCR data suggests some ability to extract information from pairing biases [9].

Taken together, these ablation studies provide evidence that autocontrastive learning is the main factor enabling SCEPTR to close the gap between PLMs and alignment-based methods.

Information-theoretic analysis of the sequence determinants of TCR specificity demonstrate that all CDR loops and their pairing are important for determining binding specificity [101]. To understand how much SCEPTR's improved performance with respect to TCR-BERT is due to the restriction of the latter model's input to the CDR3 alone, we trained variants of SCEPTR restricted to this hypervariable loop:

SCEPTR (CDR3 only) This variant only accepts the α and β chain CDR3 sequences as input (without knowledge of the V genes/first two CDR loops of each chain). It is jointly optimised for MLM and autocontrastive learning.

SCEPTR (CDR3 only, MLM only) This otherwise equivalent variant is only trained using the MLM objective, and thus uses the average-pooling representation method.

The results demonstrate that taking into account all CDR loops leads to a performance gain as expected (Figs. 4.3d / A.4d). We also see that autocontrastive learning even when restricted to CDR3 loops leads to a substantial performance gain, helping the autocontrastive CDR3 variant achieve similar performance to the full-input MLM-only variant (Fig. 4.3a,d).

4.2.4 Learning features within embedding spaces

So far we have focused on nearest-neighbour prediction using PLM embeddings as the most direct test of data-efficient transfer learning that works with as little as a single reference sequence. If slightly more data is available, another approach is to train simple supervised predictors atop PLM embeddings. To test how much such training can improve prediction performance, we trained linear support vector classifiers (SVCs) on the PLM embeddings provided by different models. In each instance, we

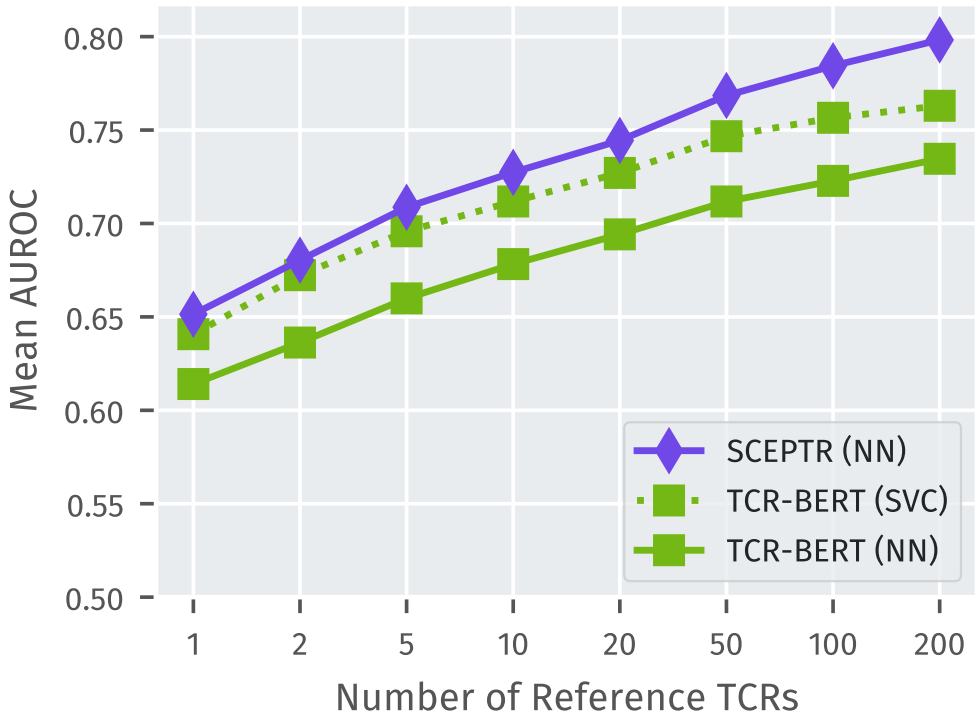


Figure 4.4: **SCEPTR with nearest-neighbour prediction outperforms other PLMs with a supervised support vector machine on top.** We trained a linear SVC on top of TCR-BERT's featurisation of the TCR to predict specificity, and compared its performance to the nearest-neighbour predictions of both SCEPTR and TCR-BERT using the benchmarking framework from section 4.2.1. We see that while training an SVC on top of TCR-BERT improves its downstream performance for few-shot TCR specificity prediction, it still does not outperform nearest-neighbour predictions as made by SCEPTR. Reprinted from my own preprint [29] (licensed CCBY).

trained the classifier to distinguish reference TCRs from 1000 randomly sampled background TCRs (see methods 4.4.4).

We find that the SVC predictors for ProtBert, ESM2 and TCR-BERT all perform better than their nearest-neighbour counterparts, but still worse than SCEPTR's nearest-neighbour predictions (Fig. 4.4 / A.6). We also trained an SVC atop SCEPTR, which did not lead to further improvement upon the nearest-neighbour prediction (Fig. A.6). These findings highlight how in the low data regime typical of most pMHCs, misalignment of pre-training to downstream tasks can only be partially remediated by training on reference TCRs.

4.2.5 Supervised contrastive learning for fine-tuning

Supervised contrastive learning provides an avenue to further optimise pre-trained embeddings for TCR specificity prediction. As a proof-of-concept, we fine-tuned SCEPTR to better discriminate between the six pMHC specificities used as the benchmarking targets in section 4.2.1.

For this task we took all the TCRs annotated against the target pMHCs from our labelled TCR dataset, and split them into a training, a validation, and a testing set. We ensured that no study used for training or validation contributed any data to the test set, so that the fine-tuned model would not be able to achieve good performance simply by exploiting inter-dataset biases. The training set included 200 binders against each target pMHC, totalling to 1200 TCRs. The rest of the TCRs from the same studies were used to construct the validation set. TCRs from all remaining studies were used for the testing set, which comprised of 5670 TCRs. SCEPTR was fine-tuned on the training set with supervised contrastive learning, using the validation loss for early stopping (appendix 4.4.5).

We used the framework from section 4.2.1 to benchmark the performance of fine-tuned SCEPTR, using the training set as the references. The results show that fine-tuning can greatly improve the ability of the model to discriminate between specificities (Fig. 4.5). Improvements are most noticeable for the pMHCs against which other methods achieve relatively low performance. However, model performance degrades with respect to unseen pMHCs (Fig. A.5), perhaps unsurprisingly given the very limited number of pMHCs represented in the training data. Interestingly, unlike other models (Fig. A.3), fine-tuned SCEPTR makes better inferences by measuring the average distance between a query TCR and all reference TCRs instead only the nearest TCR (Fig. A.7). This suggests an ability of supervised contrastive fine-tuning to help the model discover the com-

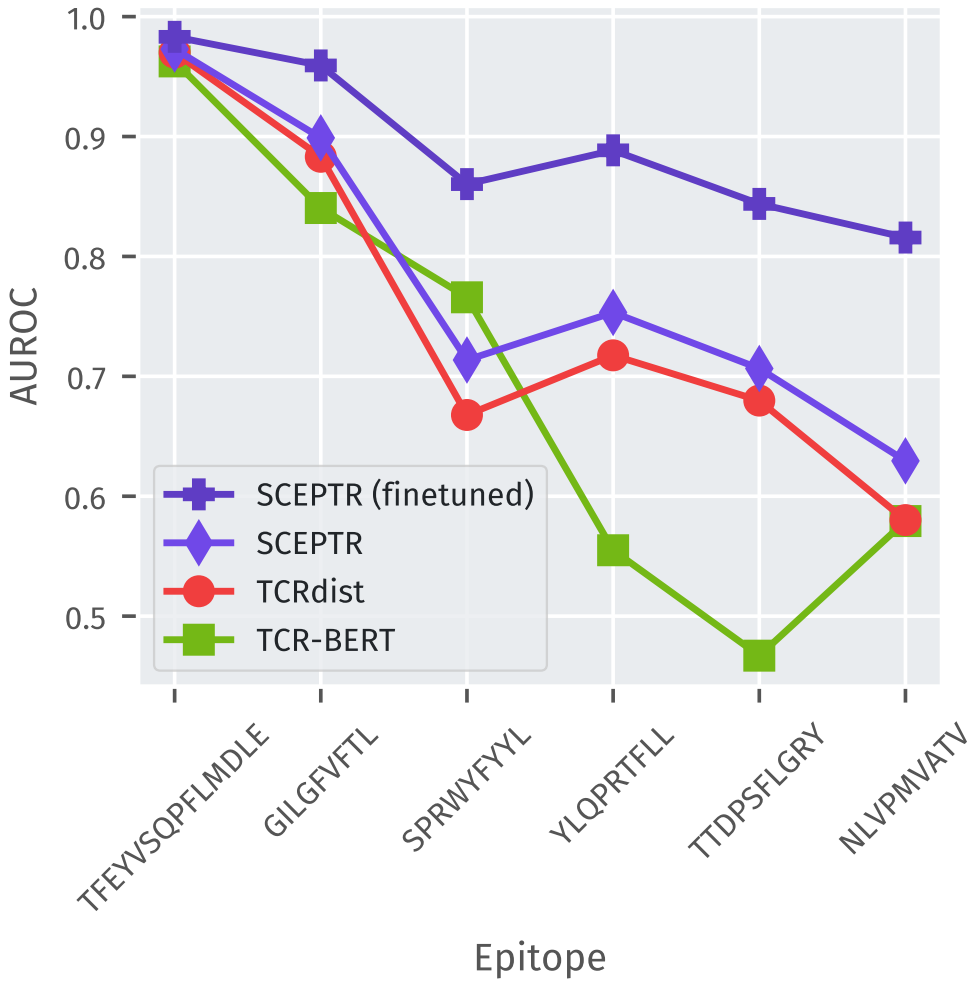


Figure 4.5: **Supervised contrastive learning improves discrimination between seen pMHCs.** Prediction performance as measured by AUROC on binary one-versus-rest classification for each of six pMHCs for different models. The fine-tuned model improves performance by exploiting the discriminative nature of the overall classification task. Reprinted from my own preprint [29] (licensed CCBY).

monalities between the multiple different binding solutions thought to exist for each pMHC.

4.3 Discussion

In this study, we have introduced SCEPTR, a pre-trained TCR PLM that achieves state-of-the-art few-shot TCR-pMHC specificity prediction accuracy. Through SCEPTR we demonstrate that joint autocontrastive and masked-language pre-training is a paradigm for learning PLMs better aligned with TCR specificity prediction tasks. Our model can be readily used for alignment-free TCR analysis in downstream applications (see code availability) including the unsupervised discovery of antigen-specific T cell groups (metaclonotypes) by sequence-based clustering [68, 82].

A limitation of our study is that we did not undertake a complete exploration of training and architectural hyper-parameters. We envisage multiple avenues that may improve SCEPTR further. Firstly, training could be made more efficient by optimising the distribution of masked/-dropped tokens during pre-training, taking into account the variable relevance of different parts of the sequence in determining specificity [101]. Secondly, as certain sequence motifs appear recurrently (e.g. CDR3 loops often begin with `CAS`), a more intelligent tokenisation scheme could offload learning of these primary sequence statistics into the tokenisation process.

Pre-trained PLMs have achieved high performance on protein stability and structural predictions [91, 92]. However, we find that existing PLMs fail to confer similar benefits to predicting TCR-pMHC interactions. This finding adds to recent work showing that current PLM pre-training is not well-aligned with certain downstream tasks [95]. Importantly, we show that autocontrastive pre-training can overcome misalignment, and thus provide a constructive path out of this impasse which could also be applied outside of the TCR domain.

What determines whether a certain downstream task is aligned with MLM pre-training? MLM teaches PLMs to predict the conditional distribution of tokens given sequence context. Thus it stands to reason that amenable downstream tasks involve predictions of properties that determine the distribution of observed proteins on sequence space. Observed proteins tend to concentrate in areas of sequence space with higher protein stability since evolution on average selects for this property [110]. For datasets containing protein families whose members have a conserved

structure despite primary sequence variation, co-evolutionary couplings driven by structural constraints influence allowed sequence variability [91, 111, 112]. These data distributional properties might explain how MLM can teach PLMs features related to both stability and structure.

In contrast, the distribution of TCRs over sequence space is primarily shaped by the biases of VDJ recombination with antigen-specific selection playing an important, but likely second-order effect [22]. While long-term evolutionary pressures may act to align recombination statistics with TCR function [10, 113], empirical evidence so far suggests recombination biases primarily anticipate thymic selection for stability and folding [114]. In contrast, studies to date have found no clear relationship between probabilities of recombination and the likelihood of receptors engaging specific pMHCs [115]. Given these considerations, we expect MLM pre-training to align better to tasks concerning VDJ recombination than to TCR specificity prediction. Indeed, previous work training PLMs on adaptive immune receptors has demonstrated that embeddings strongly depend on V/J gene usage and can be used to predict primarily generation-related properties such as receptor publicity [88, 96].

Why does autocontrastive learning help to generate embeddings better suited for specificity prediction? An interesting insight comes from an asymptotic decomposition of the contrastive loss function into the *uniformity* and *alignment* terms [103]:

$$\text{Unif.}(f) = \log \mathop{\mathbb{E}}_{x,y \overset{\text{iid}}{\sim} p_{\text{data}}} \left[e^{-\|f(x)-f(y)\|^2} \right] \tag{4.2}$$

$$\text{Align.}(f) := \mathop{\mathbb{E}}_{(x,x^+)\sim p_{\text{pos}}} \left[\|f(x) - f(x^+)\| \right] \tag{4.3}$$

Uniformity incentivises the model to make use of the full representation space, while alignment minimises the expected distance between positive pairs (e.g. co-specific TCRs) [103]. From this view, contrastive learning on adaptive immune receptor data encourages PLMs to undo the large-scale distributional biases created by VDJ recombination through the uniformity term, while helping to identify features relating to TCR (co-)specificity via the alignment term. While *auto*contrastive learning approximates the alignment term through the generation of pairs of views, it still provides a direct empirical estimate for the uniformity term. Thus, a key benefit of autocontrastive learning may be that it reduces the confounding effects of VDJ recombination in embedding space.

Comparing SCEPTR to other PLMs suggests that model complexity as measured by either parameter count or representation dimensionality is

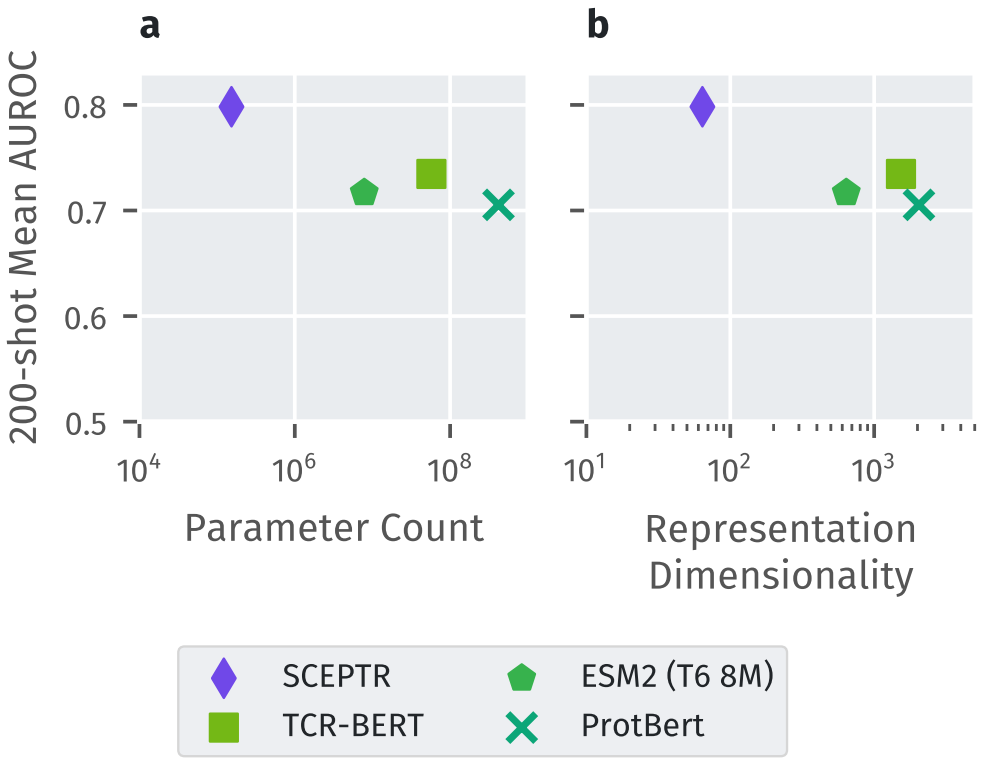


Figure 4.6: **Model complexity does not correlate with downstream performance.** Model performance as measured by mean 200-shot AUROC (section 4.2.1) does not scale with model complexity as measured by either **a)** parameter count or **b)** representation dimensionality. Despite being the smallest PLM by a wide margin, SCEPTR performs better than alternative models. Reprinted from my own preprint [29] (licensed CCBY).

not currently the limiting factor for TCR-pMHC prediction performance (Fig. 4.6). This is directly supported by how the CDR3-only, MLM-only variant of SCEPTR performs almost equivalently to the much larger but similarly pre-trained TCR-BERT (Fig. 4.3d). This finding stands in contrast to observations of performance scaling with model size in general PLMs [91] and antibody language modelling [97]. While the focus of the current study was on training a simple model, it would be interesting in future work to investigate performance scaling with model complexity and training dataset size with our novel training procedure.

Looking forward, there are many exciting avenues to further develop contrastive learning as a paradigm to crack the TCR code. For example, there may be ways to exploit the uniformity (Eq. 4.2) and alignment (Eq. 4.3) decomposition to simultaneously train on unlabelled and specificity-labelled data. A practical benefit of our contrastive learning formulation is that it does not require any optimisation with respect to the true negative distribution (i.e. TCRs that are explicitly not co-specific) – a non-trivial distribution to estimate for TCRs [116].

Another interesting avenue is the use of labels other than pMHC specificity – such as phenotypic annotations from single-cell data – as additional supervised contrastive training signals. Finally, while supervised contrastive learning does not currently lead to generalisable learning beyond training pMHCs, we expect a transition towards generalisation as larger volumes of specificity-labelled TCR data become available, as has been the case with supervised contrastive learning in other fields [107, 117–120]. Finally, while the focus of this work given current data limitations has been on learning TCR embeddings, contrastive learning may also help us learn effective joint TCR-pMHC embeddings in the future when the joint space (and particularly the pMHC space) is better sampled, and thus ultimately enable the zero-shot prediction of TCR-pMHC specificity.

4.4 Methods

4.4.1 Model benchmarking

For each pMHC, we varied the number k of reference TCRs where $k \in \{1, 2, 5, 10, 20, 50, 100, 200\}$. Within each model-pMHC-k-shot combination, we benchmarked multiple reference-test splits of the data to ensure robustness. For $k = 1$, we benchmarked every possible split. For $k \in [2, 200]$, we benchmarked 100 random splits, where we ensured that

the same splits were used across all models to reduce extraneous variance. Variations in model performance across different splits were used to generate confidence intervals for differences in performance, for example in figure A.1.

The CDR3 Levenshtein model computes the distance between two TCRs as the sum of the Levenshtein distances between the receptors' α and β CDR3 loops.

Note that while SCEPTR's architecture and training allows it to directly generate representation vectors for complete $\alpha\beta$ TCR sequences (Fig. 4.2a, methods 4.4.2), this is not the case for the other PLMs. For TCR-BERT, ESM2 and ProtBert representations for the α and β chains were independently generated, then concatenated together, and finally average-pooled to produce an embedding of the heterodimeric receptor (see appendix A.1).

4.4.2 SCEPTR architecture

SCEPTR is a BERT-like transformer encoder that maps TCR sequences to vector embeddings. Like BERT, it is comprised of a tokeniser module, an embedder module and a self-attention stack (Fig. 4.2a).

The tokeniser module represents each input TCR as the amino acid sequences of the first, second and third complementarity-determining regions (CDRs) of each chain, where each amino acid is a token. A special `<cls>` token is appended to each input TCR, as its contextualised embedding will eventually become SCEPTR's output representation vector (Fig. 4.2a,b).

SCEPTR uses a simple, non-trainable embedder module, where a one-hot vector is used to encode token identity (22 dimensions for 20 amino acids plus special tokens `<cls>` and `<mask>`), and token positions are specified by first one-hot encoding the containing CDR loop number (6 dimensions), then encoding the token's relative position within the loop as a single scalar variable (Fig. 4.2b). This results in initial token embeddings in $\mathbb{R}^{29}$, which are passed through a trainable linear projection onto $\mathbb{R}^{64}$. SCEPTR's self-attention stack then operates at this fixed dimensionality (see background section 2.4 for more details on the transformer and self-attention stack architecture). SCEPTR's self-attention stack comprises three layers, each with eight attention heads and a feed-forward dimensionality of 256, and is thus substantially simpler than existing models. Our tests suggest that relative position embedding helps SCEPTR learn better calibrated TCR co-specificity rules (see appendix A.2).

4.4.3 SCEPTR Pre-training

Data

The unlabelled paired-chain $\alpha\beta$TCR sequences used to pre-train SCEPTR were taken from a study by Tanno et al. [121], which provides 965,523 unique clonotypes sampled from the blood of 15 healthy human subjects. As opposed to traditional single-cell sequencing, Tanno et al. used a ligation-based sequencing method to resolve which α chains paired with which β chains. To mitigate potential noise from incorrect chain pairing, we applied an extra processing step to remove clonotypes that shared the same nucleotide sequence for either the α or the β chain, as previously described [22]. After filtering for functional TCRs using tidytcells, a TCR gene symbol standardiser [3], we retained 842,683 distinct clonotypes.

A random sub-sample of 10% of this data was reserved for use as an unseen test set, containing 84,268 unique clonotypes distributed across 83,979 unique TCRs. Of the remaining 90% of the data, we filtered out any clonotypes with amino acid sequences that also appeared in the test set, resulting in a training set of 753,838 unique clonotypes across 733,070 unique TCRs.

Procedure

SCEPTR was jointly optimised for MLM and autocontrastive learning, where the total loss of a training step was calculated as the sum of the MLM and autocontrastive (Eq. 4.4) losses.

We implemented MLM following established procedures [30]. Namely, 15% of input tokens were masked, and masked tokens had an 80% probability of being replaced with the `<mask>` token, a 10% probability of being replaced by a randomly chosen amino acid distinct from the original, and a 10% probability of remaining unchanged. The MLM loss was computed as the cross-entropy between SCEPTR's predicted token probability distribution and the ground truth.

Our choice of autocontrastive loss function is inspired by related work in NLP [107] and computer vision [104], but adapted to the TCR setting. Let $B = \{\tau_i\}_{i=1}^{N}$ be a minibatch of N TCRs. We generate two independent "views" of each TCR τ_i by passing two censored-variants of the same receptor through the model. Our censoring procedure removes a random subset of a fixed proportion (20%) of the residues from the tokenised representation of the CDR loops and with a 50% chance drops either the full α or β chain. To ensure that censoring does not fundamentally alter the underlying TCR sequence, the positional encoding for each token remains

fixed relative to the original TCR. In addition to the random censoring, views also differ due to dropout noise during independent model passes. Taken together, this procedure maps the minibatch B to the set of $2N$ TCR views $V = \{v_j\}_{j=1}^{2N}$, where v_{2i} and v_{2i-1} are two independent views of the same TCR τ_i ($i \in \{1...N\}$). Where $k \in I = \{1...2N\}$ is an arbitrary index of a view $v_k \in V$, let $p(k)$ be the index of the other view generated from the same TCR, and $N(k) = \{l \in I : l \neq k\}$ be the set of all indices apart from k itself. Let $\mathbf{r}_k$ denote SCEPTR's vector representation of TCR view v_k. Then the autocontrastive loss for minibatch B is computed as follows:

$$\mathcal{L}_{\text{AC}}(B) = \frac{1}{2N} \sum_{k \in I} -\log \frac{e^{\mathbf{r}_k^\top \mathbf{r}_{p(k)}/t}}{\sum_{n \in N(k)} e^{\mathbf{r}_k^\top \mathbf{r}_n/t}} \tag{4.4}$$

Here, t is a temperature hyper-parameter which we set to 0.05 during training, following previous literature [107].

We used ADAM (adaptive moment estimation) [122] to perform stochastic gradient descent. We chose a minibatch size of 1024 samples and trained for 200 epochs, which equated to 143,200 training steps. The internal dropout noise of SCEPTR's self-attention stack was set to 0.1.

Our methodology of randomly censoring residues and even entire chains stands in contrast to previous work in NLP by Gao, Yao, and Chen, who found that relying only on the internal random drop-out noise of the language model was sufficient for effective autoconstrastive learning. However, our experiments suggest that in the TCR domain, residue and chain censoring leads to embeddings with better downstream TCR specificity prediction performance (Fig. A.8).

4.4.4 Using SVCs to learn PLM features

To train the linear SVCs on top of PLM features, we sampled 1000 random background TCRs from the training partition of the unlabelled Tanno et al. dataset. We employed a similar strategy to our benchmarking in section 4.2.1 to split our dataset of curated specificity-annotated $\alpha\beta$TCRs into a reference set and testing set. For each PLM-pMHC-split combination, we trained a linear SVC using the PLM embeddings of the reference TCRs as the positives and those of the 1000 background TCRs as the negatives. The same 1000 background TCRs were used across model-pMHC-split combinations to ensure consistency. We accounted for the imbalance between the number of positive and negative samples used during SVC fitting by weighting the penalty contributions accordingly. Finally, we

tested SVCs using the same benchmarking classification task as previously described.

4.4.5 SCEPTR fine-tuning with sup. contrastive learning

Data

For supervised contrastive fine-tuning we took all TCR binders against the six best-sampled pMHC targets from our labelled TCR dataset, and split them into a training, a validation, and a test set such that no study used to construct the training or validation sets contributed any TCRs to the test set (Table A.2).

Procedure

The fine-tuning process involved the joint optimisation of SCEPTR on MLM and supervised contrastive learning. As during pre-training, the overall loss for each training step was computed as the unweighted sum of the MLM and supervised contrastive (Eq. 4.5) losses. The pre-trained state of SCEPTR was used as the starting point for fine-tuning. With only 200 TCRs for each target pMHC to train on, we limited the number of learnable parameters by only allowing the weights of the final self-attention layer to be trainable. Additionally, we monitored increases in validation loss for early stopping of fine-tuning, which occurred after 2 epochs, where one epoch is defined as the model seeing 100,000 binders for each pMHC. Given our a batch size of 1,024 TCRs, this corresponded to a total of 1,172 training steps.

Our implementation of supervised contrastive learning closely follows the formulation suggested by Khosla et al. [104]. This approach to supervised contrastive learning combines loss contributions from true positive pairs, with those from second views of each positive instance (as in autocontrastive learning) as well as all views of all other sample points with the same pMHC label. Let $B = \{\tau_i\}_{i=1}^{N}$ be a minibatch of N pMHC-annotated TCRs. We use the same procedure as in our autocontrastive framework (see methods 4.4.3) to generate two views of each of the TCRs, producing a set of $2N$ views $V = \{v_j\}_{j=1}^{2N}$. Let $Y = \{y_i\}_{i=1}^{N}$ be the index-matched pMHC labels for TCRs in B, and $\bar{y}_j$ denote the labels mapped to the indices of the views in V such that $\bar{y}_{2i} = \bar{y}_{2i-1} = y_i$. Now given arbitrary sample view index k, let $P(k) = \{l \in A(k) : \bar{y}_l = \bar{y}_k\}$ be the set of all indices whose corresponding samples have the same pMHC label as v_k, with cardinality $|P(k)|$. The supervised contrastive loss for TCR

minibatch B is:

$$\mathcal{L}_{\text{SC}}(B) = \frac{1}{2N} \sum_{k \in I} \frac{1}{|P(k)|} \sum_{p \in P(k)} -\log \frac{e^{\mathbf{r}_k^\top \mathbf{r}_p / t}}{\sum_{n \in N(k)} e^{\mathbf{r}_k^\top \mathbf{r}_n / t}} \tag{4.5}$$

Each batch during fine-tuning has an equally balanced number of binders to each of the six pMHCs.

4.5 Code Availability

`https://github.com/yutanagano/sceptr` This is a readily usable deployment of SCEPTR and all its variants seen in this publication.

`https://github.com/yutanagano/tcrlm` This repository houses code used for designing and training our models.

`https://github.com/yutanagano/libtcrlm` This repository houses some library code that powers both the `sceptr` and `tcrlm` repositories above.

Chapter 5

Calibrating SCEPTR's distances to a probability of specificity

Robust model calibration and uncertainty quantification is an important aspect of machine learning, especially for domains like TCR analysis where the ultimate aim is for models to be applied to downstream medical applications. In chapter 4, we saw that distance-based TCR specificity predictions – in particular those made using the nearest neighbour prediction scheme – remain competitive in the few-shot setting, in which most real-world applications of TCR modelling technologies will have to operate. However, none of the available distance-based TCR models have any way to calibrate observed TCR distances to posterior probabilities of (co-)specificity. In this chapter, I present some early work on Bayesian nearest neighbour association (BANANAS), a framework to convert distances from a query TCR of unknown specificity to its nearest neighbour in a reference set of known binders (see section 4.2.1) to the posterior probability that the query TCR is specific to the pMHC of interest. BANANAS makes use of the machinery proposed by Mayer and Callan for the study of TCR coincidence statistics (see [22] and section 2.3) and should theoretically be able to convert TCR distances as measured by any arbitrary TCR metric[1] into a posterior probability of specificity. In my preliminary experiments shown later in section 5.2, I focus on applying BANANAS to SCEPTR's distances. This work is jointly led by myself alongside Sankalan Bhattacharyya and James Henderson, under the supervision of Andreas Tiffeau-Mayer. All of the writing and analysis presented in this chapter is my own.

[1] By *metric* here I mean a distance metric – that is, some symmetric and positive definite function that can measure distances between TCRs.

5.1 Theory

To recap, making inferences using the nearest neighbour prediction scheme requires three things: a TCR metric d, a set of query TCRs $\mathcal{Q}$, and a set of known binder (reference) TCRs $\mathcal{R}$ to the pMHC of interest π. The TCR metric is used to measure the distance from every query TCR in $\mathcal{Q}$ to its nearest neighbour in $\mathcal{R}$, and this distance is used to generate inferences about whether each query TCR is also likely to be binder to π.

5.1.1 Posterior probability of specificity

We will begin by constructing a simple model for the distribution of TCR sequences in the query set $\mathcal{Q}$. Let p_B and $p_{S|\Pi}(\cdot|\pi)$ be two probability distributions defined over the set of all TCRs, where p_B is the background distribution and $p_{S|\Pi}(\cdot|\pi)$ is the distribution of binders to π. Suppose that $\mathcal{Q}$ is a sample from a mixture distribution where a fraction $\lambda \in [0, 1]$ of the TCRs are "spiked-in" binders to π and the rest are background sequences. Then, a random sample Q from $\mathcal{Q}$ will be distributed accordingly:

$$Q \sim p_Q, \quad p_Q(\tau) = \mathop{\mathbb{E}}_{z \sim p_Z} \left[p_{Q|Z}(\tau|z) \right] \tag{5.1}$$

where:

$$p_{Q|Z}(\tau|z) = \begin{cases} p_{S|\Pi}(\tau|\pi) & \text{if} \quad z = 1 \\ p_B(\tau) & \text{if} \quad z = 0 \end{cases} \tag{5.2}$$

$$p_Z(z) = \text{Bernoulli}(\lambda) \tag{5.3}$$

Notice that the latent variable Z acts as a "label" for specificity, since $Z = 1$ indicates that Q was sampled from the spike-in fraction of $\mathcal{Q}$, and vice versa.

Let us start with the simplest case, and imagine that $\mathcal{R}$ consists of a single known binder TCR $R \sim p_{S|\Pi}(\cdot|\pi)$. We will denote the distance between our query and reference as $\Delta = d(Q, R)$. Note that since Q and R are both random variables, Δ is also a random variable.

We can approximate[2] the probability that Q is also a binder to π with the probability that Q was sampled from the spike-in fraction of $\mathcal{Q}$, which is equivalent to the probability that $Z = 1$. Given that we observe some

[2]The reason why this is technically an approximation, and the underlying theoretical limitation of this framework is further expanded on in the later discussion section 5.3.

distance δ between Q and our single reference R, this is:

$$P(Z=1|\Delta=\delta) = \frac{P(\Delta=\delta|Z=1)P(Z=1)}{P(\Delta=\delta)} \tag{5.4}$$

$$= \frac{1}{1+\frac{P(\Delta=\delta|Z=0)P(Z=0)}{P(\Delta=\delta|Z=1)P(Z=1)}} \tag{5.5}$$

Now, we can use the idea of near-coincidence distributions from Mayer and Callan to express the conditional probability distributions of Δ:

$$P(\Delta=\delta|Z=1) = p_C[p_{S|\Pi}(\cdot|\pi)](\delta) \tag{5.6}$$

$$P(\Delta=\delta|Z=0) = \sum_{\tau\in\mathcal{T}}\sum_{\tau'\in\mathcal{T}} p_B(\tau)p_{S|\Pi}(\tau'|\pi)I_{d(\tau,\tau')=\delta} \approx p_C[p_B](\delta) \tag{5.7}$$

where $\mathcal{T}$ denotes the set of all TCRs. Equation 5.6 follows from the fact that $Z=1$ implies both Q and R are sampled independent and identically distributed (IID) from the binder distribution $p_{S|\Pi}(\cdot|\pi)$. The motivation behind the approximation in equation 5.7 can be seen in appendix C. Finally, $P(Z=1)=\lambda$ and $P(Z=0)=1-\lambda$. This gives us all the pieces necessary to evaluate the posterior probability of specificity:

$$P(Z=1|\Delta=\delta) = \frac{1}{1+\chi(\delta)^{-1}\mathbb{O}^{-1}} \tag{5.8}$$

where:

$$\chi(\delta) := \frac{p_C[p_{S|\Pi}(\cdot|\pi)](\delta)}{p_C[p_B](\delta)} \tag{5.9}$$

$$\mathbb{O} := \frac{\lambda}{1-\lambda} \tag{5.10}$$

This simple model assumes that there exists only one reference TCR against which we are measuring a distance. With greater numbers of references, the probability of spuriously detecting smaller distances will increase. Therefore, this model may overestimate the posterior probabilities in such scenarios. In the following section, I propose a way to correct for this behaviour.

5.1.2 Correction for number of reference TCRs

Let us now consider the case where we have $K>1$ reference sequences, where each reference is IID with $p_{S|\Pi}(\cdot|\pi)$:

$$\mathcal{R} = \left\{R_i : R_i \sim p_{S|\Pi}(\cdot|\pi)\right\}_{i=1}^{K} \tag{5.11}$$

We can now define a set $\mathcal{D}$ of K distances between Q and each element $R_i \in \mathcal{R}$:

$$\mathcal{D} = \{\Delta_i : \Delta_i = d(Q, R_i)\}_{i=1}^{K} \tag{5.12}$$

Note that for all i, $\Delta_i \sim \Delta$, with $\Delta = d(Q, R)$ from the earlier simpler case considering only one reference TCR.

Let $\Delta_{(1)}$ be the minimum of $\mathcal{D}$ – that is, the distance from Q to its nearest neighbour in $\mathcal{R}$. Then:

$$P(\Delta_{(1)} = \delta | Z = 1) = KP(\Delta = \delta | Z = 1)P(\Delta > \delta | Z = 1)^{K-1} \tag{5.13}$$

$$= Kp_C[p_{S|\Pi}(\cdot|\pi)](\delta) \left[1 - \int_0^\delta p_C[p_{S|\Pi}(\cdot|\pi)](t)dt\right]^{K-1} \tag{5.14}$$

Note that the integral above would be replaced by a sum for discrete distance metrics. Similarly:

$$P(\Delta_{(1)} = \delta | Z = 0) \approx Kp_C[p_B](\delta) \left[1 - \int_0^\delta p_C[p_B](t)dt\right]^{K-1} \tag{5.15}$$

The above is an approximation for the same reason as equation 5.7 (see appendix C).

Thus, the posterior probability of specificity given the distance to the nearest neighbour reference corrected for multiple reference binders is:

$$P(Z = 1 | \Delta_{(1)} = \delta) = \frac{1}{1 + \chi(\delta)^{-1} \left(\frac{1 - P_C[p_B](\delta)}{1 - P_C[p_{S|\Pi}(\cdot|\pi)](\delta)}\right)^{K-1} \mathbb{O}^{-1}} \tag{5.16}$$

where:

$$P_C[p](\delta) \coloneqq \int_0^\delta p_C[p](t)dt \tag{5.17}$$

5.1.3 Model tractability

With equation 5.16, we now have a calibration model for the posterior probability for specificity. But is the model tractable? Using available datasets of specificity-annotated and background TCRs, we should be able to obtain empirical estimates for the cumulative distribution functions (CDFs) $P_C[p_{S|\Pi}(\cdot|\pi)](\delta)$ and $P_C[p_B](\delta)$. This would allow us to estimate the correction term for having multiple reference TCRs. Furthermore, we should also be able to obtain estimates for $p_C[p_{S|\Pi}(\cdot|\pi)](\delta)$

and $p_C[p_B](\delta)$ either directly or by differentiation of our estimates of the above CDFs, which allows us to estimate $\chi(\delta)$. The only free term that remains is the prior odds, $\mathbb{O}$.

Given a set of query TCRs $\mathcal{Q}$ and estimates for $P_C[p_{S|II}(\cdot|\pi)](\delta)$ and $P_C[p_B](\delta)$, we can self-consistently generate an estimate for $\mathbb{O}$ using the near-coincidence statistics of $\mathcal{Q}$ by considering the following. According to our model for the distribution of the elements of $\mathcal{Q}$ (Eq. 5.1), the distribution of distances within elements of $\mathcal{Q}$ should be:

$$p_C[p_Q](\delta) = \sum_{\tau\in\mathcal{T}}\sum_{\tau'\in\mathcal{T}} \mathop{\mathbb{E}}_{z\sim p_Z}\left[p_{Q|Z}(\tau|z)\right] \mathop{\mathbb{E}}_{z\sim p_Z}\left[p_{Q|Z}(\tau'|z)\right] I_{d(\tau,\tau')=\delta} \tag{5.18}$$

$$= \sum_{\tau\in\mathcal{T}}\sum_{\tau'\in\mathcal{T}} \Big(\lambda^2 p_{S|II}(\tau|\pi)p_{S|II}(\tau'|\pi) + 2\lambda(1-\lambda)p_{S|II}(\tau|\pi)p_B(\tau') + (1-\lambda)^2 p_B(\tau)p_B(\tau')\Big) I_{d(\tau,\tau')=\delta} \tag{5.19}$$

$$\approx \lambda^2 p_C[p_{S|II}(\cdot|\pi)](\delta) + (1-\lambda^2)p_C[p_B](\delta) \tag{5.20}$$

where line 5.19 above is obtained from the fact that:

$$\mathop{\mathbb{E}}_{z\sim p_Z}\left[p_{Q|Z}(\tau|z)\right] = \lambda p_{S|II}(\tau|\pi) + (1-\lambda)p_B(\tau)$$

and the approximation in line 5.20 is the same as that used for equation 5.7 (see appendix C). Along similar lines, we get:

$$P_C[p_Q](\delta) \approx \lambda^2 P_C[p_{S|II}(\cdot|\pi)](\delta) + (1-\lambda^2)P_C[p_B](\delta) \tag{5.21}$$

Therefore, by fitting either equation 5.20 or 5.21 to the empirical distribution of distances within TCRs from $\mathcal{Q}$ with λ as the free parameter, we get an estimate of the spike-in fraction of $\mathcal{Q}$ which is comprised of specific binders. Since $\mathbb{O}$ depends only on λ, we also get an estimate for the odds.

5.2 Preliminary experiments

With the theory developed, I conducted some preliminary experiments to investigate the quality of calibrations that the framework could produce using SCEPTR as the underlying TCR metric.

5.2.1 Estimating SCEPTR's near-coincidence statistics

The first order of business was to obtain estimates for TCR near-coincidence statistics using SCEPTR's distances. Due to the continuous (as opposed to discrete) nature of SCEPTR distances[3], I chose to obtain empirical estimates for the CDFs ($P_C[\cdot](\cdot)$), fit a parametric model to them, and differentiate the parametric models to obtain the respective density functions ($p_C[\cdot](\cdot)$). Using specificity-annotated TCR data from Minervina et al. [25], I generated empirical estimates for the cumulative near-coincidence curves for the binder distributions against several different pMHCs, shown through solid grey and orange lines in figure 5.1. The curve for each individual pMHC is shown in transparent grey, while an average over all the pMHCs considered is shown in orange. I also used 10,000 background TCRs generated by a synthetic model of VDJ recombination [7] to estimate the background distance statistics, which is shown using a solid purple line (Fig. 5.1).

Plotting these empirical estimates of the near-coincidence CDFs shows two things. Firstly, the background curve seems to display a steady exponential increase until plateauing around a distance of 1.5. This is a hint that the CDF may be well-approximated by a logistic function. Secondly, we see that there is some variation in the co-specific near-coincidence curves between different pMHCs. However, especially in the regime of very small distances, the inter-pMHC variation is a couple of orders of magnitude smaller than the difference between the average pMHC-specific (orange) curve and the background (purple) curve. This motivates the approximation:

$$p_C[p_{S|II}(\cdot|\pi)](\delta) \approx \mathop{\mathbb{E}}_{\pi'}\left[p_C[p_{S|II}(\cdot|\pi')](\delta)\right] \tag{5.22}$$

for any pMHC of interest π, which allows us to fit our BANANAS model once to the average pMHC-specific curve, and reasonably apply it to predictions against multiple different pMHCs.

As I have alluded to in the previous paragraph, the CDF for the background near-coincidence probabilities closely resembles a logistic function. Therefore, I fitted to it the model shown in equation 5.23, which is a logistic function corrected such that $\text{logistic}_{\text{corr}}(2) = 1$. The correction

[3]While SCEPTR's distances are indeed continuous with respect to its model parameters, the fact that the TCR space is discrete means that SCEPTR's distances are still technically discrete with respect to its TCR inputs. That being said, the set of all possible discrete distances across all TCRs is intractable. Therefore, we will approximate SCEPTR distances as being continuous for the rest of this chapter.

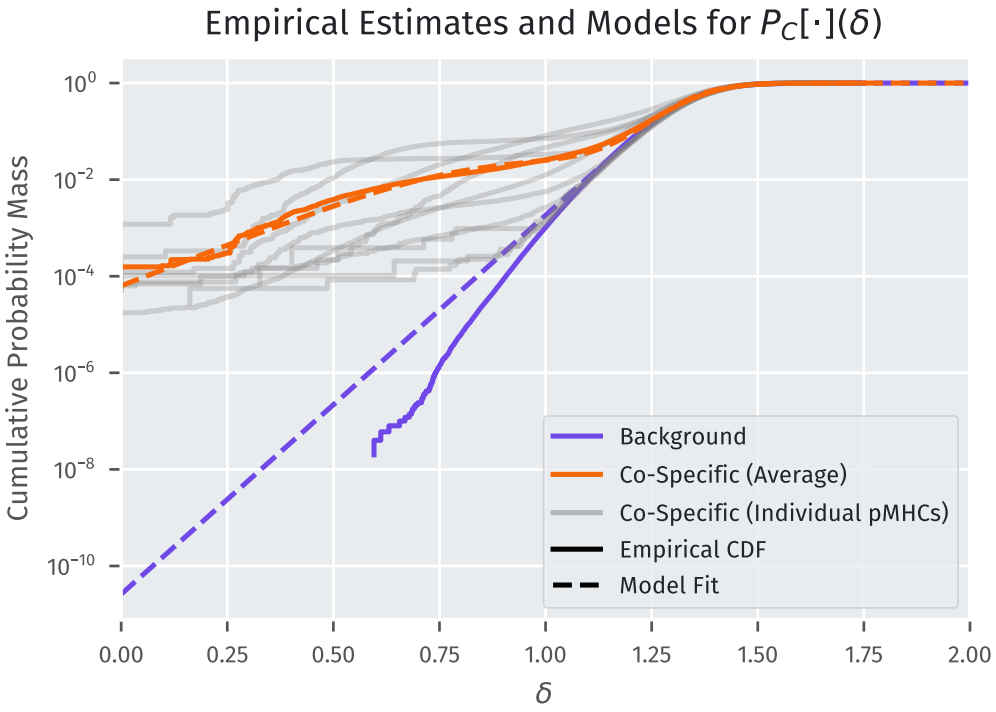


Figure 5.1: **Empirical estimates and model fits to the cumulative near-coincidence probability distributions $P_C[\cdot](\cdot)$ of SCEPTR.** The purple curve shows the CDF for the background distribution, whereas the orange curve shows the CDF for co-specific TCRs averaged over the 10 pMHCs in the Minervina et al. dataset which had more than 100 known binder TCRs [25]. The transparent grey lines show the curves individually for each of the 10 pMHCs. The solid lines represent empirical estimates of the CDFs, while the dotted lines show model fits to the background and co-specific CDFs. The y axis is logged and shows the cumulative probability of sampling distances less than or equal to a certain value. The x axis shows the distances.

is applied because a CDF must evaluate to one at the maximum of the distribution's domain (if it exists), and the maximum possible Euclidean distance within SCEPTR's representation space is 2 (recall that SCEPTR's representations live on the unit hypersphere).

$$\text{logistic}_{\text{corr}}(x; a, b) = \frac{1}{1 + e^{-a(x-b)} - \epsilon}, \quad \epsilon := e^{-a(2-b)} \tag{5.23}$$

The average co-specific near-coincidence CDF appears to behave very similarly to the background distribution at high distances, but is enriched for smaller distances. In fact, the co-specific curve seems to have a shallower exponential decay rate from roughly distance 1 towards 0 compared to the background curve. I decided to model this with a mixture of logistic curves:

$$F_{\text{cospec}}(x; l, a, b) = l \times \text{logistic}_{\text{corr}}(x; a, b) + (1 - l) \times F_{\text{back}}(x) \tag{5.24}$$

where F_{back} is fixed, and is the result of fitting the corrected logistic function to the background CDF.

The result of both fits are shown in figure 5.1 as the dotted lines. Although the model fit to the background curve tends to overestimate the cumulative probability mass at smaller distances, the two model fits are reasonably close to the empirical data. Differentiating the model fits with respect to the distance inputs x and taking the ratio of the co-specific to the background curve gives us an estimate for χ, shown in figure 5.2. Since my model fit overestimates the probability of small distances in the background distributions, one may expect the model-based approximation of χ to underestimate its value in this regime. To investigate whether this was the case, I compared the model-based approximation of χ to a more direct estimate, which was obtained by first discretising SCEPTR distances through binning, then computing direct estimates of the (now probability *mass*-) functions $p_C[p_{S|\Pi}](\delta)$ and $p_C[p_B](\delta)$, and finally taking their ratio. The results of this direct estimation is shown with orange crosses in figure 5.2. As expected, the model-based approximation of χ underestimates its value compared to the direct statistics obtained from the data. However for the purposes of my preliminary experiments, I decided to use the model-based approximation, since it extrapolates to regions of very low and very high distances, where there is a lack of sufficient data for direct estimation.

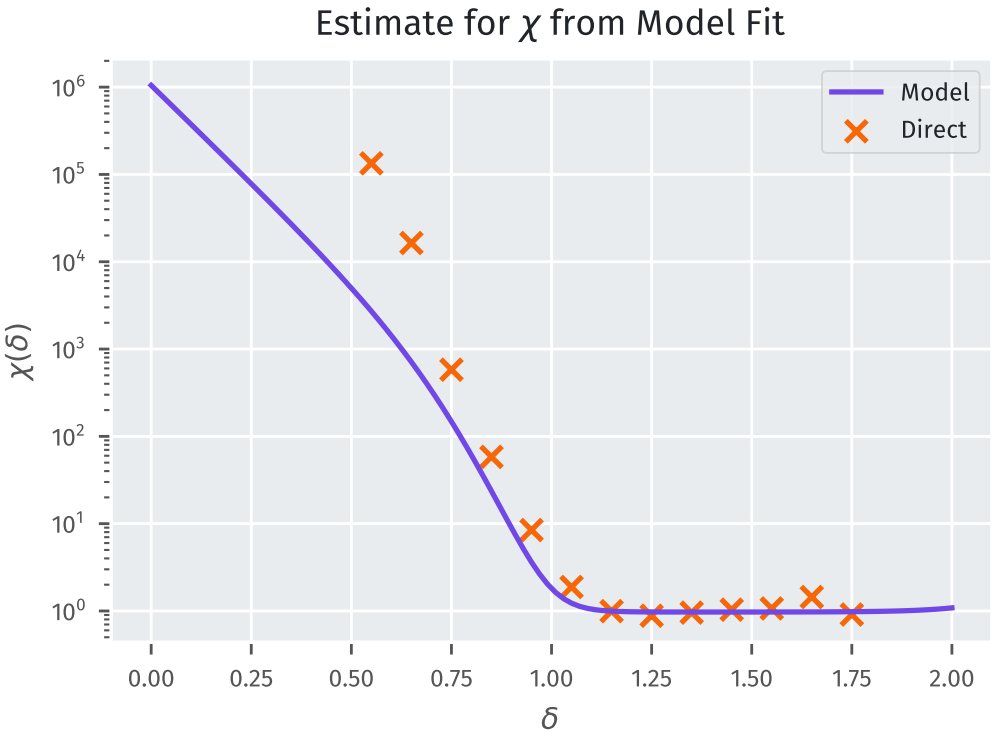


Figure 5.2: **Estimate for χ according to the model fits to the cumulative $P_C[\cdot](\cdot)$ distributions.** The purple curve shows the estimate for $\chi(\delta)$ obtained by taking the ratio of the derivates of the model fits to the co-specific and background cumulative near-coincidence probability distributions. The orange crosses show a more direct estimate of $\chi(\delta)$ which was obtained by discretising SCEPTR distances into bins of width 0.1 from $\delta = 0$ to $\delta = 2$, directly estimating the probability mass functions of the discretised distances for the background and co-specific distributions, and taking the ratio. The y axis corresponds to the value of $\chi(\delta)$, while the x axis corresponds to the distance δ. The model fit underestimates the value of χ at smaller distances.

5.2.2 Uncorrected model

Using the obtained estimates for χ, $P_C[p_{S|\Pi}](\cdot)$ and $P_C[p_B](\cdot)$, I conducted some empirical calibration studies of the BANANAS model. At the current preliminary stage, I chose to conduct these evaluations with the same datasets used to fit the model. This was done to ensure that the data statistics during model evaluation were kept similar to those during model fitting, such that any inaccuracies in the resulting calibration can be attributed to inherent problems with the model itself, such as incorrect choices of approximations, or bad fits.

A calibration test consisted of 100 repeated and random constructions of a reference set of K known TCR binders against a particular pMHC target π, and a query set of 200 TCRs of which some spike-in fraction λ were also binders to π, and the rest were sampled from the background set. All sampling during the reference and query set construction were done without replacement, which ensured that none of the TCRs used in a particular reference set would ever appear in its corresponding query set. Within each split, the BANANAS model was used to compute the probabilities of specificity for all members of the query set. These probability scores, along with the true label of the corresponding TCR, were collected across all splits, and used to compute a calibration curve. A calibration curve compares the predicted probabilities of specificity with the empirical fraction of truly specific TCRs that were assigned such scores. I conducted calibration tests for the 5 different pMHCs in the Minervina et al. dataset that had more than 200 known binders, across 7 different settings for the number of reference sequences $K \in \{1, 2, 5, 10, 20, 50, 100\}$, and 4 different settings for the spike-in fraction of positive query sequences $\lambda \in \{0.01, 0.05, 0.10, 0.50\}$.

In these preliminary calibration experiments, the BANANAS model was told the true prior $\lambda/(1-\lambda)$. This was done again to separate the issue of the quality of the model itself from that of prior estimation. I will address the issue of testing how well we can use the distance statistics to infer the prior odds in the later section 5.2.4.

Let us begin by examining the calibration of the BANANAS model without the correction for multiple reference sequences – that is, the model shown in equation 5.8. The calibration plot for this model is shown in figure 5.3. Firstly, we can see that the predicted probability scores do not scale linearly with the true posterior probabilities. In particular, the model tends to relative under-confidence towards the upper end of the posterior probabilities. At the same time, we also see that the model becomes generally overconfident in its predictions during testing condi-

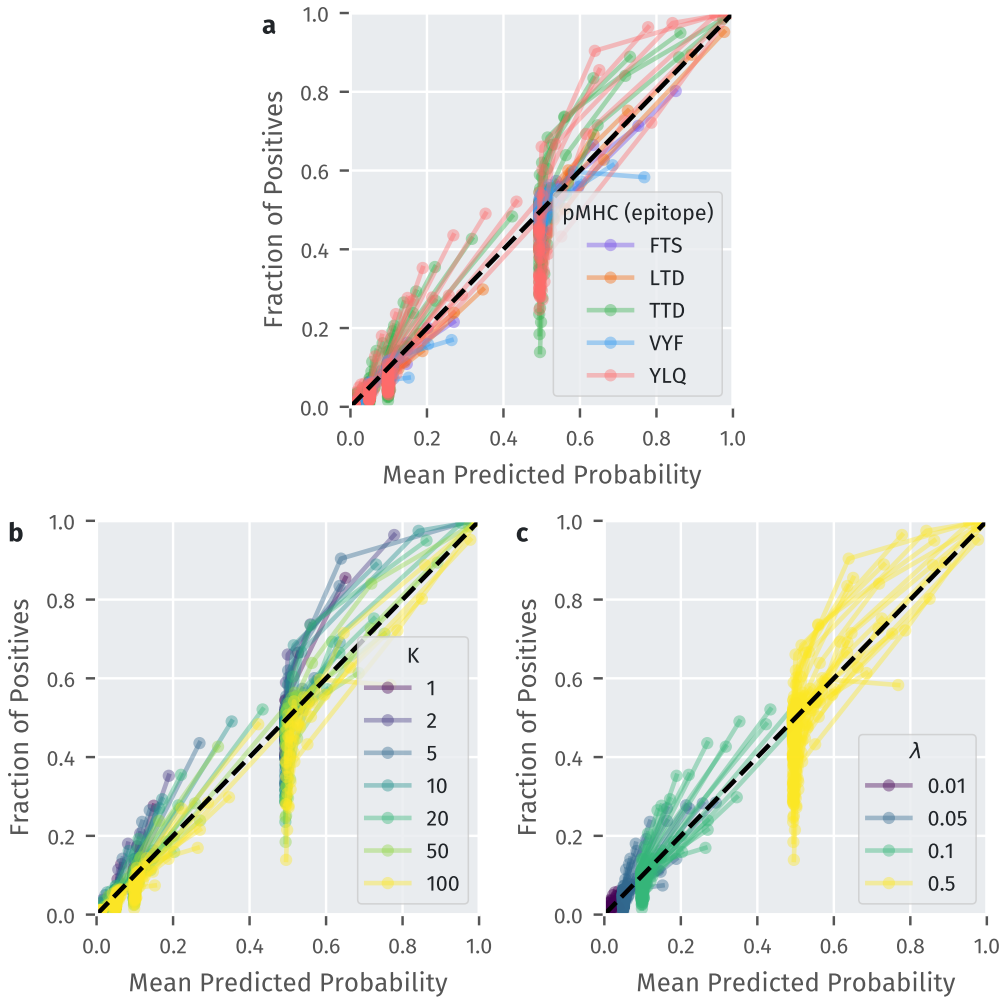


Figure 5.3: **Calibration plot for the uncorrected BANANAS model.** This plot visualises the calibration of BANANAS without correcting for the multiplicity of reference sequences (Eq. 5.8). Within a calibration experiment, all test samples were collected into deciles according to their predicted probability scores. Within each decile group, the mean predicted probability (x axis) was calculated, as well as the fraction of true positives (y axis). One curve is drawn per experiment. The curves are coloured according to: **a)** the pMHC specificity of interest, **b)** the number of references K, and **c)** the spike-in fraction λ. The black dotted line shows the $y = x$ diagonal, representing a theoretical perfectly-calibrated model. The uncorrected BANANAS model generates overconfident predictions when run with greater numbers of reference sequences.

tions with greater numbers of reference TCRs. This is expected, since this model does not correct for the multiplicity of reference TCRs. Next, let us inspect what the calibration curves look like for the corrected model.

5.2.3 Model corrected for reference multiplicity

The calibration plot for the corrected model is shown in figure 5.4. These preliminary results show that the correction for multiple reference sequences does indeed help reduce overconfidence in the model. However, this comes at the cost of accentuating the model's tendency to increasingly generate under-confident predictions towards higher true fractions of positives.

5.2.4 Inferring prior odds

In section 5.1.3, I proposed a way to self-consistently infer the spike-in fraction of the query set, and hence the prior odds $\mathbb{O}$. How well does this work in practice? Figure 5.5 shows some empirical results of implementing the proposed method, where I have summarised the results separately according to the pMHC specificity of interest. As it stands, the accuracy of the inferences is poor. Furthermore, we also see that there is a large variance in the predicted spike-in fractions between similar conditions across different pMHCs.

5.3 Discussion

In this chapter, I have presented some preliminary work on BANANAS, a statistical framework through which to calibrate nearest neighbour distance-based predictions of TCR specificity. The results presented demonstrate that a prototype implementation of the framework on SCEPTR's distances can produce monotonically increasing calibration profiles that remain within reasonable distance to the diagonal. Furthermore, there are promising signs that correcting for the multiplicity of reference sequences does indeed improve calibration – notably, it makes it less likely for the model to generate over-confident predictions. However, the quality of calibration in this prototype implementation is currently poor, especially considering the fact that the preliminary evaluations were conducted using the same dataset to which the model was fit. Below I discuss some of the limitations of the current work and possible ideas for addressing these issues, as well as points for further investigation.

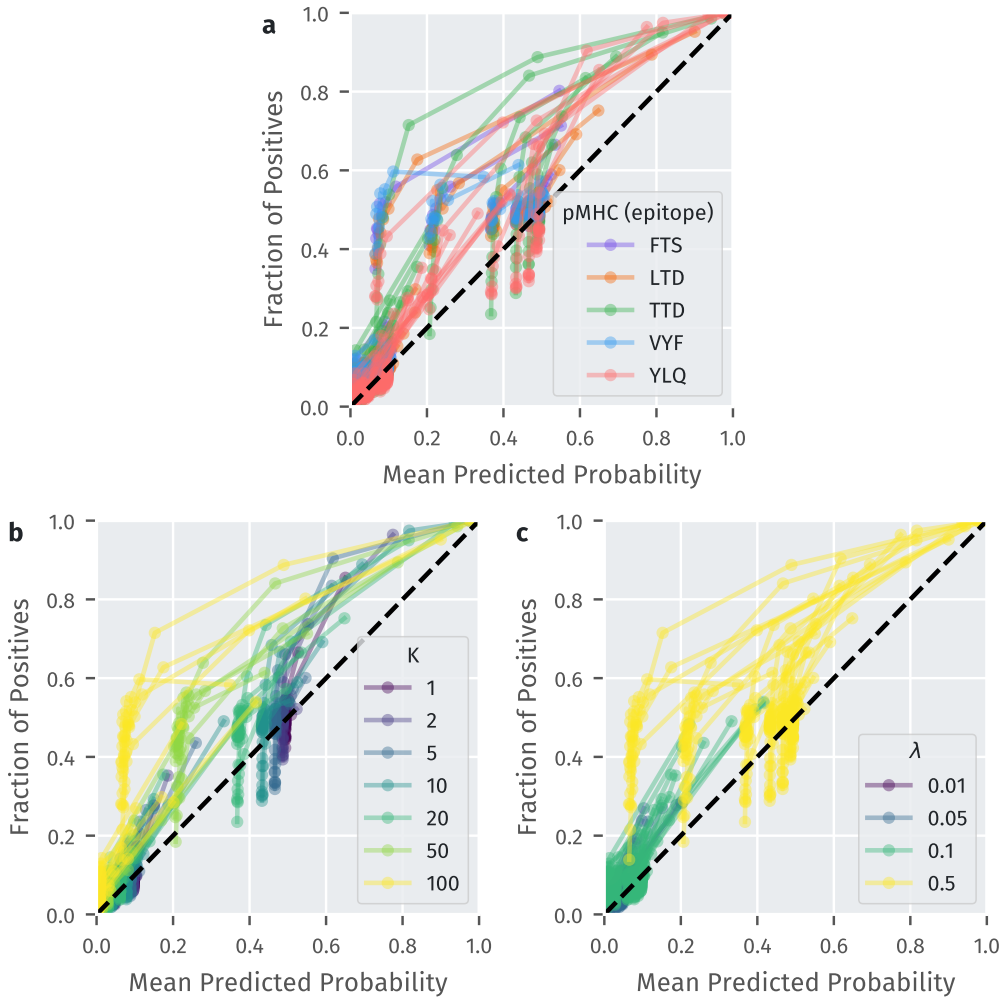


Figure 5.4: **Calibration plot for the full BANANAS model with a correction term for the multiplicity of reference TCRs.** This plot is complementary to figure 5.3, but shows results for BANANAS with the correction term for the multiplicity of reference TCRs (Eq. 5.16). In settings with larger K, adding the correction term shifts the calibration curves to the left, making the model less likely to produce overconfident estimates of the posterior probability of specificity. However, this comes at the expense of increased under-confident predictions in the regime of high values for the true fraction of positives.

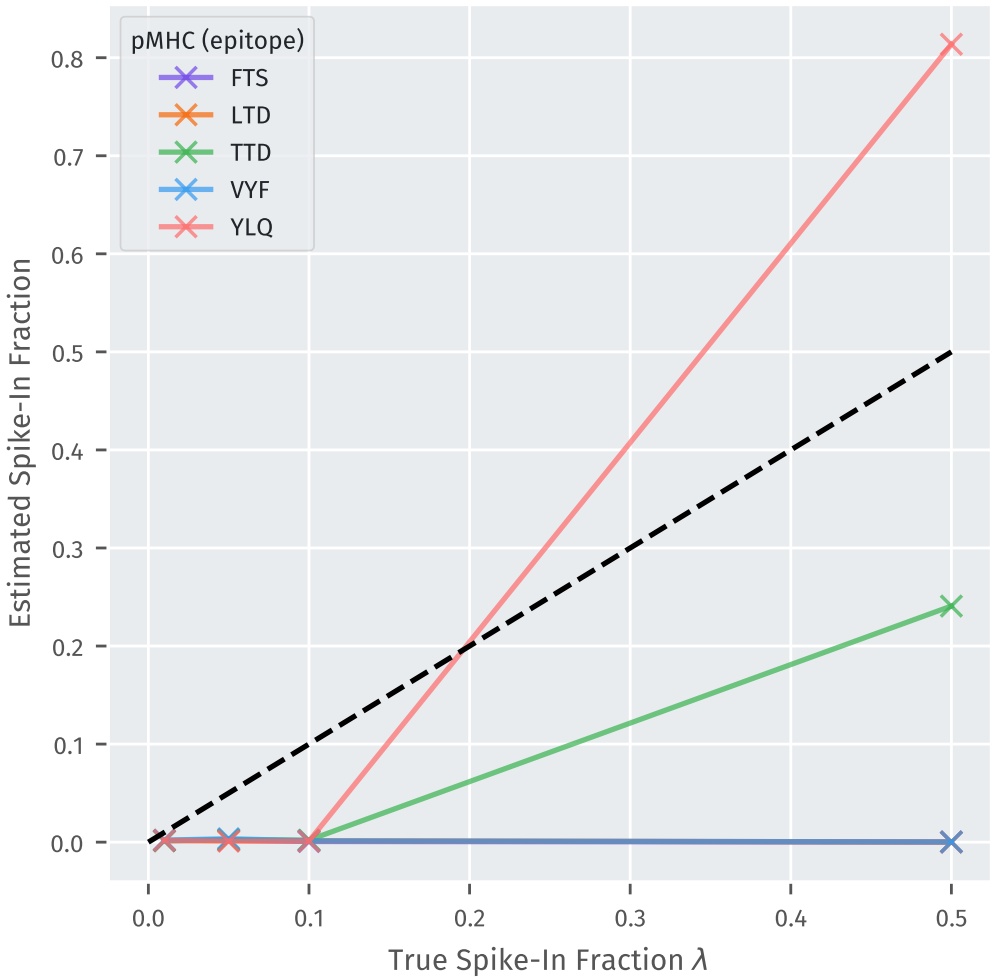


Figure 5.5: **Estimating the spike-in fraction λ of true positives in the query set.** This plot visualises the results of estimating the fraction of the query set that is comprised of co-specific binders using the model from equation 5.21 and the near-coincidence statistics within the query set. The results are shown separately for calibration experiments using binder data for different pMHCs. The y axis shows the mean predicted value for λ using the proposed method, while the x axis shows the true value for λ. The dotted black line shows the $y = x$ diagonal, on which perfect predictions would lie. The results of the proposed method produce inaccurate predictions, and there is a large degree of variance seen between calibration experiments using data from different pMHCs.

Both the uncorrected and corrected BANANAS models tend to make under-confident predictions towards the upper end of the posterior probabilities. While this can be due to multiple reasons, the most likely culprit is the inaccuracies in my approximation of χ. In particular, my model-based approximation of $\chi(\delta)$ underestimates its value at small δ, which would have led to lower posterior probabilities in that region. Since the query sequences with lowest δ should be the ones that are most likely to be truly positive, this may explain both models' relative under-confidence in this regime. One way to improve in this regard may be to use more flexible classes of models to estimate the CDFs $P_C[\cdot](\cdot)$. One promising candidate is the cubic basis splines: as well as providing the flexibility to approximate arbitrary shapes, they also guarantee the existence of a smooth first derivative, which is desirable if we choose to estimate χ using the first derivatives of the fit curves. It is worth noting here that an application of BANANAS to simpler metrics such as Levenshtein distance on the CDR3 sequences may pose less difficulties in estimating the near-coincidence probability distributions. Firstly, fully discrete metrics like Levenshtein distance are more coarse-grained in nature, which means each possible distance value will have more empirical data-points associated to it. Secondly, such metrics where the two chains of the TCR independently contribute to the distance allow for factorisation assumptions to be made when estimating the background near-coincidence probability distributions, which is useful since the probabilities of encountering exact/very-near coincidences in the background distribution are extremely low.

My preliminary attempts at estimating the prior odds from data have not shown great success. However, this may partly be a result of the practical limitations of my experimental setup, where the size of the query set was fixed at a relatively small 200 TCRs. The reason for this was data volume. For some of the pMHCs investigated in the presented experiments, only around 200 binder TCRs were available. I wanted to keep the query set size consistent across experiments and data splits, to keep conditions as similar as possible. To do this while ensuring that up to half of the query set can be comprised of spiked-in positive TCRs, under the constraint that up to 100 binder TCRs must be reserved for the construction of a reference set, the query set size needed to be kept at 200. It is possible that in real-life scenarios where the query set is much larger, we would have enough statistics to more reliably estimate the spike-in frequency.

That being said, there are other possible reasons for the poor quality of the inferred spike-in fractions. For example, if the parametric models I fit to the co-specific and background near-coincidence distributions

are inaccurate as alluded to in the previous paragraphs, this may have precluded the model in equation 5.21 from reliably producing accurate results. Another possibility with wider consequences for the validity of BANANAS's theory is that the approximation in equation 5.20 (and also in Eq. 5.7):

$$\sum_{\tau \in \mathcal{T}} \sum_{\tau' \in \mathcal{T}} p_B(\tau) p_{S|\Pi}(\tau'|\pi) I_{d(\tau,\tau')=\delta} \approx p_C[p_B](\delta)$$

is not a good one. To investigate whether this is the case, an empirical study of the distance distributions between TCRs sampled from various pMHC-specific datasets and some background dataset should be conducted. Thirdly, it is worth remembering that data outside of TCR sequencing can also inform one's prior beliefs about the odds – for example, functional assays can reveal what fraction of T cells from the same sample show markers of activation after being stimulated by antigens that produce the pMHC of interest. Therefore, even if estimating the spike-in frequency using TCR sequencing data were to remain challenging, this would not preclude the utility of the BANANAS framework.

A nuanced theoretical limitation of the BANANAS framework is that the posterior probability computed, $P(Z = 1|\Delta = \delta)$, is only an approximation to the posterior probability that a query TCR is able to bind the pMHC of interest π. To better understand the reason why, let us remind ourselves of the meaning behind the posterior event of consideration $Z = 1$. In our model, Z is a latent variable which indicates whether the query TCR was sampled from the spike-in fraction of the query set whose elements follow the distribution $p_{S|\Pi}(\cdot|\pi)$ of binders to π, or the remaining portion of the query set that is distributed according to the background distribution p_B. Indeed if a query TCR is a sample from the spike-in fraction, then by definition it is a binder to π. Thus, $Z = 1$ implies that the query TCR is a binder. However, $Z = 0$ does *not* imply that the query TCR is not a binder. This is because by definition, the background distribution p_B should assign non-zero probabilities to all possible TCR sequences, including all binders to π. Thus, the probability being estimated by the BANANAS framework is fundamentally an approximation. That being said, it can be argued that in most if not all real-life scenarios where a method such as BANANAS may be deployed (e.g. examining the repertoire of an individual infected with a pathogen and fishing out the TCRs that are likely to be responding to a pMHC produced by said pathogen), the spike-in fraction λ for TCRs with the specificity of interest is likely to be orders of magnitude larger than the probability mass assigned to the set of binders in the background dis-

tribution. Therefore, in most situations, this approximation should be justified.

Finally, one of the biggest limitations of this preliminary work is the fact that the BANANAS model has only been tested within strictly controlled artificial conditions where many of its arguably non-trivial assumptions are made to be true. For example, the model assumes that the query set is a simple mixture of a background TCR distribution and a binder distribution to one pMHC specificity of interest. By the construction of my experimental setup, this assumption holds. However, the true statistics of query data distributions in the real world will never be this simple. Furthermore, the preliminary tests documented in this chapter were produced with the same dataset used to fit the parameters of the BANANAS model itself. Thus, further investigation is necessary in order to examine how model performance will generalise to independent datasets. The fact that my implementation of BANANAS still struggles with certain aspects of the calibration even in such controlled environments is evidence that there is much work left to be done.

In conclusion, BANANAS is a Bayesian framework that transforms nearest neighbour distance measurements to a posterior probability of specificity. I present a prototype implementation of this framework on top of SCEPTR's TCR distances. Preliminary results demonstrate that the BANANAS model produces monotonically increasing calibration curves, and provides a way to correct for the number of reference TCRs used, which can otherwise bias predicted probabilities towards over-confident values. However, the model transitions to significant under-confidence in regions of high posterior probability, which may be due to a poor estimation of the near-coincidence probabilities according to SCEPTR's distances. Additionally, a proposed method of self-consistent prior odds estimation produces results of poor quality. Further work is required to determine whether changes to the implementation can improve BANANAS model fit to SCEPTR's distance statistics, and if that can produce better-calibrated results. Finally, and crucially, more rigorous evaluation of this framework should be conducted using a wider diversity of datasets (with datasets that are independent of model training as an important first step) in order to gain insight into the generalisability of this framework. Model calibration and uncertainty quantification should be a key component of quantitative models of TCR specificity, given its potential impact in downstream biomedical applications. However, its implementation remains challenging in the face of limited specificity-annotated TCR data. Further thoughts around uncertainty quantification and probabilistic modelling of TCRs can be found section 6.3.2 of the executive discussion.

Chapter 6

Discussion

6.1 My contributions

In chapter 3, I presented my work on a piece of open-source software named tidytcells that standardises TCR and MHC nomenclature to make them compliant with the IMGT naming conventions. Through this work, I contribute a tool to the quantitative immunology community which makes the preparation of clean, standardised TCR and pMHC datasets easier and less error-prone. This is important for making TCR datasets more computer-readable – particularly when conducting analyses that involve comparing and contrasting different datasets (e.g. querying TCRs of unknown specificity against a database of specificity-annotated TCR data), where correct comparisons depend on a consistent encoding of the underlying data.

In chapter 4, I presented my work investigating autocontrastive learning as a strategy for pre-training PLMs. Through this work, I demonstrate that existing PLMs underperform at few-shot TCR specificity prediction compared to simpler and more traditional sequence alignment-based methods. In search of an explanation for this surprising finding, I speculate that MLM is ill-aligned to TCR specificity prediction, and demonstrate that the addition of an autocontrastive pre-training signal closes this performance gap. Within the context of recent trends in training expressive models using large volumes of unlabelled data, my findings highlight the importance of choosing well-aligned pre-training objectives in order to achieve high downstream performance. In addition to these insights, I contribute to the community a readily-usable and highly effective TCR representation model by open-sourcing a user-friendly deployment of my model SCEPTR.

Finally in chapter 5, I presented some early work on BANANAS: a Bayesian framework for predicting the probability of specificity of a TCR based on the distance between it and its nearest neighbour in a reference set of known binders to some pMHC of interest. While the work is still in its preliminary stages, it addresses an important need for robust uncertainty quantification in the computational modelling of TCR specificities, and to the best of my knowledge, is the first of its kind for distance-based TCR models. The generality of the theory underpinning BANANAS means that it can calibrate distances as measured by any arbitrary TCR metric, and gives it the potential to stay useful and relevant for the quantitative immunology community regardless of preferences towards certain TCR metrics over others, or future developments of improved TCR representation models and metrics. If a fuller and more polished version of this work can demonstrate its ability to reliably produce well-calibrated specificity predictions, then it could be combined with:

1. A semi-automated collation and standardisation of specificity-annotated TCR data using tidytcells

2. Distance measurements within SCEPTR's highly informative representation space

to create a fully free and open-source pipeline for TCR specificity prediction with robust uncertainty quantification.

6.2 Overall limitations

Here I take the opportunity to go over a few overarching limitations of my work that have not yet been addressed in the discussion sections of the preceding chapters.

Firstly, there are some fundamental limitations to the co-specificity-focused approach that I have taken throughout the work of my PhD candidacy. While there is strong evidence that TCR sequence similarity enriches for co-specificity, many if not most pMHCs have multiple binding solutions comprised of TCRs with significantly divergent primary sequences [22, 101]. As such, it is not uncommon for large fractions of binder TCRs against one pMHC to fail to cluster with any of the other co-specific TCRs based on measures of sequence similarity alone [72]. In many ways, it is this difficulty in resolving disparate binding solutions that is the biggest bottleneck to significant improvements in TCR specificity modelling. Although SCEPTR provides improved performance on

TCR specificity prediction compared to existing models, SCEPTR's reliance on unsupervised training objectives such as MLM and autocontrastive learning mean that it is very unlikely to have learned how to resolve co-specific TCRs with very different sequences. Indeed, the fact that SCEPTR generates comparable AUROC scores to the sequence alignment-based TCRdist is evidence that the co-specificity rules as learned by SCEPTR is still tightly aligned to some notion of TCR sequence similarity (Fig. 4.1d). Without, for example, significant increases in the volume of available specificity-annotated TCR data enabling large-scale supervised learning, or further innovations allowing us to use the currently limited amounts in a generalisable manner, the contrastive learning approach is likely to remain fundamentally limited in this way.

Secondly, more work can be done to better compare my own methods to other existing ones. For example in chapter 4, I compare SCEPTR's performance to 3 other PLMs and 2 sequence-alignment-based TCR metrics, but this is only a subset of the many existing TCR specificity models. Furthermore, while I did my best in the same chapter to compare SCEPTR and the aforementioned 5 models through a variety of inference modalities including nearest-neighbour prediction (Fig. 4.1d), linear SVCs (Fig. 4.4), and the average distance between a query TCR and all reference binders (Fig. A.3), there are other simple yet interesting approaches that have not yet been tried. Some candidates for future investigation are the use of non-linear kernel-based SVCs, and the fine-tuning of the general PLMs on TCR sequences. A similar criticism can be made towards the preliminary work on BANANAS in chapter 5. A more polished future version of this work should contain comparisons between the calibration quality of BANANAS to other methods of calibration such as naive k-nearest neighbours, or even other Bayesian methods such as Gaussian processes fitted on SCEPTR's representation space.

Thirdly, I have not presented any work on applying the developed models and methods on downstream applications. The end goal of computational TCR analysis is for the technology to better inform downstream biomedical applications of immunological knowledge. While my work does improve on the status quo of TCR representation modelling and specificity prediction, it is not yet clear to what extent these improvements translate to performance on higher-level tasks such as TCR repertoire classification [13–18], or the needle-in-a-haystack style identification of candidate TCRs as the payload of T cell therapies or reverse epitope discovery for vaccine design [82, 83] (e.g. due to its probably specificity to a pMHC of interest). To maximise the positive impact of my work, applications of SCEPTR and BANANAS on the above downstream ap-

plications are high-priority candidates for future investigation (see the following section).

6.3 Future work

While the limited 3-year timespan of my PhD candidacy has prevented me from fully addressing the above limitations, this leaves plenty of room for exciting avenues for further research. Below I share some ideas on future extensions to the work presented in this thesis.

6.3.1 Downstream applications of SCEPTR

Repertoire classification

From the diagnostic point of view, a major downstream application of computational TCR analysis is repertoire classification. That is, could TCR repertoire sequencing be used as a minimally invasive biomarker to diagnose diseases? SCEPTR's ability to produce highly informative embeddings of the TCR sequence may make it a useful component for a wider repertoire classification model.

Recall from my earlier review of existing works on TCR machine learning (see section 2.6) that repertoire classification is a multiple instance learning problem. In other words, repertoire classifiers should be able to make accurate predictions even in situations where only a small proportion of the TCRs in an input repertoire are predictive of the underlying disease state. With that in mind, one way to apply SCEPTR to downstream repertoire classification is shown in figure 6.1.

Here, the primary trainable component is a repertoire-level transformer. Its key and value vectors each represent a TCR, and are generated using SCEPTR's embeddings. It has one query vector ξ that is trainable, whose job is to adaptively pool information from across all input TCRs to generate a single repertoire-level representation vector. Finally, a single- (or multi-) layer perceptron maps the repertoire representation to a classification of disease state. Such a transformer-based architecture is a theoretically justified choice for the multiple instance learning setting, since the model can learn to adaptively attend to the most predictive TCRs within the input repertoire, even if they represent only a small proportion of the whole input set. This architecture also enables flexible scaling in complexity through 1) increasing the number of attention heads inside the repertoire transformer, or 2) allowing for multiple learned query vec-

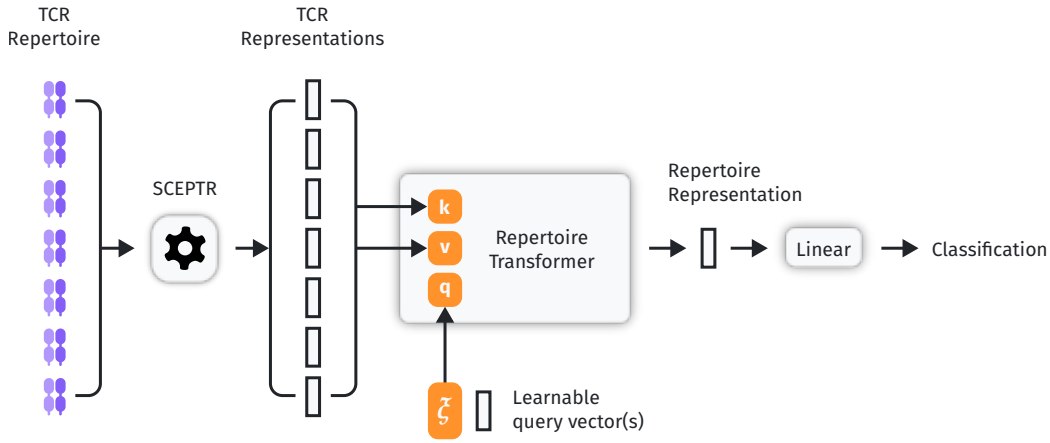


Figure 6.1: **Repertoire classification with SCEPTR.** TCR repertoire classification can be difficult because it is a multiple instance learning problem. One principled way [13] to use SCEPTR's embeddings for repertoire classification is to train a simple transformer that uses a fixed number of trainable query (q) vectors $[\xi_i]_{i=1}^{N}$, and uses transformations of SCEPTR's TCR embeddings as its key (k) and value (v) matrices (see section 2.4 for details on the transformer architecture). This type of architecture would be appropriate for a multiple-instance problem like repertoire classification as it will be able to flexibly attend to only the relevant and informative TCRs within a repertoire (some bystander TCRs may not be informative), and generate an overall repertoire representation selectively from that information. This repertoire representation can be passed through a simple perceptron to generate a repertoire classification.

tors $[\tilde{\zeta}_i]_{i=1}^{N}$. Both strategies to increase complexity may help the model more fully capture repertoire-level signals in cases where a set of multiple diverse disease-associated TCR metaclonotypes are thought to exist, whereas tactical reductions in complexity can improve model tractability when either 1) the available data is sparse and there is a high risk of model over-fitting, or 2) available compute is a limiting factor.

The architecture proposed above is similar to Widrich et al.'s DeepRC, which instead uses a CNN-based TCR representation model upstream of the repertoire transformer [13]. However, a key difference between my proposition above and existing repertoire classifiers including DeepRC is that SCEPTR is already pre-trained – thus, its weights can remain fixed. This leaves the repertoire transformer and final perceptron as the only trainable parts and helps protect the overall model from over-fitting to patient repertoire training data, which can often be noisy due to differences in donor age, MHC restriction, and sequencing depth and technology. If such a methodology were to work, it would be a useful step towards more robust models of TCR repertoire classification.

Meta-clonotyping/reverse epitope discovery

TCRs within a repertoire can be clustered using measures of sequence similarity in order to identify groups of receptors that are likely to respond to the same target. Mayer-Blackwell et al. refer to these groups of receptors as *meta-clonotypes* [82]. Pogorelyy et al. have shown in the context of SARS-CoV-2 that meta-clonotypes identified from real-world patient data can be used in conjunction with laboratory-generated specificity-annotated TCR data in order to conduct what they call *reverse epitope discovery* – that is, an initial identification of immunodominant TCR meta-clonotypes, followed by a targeted search for their cognate pMHC [83].

Meta-clonotype identification and reverse epitope discovery are important downstream applications of quantitative TCR analysis from the therapeutic point of view, as it can help identify candidate TCRs for engineered T cell therapies, and can also better inform designs for vaccines that aim to elicit cellular immune memory. It is also synergistic with the aforementioned application of TCR repertoire classification, as such identified TCRs may act as biomarkers of interest, whose presence may be predictive of certain disease states.

The simplest way to apply SCEPTR to this use-case is to use the distances within its representation space to generate TCR clusters. This should already improve upon available methods, given how our findings from chapter 4 indicate that distances within SCEPTR's representation

space do indeed correlate better with receptor co-specificity compared to existing models. There is also potential to combine SCEPTR distances with an improved BANANAS framework, which may enable intelligent and adaptive decisions on what distance thresholds to use in order to construct TCR cluster candidates based on posterior probabilities of TCR co-specificity.

6.3.2 Further improving SCEPTR

Intelligent tokenisation of TCR sequences

Current PLMs including SCEPTR treat input proteins as sequences of individual residues [44, 60, 61, 91, 92]. In other words, each residue by itself is considered a token – which would be the equivalent to a word or a word-piece in natural language. On one hand, this choice of metaphor between proteins and natural language has some nice properties. Firstly, it keeps protein tokenisation inexpensive, since all that needs to be done is to separate input strings by character. Secondly, it may also improve the interpretability of PLMs, since a model's understanding of a particular amino acid type can be unambiguously interrogated by examining the contents of, and information flow from, its unique corresponding token.

However, I would argue that it also comes with drawbacks, particularly when considering TCR language models. Apart from the central region of the CDR3 junction, the amino acid sequence of a TCR is fairly low-entropy [9, 101]. In such low-entropy regions, a residue-by-residue tokenisation of the TCR sequence forces earlier layers of a TCR language model to "memorise" predictable germline-encoded sequences to achieve high performance on MLM. This wastes model capacity on learning information that has very little if anything to do with the pMHC specificity of the TCR.

What are some possible ways to address this issue? Let us use SCEPTR as a case study. The only germline-encoded sequences within SCEPTR's input are the first and second CDR sequences of the α and β chains, both of which are contained in their respective chain's V gene sequence. Therefore, a possible solution could be to replace the sequences of the first two CDRs with a single token representing a V gene, while maintaining residue-by-residue tokenisations of the CDR3 sequences. While this does compress the germline-encoded V gene CDRs into a single token, it does not address the fact that the CDR3 sequence at the N- and C-termini still have relatively low entropy and can be trivially predicted given knowledge of the V- and J gene usages. Furthermore, this new design prevents

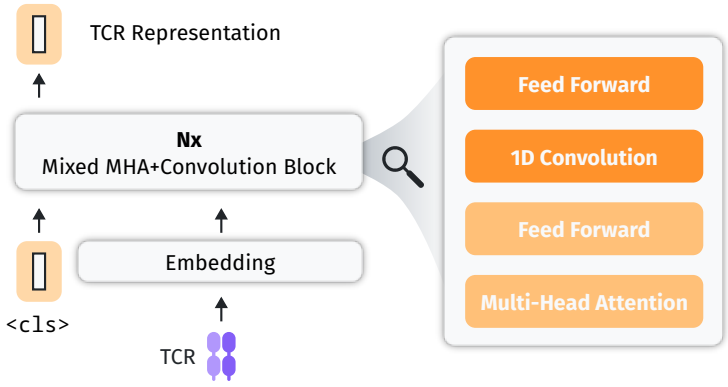


Figure 6.2: **Adding convolutional layers to a TCR language model.** A TCR language model like SCEPTR may benefit from being augmented with convolutional layers on top of its MHA-based transformer architecture. Compared to natural language, relationships between close-range neighbour residues on the CDR loops seem to contain significant information regarding a TCR's pMHC specificity. Therefore, the addition of one-dimensional convolutions between MHA layers inside the transformer encoder stack may help the model better pick up on such patterns.

the model from generalising any information learned from one V gene to another, since no V gene pairs share any common tokens. As a result, the model's understanding of the least frequently used V genes is likely to become unreliable.

Is there any way to interpolate between the extreme options of residue-by-residue tokenisation and compressing the entire V gene into one token? The problem of learning optimal tokenisation strategies from data is well-studied in NLP, where variational inference methods such as Google's SentencePiece have shown great success across different languages [28]. We know from multiple sequence alignments of germline-encoded TCR genes that there exist shared or similar substrings across sequences of different V, D and J gene segments. Therefore, methods similar to SentencePiece may also work for learning bespoke tokenisation strategies for TCR sequences. A successful implementation of this idea would effectively be trading an increase in the vocabulary size of the TCR language with easier access by the language model proper to higher-order sequence features.

Adding convolutional layers

The success of string k-mer kernel methods and CNNs in the field of TCR machine learning suggest that local patterns in the TCR sequence are highly informative of the receptor's target specificity [19, 42, 43, 72, 123]. This makes sense – since the CDRs are unfolded loops, the structural and physical properties of a particular residue within it would be most influenced by their closest neighbours. Therefore, adding dedicated model components like CNNs designed to detect such short-range patters into the language model stack of SCEPTR may help it better focus on such signals (Fig. 6.2).

Furthermore, transferring the load of learning such close-range dependencies to embedded CNN layers may free up the self-attention layers to focus more on longer-range dependencies where its strengths will shine brighter. In this sense, incorporating CNNs into SCEPTR may be a soft solution to the issue of low-entropy TCR sequences discussed in the previous section, if the CNN layers were to take over the responsibility of memorising germline-encoded TCR sequence patterns.

Intelligent distribution of token masking/censoring

My colleagues and I have demonstrated that information regarding epitope specificity is not uniformly distributed within the residues of the TCR sequence [101]. Namely, the β chain is more informative than the α chain, and the CDR3 sequences are more informative than the V genes (and thus the first two CDRs). In a separate preprint, my colleagues and I also provide evidence that information is more concentrated around the central region within the CDR3 loop [9]. However, token masking and censoring during SCEPTR's pre-training was done uniformly across the chains, CDR loops and token positions. Better aligning the distribution of masked/censored sequence features to that of information regarding pMHC specificity may help improve the efficiency of SCEPTR's pre-training. Furthermore, biasing the training towards such informative features may be yet another way of mitigating the issue of model capacity being wasted on learning germline-encoded sequence patterns that are in contrast uninformative.

Semi-supervised learning

In chapter 4, I briefly introduced Wang and Isola's asymptotic decomposition of the contrastive learning loss into the uniformity and alignment

terms. For clarity, I provide a simplified version of the decomposition below (see [103] and appendix B for a more in-depth exploration).

$$\mathcal{L}_{\text{contrastive}}(f) \coloneqq \mathop{\mathbb{E}}_{\substack{(x,x^+)\sim p_{\text{pos}} \\ \{y_i\}_{i=1}^N \overset{\text{iid}}{\sim} p_{\text{data}}}} \left[-\log \frac{e^{f(x)^\top f(x^+)}}{e^{f(x)^\top f(x^+)} + \sum_i e^{f(x)^\top f(y_i)}} \right]$$

$$\overset{\lim}{=}_{N\to\infty} \text{Uniformity}(f) + \text{Alignment}(f) + \log N$$

$$\text{Uniformity}(f) \coloneqq \log \mathop{\mathbb{E}}_{x,y \overset{\text{iid}}{\sim} p_{\text{data}}} \left[e^{f(x)^\top f(y)} \right]$$

$$\text{Alignment}(f) \coloneqq \mathop{\mathbb{E}}_{(x,x^+)\sim p_{\text{pos}}} \left[-f(x)^\top f(x^+) \right]$$

What this decomposition allows us to do is to separate the terms within the contrastive loss that are optimised with respect to the positive pair distribution p_{pos}, and the background data distribution p_{data}. In chapter 4, I used the decomposition as a theoretical tool to reason about why an autocontrastive objective helped improve the pre-training of SCEPTR. I noted that while autocontrastive learning approximates the alignment term using random views of the same underlying TCR, it still gives us an unbiased estimate of the uniformity term (assuming that the training data is IID with the background TCR distribution). I speculated that having a high-quality estimate of the uniformity term helped SCEPTR "unlearn" VDJ recombination statistics such that TCR distances in its representation space account for biases in generation probability, and lead to better co-specificity prediction.

But studying this decomposition begs the question: can we go further than using alignment and uniformity in a theoretical capacity? If we were to directly use the decomposed form of the loss to optimise SCEPTR, then the separation afforded by the decomposition would allow us to continue using the plentiful unlabelled background TCR data to approximate the uniformity term, while simultaneously using specificity-labelled TCR data to obtain a better approximation of the alignment term. This could lead to improvements in SCEPTR's representation quality, since it would then be learning from *real* examples of TCR pairs that are different in sequence but co-specific in function.

On one hand, I do believe that such strategies which can combine unsupervised and supervised learning (sometimes referred to as *semi-supervised* learning) will eventually yield the best results. Thus, I think

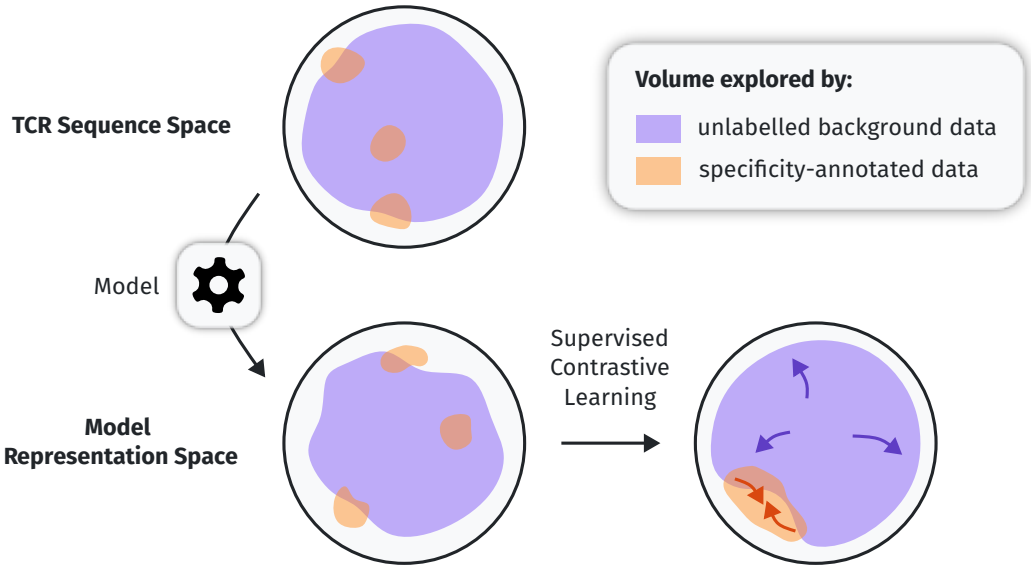


Figure 6.3: **The challenge of effective semi-supervised contrastive learning on expressive language models with limited TCR data.** The volume of TCR sequence space explored by specificity-annotated TCR data is relatively small compared to that explored by unlabelled background data. Because of this stark difference in the marginal TCR distributions of the two datasets, optimising alignment using labelled data and uniformity using unlabelled data can easily lead to a situation where the alignment penalty can be made small by pushing the TCR subspace explored by the labelled data into one corner of the representation space, while the uniformity penalty is kept low by spreading out the rest of the TCR space that is well-explored by the unlabelled data. This will not lead to a generalisable model, as none of the co-specificity rules learned from the labelled data necessarily restrict the model behaviour in the other parts of the representation space.

this general idea of optimising uniformity and alignment separately is worth studying. However, I suspect that current limitations in data availability pose significant challenges to realising the idea successfully. The main issue I see with a naïve implementation of the above idea is that the proportion of TCR space explored by currently available specificity-annotated data is small compared to that explored by the unlabelled background data. This poses a challenge, since independent minimisations of the alignment and uniformity terms over the two datasets is likely to lead to a sort of "false optimum" – one where the specific areas of TCR space explored by the labelled data are pushed to one corner of the representation space to minimise the alignment penalty, while the rest of the TCR space that is only explored by the background data is spread out widely to optimise for the uniformity penalty (Fig. 6.3). Indeed, this is what I have seen when conducting my own preliminary experiments trialling this approach.

This may sound contradictory to how I introduced the study of TCR coincidence statistics in section 2.3. After all, one of its motivating factors is its focus on local TCR co-specificity rules, which seemed to remain invariant regardless of where you looked in TCR space [22]. If that were truly the case, shouldn't learning the co-specificity rules in one area be enough to generalise to all other areas? Not necessarily – the key factor here is model flexibility. As far as PLMs go, SCEPTR is a relatively small model, but it still contains roughly 10^5 parameters. Because of its expressiveness, the local rules that SCEPTR learns in one corner of the TCR space need not significantly inform or restrict the local rules in other parts of the TCR space. This is how we can end up in a situation such as the one described in the previous paragraph.

With all of this in mind, I think the best way to implement semi-supervised contrastive learning is to first focus on very simple TCR metrics. Specifically, the model must be designed such that any co-specificity rules learned in one area of TCR space *do* highly restrict its predicted co-specificity rules in all other areas. This model property should mitigate the consequences of the aforementioned data limitations. One example of such a model might be a simple weighted edit distance over CDR3 sequences – like TCRdist – but where the substitution matrix is learnable.

To be clear, this this is not to say that there is no longer a place for more expressive representation models like SCEPTR. Firstly, they should eventually become the better choice – even with naïve supervised contrastive learning – once there is enough specificity-annotated TCR data available. Secondly, the flexibility of such models mean that there exist other ways of potentially improving their training (for example, see the

following two sections). Thirdly, the fact that such models map TCRs to a numerical representation space come with various practical benefits that make them easier to use as foundational components of downstream models, including the fact that distances are cheaper to compute on numerical vectors (and particularly fast if graphics processing units (GPUs) are available), and the existence of a vast literature studying parametric models over numerical vector spaces. I am simply speculating that from the very specific point of view of semi-supervised contrastive learning, a direct application to SCEPTR may not yield the best results.

That being said, if we find that simpler models of TCR co-specificity *can* be trained effectively through semi-supervised contrastive learning, then those simpler models could prove useful in guiding or improving the training of SCEPTR. For example, the rules as learned by the simpler model could be used to: 1) better inform token masking or censoring during MLM and autocontrastive learning (further discussed in the previous section), 2) generate data-informed synthetic positive TCR pairs, and/or 3) mine TCR pairs from unlabelled data that are highly probable to be co-specific in order to conduct weakly-supervised contrastive learning. If such a guided training regime can help SCEPTR recapitulate the co-specificity rules as learned by the simpler model, we can reap the rewards of semi-supervised contrastive learning while also retaining the benefits of a numerical representation model. Similar strategies of using simpler models to guide the training of larger models in settings with limited data availability have been used to great effect when fine-tuning large language models (LLMs) in NLP [32, 33], so it is not unreasonable to think that such an approach may also be effective in the TCR domain.

Using phenotypic labels as supervised signals in contrastive learning

In the previous section, I discussed the distributional properties of specificity-annotated TCR data that currently make it hard to use for supervision during contrastive learning. Namely, we want the marginal TCR distribution represented by specificity-annotated TCR data to be similar to the background TCR distribution, but this is currently not the case. From the data generation point of view, what is required for us to be able to generate a dataset that fits this criteria? Given current assay and sequencing technologies, specificity-annotated TCR data is generated by first fixing a set of pMHC targets, and subsequently testing a large number of TCRs against them for specificity. The implication of this is that we need to

know *a priori* a selection of pMHCs $\mathcal{P}$ such that:

$$\mathop{\mathbb{E}}_{\pi \in \mathcal{P}} \left[p_{S|II}(\tau|\pi) \right] \approx p_B(\tau) \tag{6.1}$$

where $p_{S|II}(\cdot|\pi)$ is the distribution of TCR binders to a pMHC π, and p_B is the background TCR distribution. However, this presents a chicken-and-egg problem since it is difficult to know prior to data generation what the binder distributions to each of these pMHCs look like. Furthermore, the vastness of the TCR space and its wide coverage by the background distribution means that the cardinality of $\mathcal{P}$ will likely need to be very large for equation 6.1 to be a good approximation. This makes the generation of a specificity-annotated TCR dataset suitable for contrastive learning a fundamentally hard problem.

In contrast, *phenotypic annotations* (e.g. CD4/CD8 positivity) can directly be obtained for TCRs sampled IID from background. This can be done, for example, by conducting single-cell ribonucleic acid (RNA) sequencing on T cells isolated from peripheral blood samples of healthy donors. Furthermore, such labels for TCRs are relatively cheaper to produce at a large scale. Finally, the associated T cell CD4/CD8 positivity of a TCR conveys information on what MHC structures that TCR can engage; thus, we can expect these labels to contain partial information regarding our primary dependent variable of interest, pMHC specificity. For these reasons, phenotypic labels present a promising interpolating step between unsupervised contrastive learning and fully supervised specificity-based contrastive learning on TCRs.

Probabilistic machine learning

In many ways, the end goal of quantitative TCR analysis is for the insights gained to be applied in the medical setting, where the cost of incorrect predictions is high. Therefore, robust uncertainty quantification is an important factor to consider.

On one hand, there are ways to implement uncertainty quantification by building additional machinery on top of SCEPTR. The BANANAS framework introduced in chapter 5 is one example of this. As alluded to earlier in section 6.2, there also exist other Bayesian methods such as Gaussian processes that can be fitted over SCEPTR's representation space to produce probabilistic predictions. In fact, Jokinen et al. have demonstrated that training a Gaussian process on simple BLOSUM62-based representations of the TCR can produce competitive specificity predictors [46] – thus, trying a similar methodology using SCEPTR's poten-

tially better-behaved representation space may be an interesting avenue for future work in its own right.

However, I argue that there are reasons to build uncertainty quantification directly into the representation model itself. This is because sometimes, we do not have full information about an input TCR. For example, consider that many TCR datasets are obtained through bulk RNA sequencing, and therefore only contain data on one (usually the β) chain. When embedding a lone β chain sequence, an ideal TCR representation model should be able to output a *distribution* over its latent space, to represent the scope of possible specificities the true underlying TCR could have depending on its unknown complementary chain.

One way to turn a TCR language model like SCEPTR into a probabilistic representation model is to train it as a *variational autoencoder (VAE)* [124]. The VAE is an application of deep neural networks to *variational inference*, where the objective is to train a model q to approximate the posterior distribution of some latent variable Z given some observable variable T. In our case, the observed variable would be the amino acid sequences of a TCR, and the latent variable would be a numerical representation of that receptor. The probabilistic framing of the variational inference problem necessitates that the model q outputs a distribution over possible representations.

More explicitly, the VAE paradigm is set up in the following way. Let Z be a random latent vector and T a random variable in TCR sequence space. Let us define a directed graphical model (Fig. 6.4) such that an observation on Z induces a distribution on T. We set a prior p_Z for Z, use one neural network with parameters θ to specify the likelihood $p_{T|Z;\theta}(\tau|z)$, and use a second network $q_{Z|T;\phi}(z|\tau)$ with parameters ϕ to approximate the posterior $p_{Z|T;\theta}(z|\tau)$ (solving for the exact posterior involves an intractable integral over the support of Z). Note how this setup resembles the general autoencoder approach, where the $q_{Z|T;\phi}$ network acts as an encoder/representation model and the $p_{T|Z;\theta}$ network acts as a decoder model. Like with any other variational inference problem, we make observations τ of T and maximise the evidence lower bound (ELBO):

$$\mathcal{L}_{\text{ELBO}}(\theta, \phi; \tau) = \log p_{T;\theta}(\tau) - KL(q_{Z|T;\phi}(\cdot|\tau) \| p_{Z|T}(\cdot|\tau)) \tag{6.2}$$

$$= \mathop{\mathbb{E}}_{z \sim q_{Z|T;\phi}(z|\tau)} \Big[\log p_{T|Z;\theta}(\tau|z) \Big] - KL(q_{Z|T;\phi}(\cdot|\tau) \| p_Z) \tag{6.3}$$

where the *evidence* $p_{T;\theta}(\tau) = \mathbb{E}_{z \sim p_Z}\, p_{T|Z;\theta}(\tau|z)$, and $KL(a\|b)$ is the Kullback-

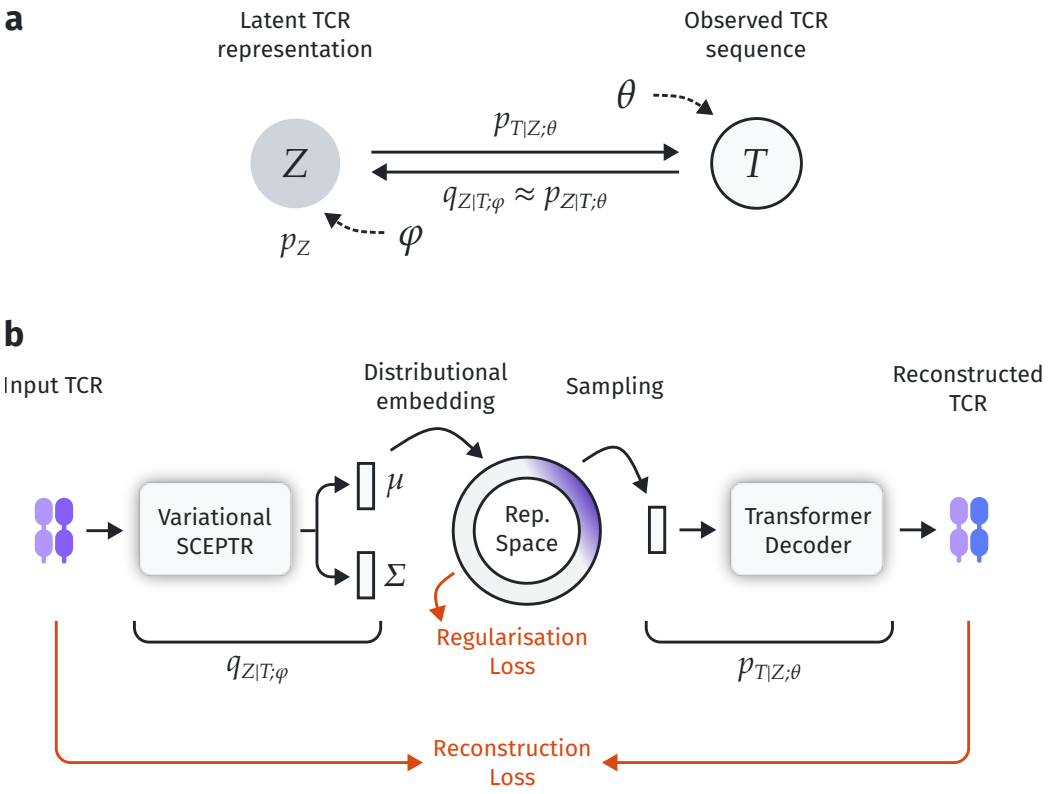


Figure 6.4: **Variational inference with a TCR representation model. a)** Let us define a directed graphical model where a latent TCR representation Z induces a distribution of possible TCR sequences followed by observed random variable T. Setting some simple prior distribution p_Z on the latent representations, one can optimise a ("decoder") neural network with parameters θ to model the generative distribution $p_{T|Z}$, while simultaneously optimising a second ("encoder") neural network q with parameters ϕ to approximate the posterior distribution $p_{Z|T}$. The q encoder network can then be used to get probabilistic embeddings of TCRs. **b)** A SCEPTR-like TCR language model can be used as the encoder network $q_{Z|T;\phi}$ that takes in a TCR sequence and outputs some sufficient statistics for the approximated posterior distribution of the latent variable Z. A realisation of Z can be sampled from the predicted distribution, and a generative model (e.g. transformer decoder, see section 2.4) can be used as the decoder network $p_{T|Z;\theta}$ to reconstruct the TCR sequence from the sampled value of Z. The ELBO loss (Eq. 6.3) is used to optimise both the encoder and decoder networks, where the regularisation term ensures that the predicted posteriors by q does not stray far from the set prior p_Z, and the reconstruction term ensures that the reconstructed TCR by p closely matches the original input TCR.

Leibler (KL) divergence between two probability distributions a and b:

$$KL(a\|b) \coloneqq \mathop{\mathbb{E}}_{x\sim a}\left[\log \frac{a(x)}{b(x)}\right] \tag{6.4}$$

The KL divergence is positive definite[1] and can be thought of as a measure of how dissimilar two probability distributions are.

Line 6.2 above motivates the use of the ELBO by showing that its maximisation is equivalent to jointly maximising the evidence (fitting the generative model to the data) and minimising the KL divergence between q and the true posterior distribution (making the probabilistic representation model q a better approximation to the true posterior). To make the maximisation tractable, we must rearrange terms to get line 6.3. The first term in line 6.3 can be thought of as a reconstruction error (if we used q to encode a TCR, could we generate it back with high probability?) and the second term acts as a regularisation factor which pulls q towards the prior p_Z.

Why would I now suggest a VAE approach, after having spent the entirety of chapter 4 advocating for the contrastive approach? The first and obvious reason is the motivation of this section, which is uncertainty quantification. Secondly, the two approaches are not mutually exclusive and can be combined either jointly or in sequence when training models. Thirdly, the VAE framework gives us a two-way mapping between the representation space and the TCR space – when only using the contrastive approach, mapping from the representation space back to TCR sequences is not possible. Finally, recall how I speculated in chapter 4 that a primary benefit of introducing an autocontrastive objective to SCEPTR's pre-training was its tendency to push the mapping of the background TCR distribution to a uniform distribution over the representation hypersphere. If this is indeed the case, then there is good reason to believe that the VAE approach will also produce good results. This is because the VAE approach allows us an explicit choice in assumption of the prior latent distribution p_Z – the second "regularisation" term in line 6.3 reflects this, as it does not let the posterior predicted by q to venture too far from the set prior. With a hyper-spherical representation space, we can for example *set* the prior as the uniform distribution, and use the von Mises-Fisher (vMF) distribution over the unit hypersphere as the family of posteriors generated by the q network. In fact, VAEs of this type have shown superior performance compared to traditional VAEs that use the multivariate Gaussian distribution as the latent prior and posterior [125].

[1] $KL(a\|b) \geq 0, \quad KL(a\|b) = 0 \iff a \equiv b$

6.4 Closing remarks

TCR-pMHC interaction is a central mechanism that enables T cell function, and is what endows T cells with target specificity. As such, uncovering the rules that govern which TCRs can engage which pMHCs is an important step in our journey as the collective scientific community to better understand our own adaptive immune system. However, the immense complexity of the system and the problem space combined with a limited volume of annotated TCR-pMHC pairs makes this a hard problem to solve. An effective machine learning-based solution to the problem requires three things: 1) the collation of large volumes of high-quality TCR data, 2) a way to train a model on the said data, and 3) a way to obtain useful predictions from the said model. To this end, I have presented 1) tidytcells, a free and open-source Python library for automating the preprocessing and standardisation of TCR and MHC data, 2) SCEPTR, a small transformer-based TCR representation model that shows state-of-the-art few-shot TCR specificity prediction performance, and provides us with some insight on how to best train PLMs on TCR data, and 3) BANANAS, a Bayesian framework for calibrating TCR metrics like SCEPTR for distance-based TCR specificity prediction. These three works all build on and are only made possible by the past hard work of the collective scientific community. My hope is that my work presented here may also one day contribute as a stepping stone and inspiration for further progress towards cracking the immune code.

Abbreviations

APC antigen-presenting cell.

AUC area under the curve.

AUROC area under the receiver operating characteristic.

BANANAS Bayesian nearest neighbour association.

BERT Bidirectional Encoder Representations from Transformers.

CDF cumulative distribution function.

CDR complementarity determining region.

CNN convolutional neural network.

ELBO evidence lower bound.

FN false negative.

FP false positive.

GPT generative pre-trained transformer.

GPU graphics processing unit.

IEDB Immune Epitope Database.

IID independent and identically distributed.

IMGT International immunogenetics information system.

KL Kullback-Leibler.

LLM large language model.

MHA multi-head attention.

MHC major histocompatibility complex.

MLE maximum likelihood estimation.

MLM masked-language modelling.

NLP natural language processing.

NPV negative predictive value.

PLM protein language model.

pMHC peptide-major histocompatibility complex.

PPV positive predictive value.

PR precision-recall.

PyPI Python package index.

ReLU rectified linear unit.

RNA ribonucleic acid.

ROC receiver operating characteristic.

SCEPTR Simple contrastive embedding of the primary sequence of T cell receptors.

SVC support vector classifier.

TCR T cell receptor.

TN true negative.

TP true positive.

VAE variational autoencoder.

vMF von Mises-Fisher.

Appendices

Appendix A

Supplementary analyses of SCEPTR

The following are supplementary material complementing the work discussed in chapter 4. All work presented in this chapter of the appendix has been distributed publicly as part of a preprint prior to the submission of this thesis under my authorship and copyright [29] (licensed CCBY).

A.1 Embedding TCRs with existing PLMs

A.1.1 TCR-BERT

The TCR-BERT model was downloaded through HuggingFace at `https://huggingface.co/wukevin/tcr-bert`. Since TCR-BERT is trained to read one CDR3 sequence at a time, we generated TCR representations by generating two independent representations of the α and β chain, and concatenating them together. The TCR-BERT representation of a chain was generated by feeding the model its CDR3 sequence, then taking the average pool of the amino acid token embeddings in the 8th self-attention layer, as recommended by the study authors [44].

A.1.2 ESM2

The ESM2 (T6 8M) model was downloaded through HuggingFace at `https://huggingface.co/facebook/esm2_t6_8M_UR50D`. ESM2 is trained on full protein sequences, but not protein multimers. Therefore, we generated ESM2 representations for the α and β chains separately, and concatenated them to produce the overall TCR representation. To generate

the representation of a TCR chain, we first used Stitchr [126] to reconstruct the full amino acid sequence of a TCR from its CDR3 sequence and V/J gene. Then, the resulting sequence of each full chain was fed to ESM2. We took the average-pooled result of the amino acid token embeddings of the final layer to generate the overall sequence representation, as recommended [91].

A.1.3 ProtBert

The ProtBert model was downloaded through HuggingFace at `https://huggingface.co/Rostlab/prot_bert`. Similarly to ESM2, ProtBert is trained on full protein sequences. Therefore, we again used Stitchr to generate full TCR chain amino acid sequences, and fed them to ProtBert to generate independent α and β chain representations. We again as recommended average-pooled the amino acid token embeddings of the final layer [92].

A.2 On different position embedding methods

To better understand TCR similarity rules as learned by PLMs, we measured the average distance penalty incurred within a model's representation space as a result of a single amino acid edit at various points along the length of the α/β CDR3 loops. To do this, we randomly sampled real TCRs from the testing partition of the Tanno et al. dataset [121] and synthetically introduced single residue edits in one of their CDR3 loops. Then, we measured the distance between the original TCR and the single edit variant according to a PLM. For each model, we sampled TCRs until we had observed at least 100 cases of: 1) each type of edit (insertions, deletions, substitutions) at each position, and 2) substitutions from each amino acid to every other. Since CDR3 sequences vary in length, we categorised the edit locations into one of five bins: `C-TERM` for edits within the first one-fifth of the CDR3 sequence counting from the C-terminus, then `M1`, `M2`, `M3`, and `N-TERM`, in that order. For this analysis, we investigated SCEPTR and TCR-BERT, since they are the two best performers out of the PLMs tested (Fig. 4.1d).

Both SCEPTR and TCR-BERT generally associate insertions and deletions (indels) with a higher distance penalty compared to substitutions (Fig. A.9a). While SCEPTR uniformly penalises indels across the length of the CDR3, TCR-BERT assigns higher penalties to those closer to the C-terminus. We hypothesised that the variation in TCR-BERT's indel

penalties is a side-effect of its position embedding system. TCR-BERT, like many other transformers, encodes a token's position into its initial embedding in a left-aligned manner using a stack of sinusoidal functions with varying periods [27, 30, 44] This results in embeddings that are more sensitive to indels near the C-terminus, which cause a frame-shift in a larger portion of the CDR3 loop and thus lead to a larger change in the model's underlying TCR representation. To test this hypothesis, we trained and evaluated a new SCEPTR variant:

SCEPTR (left-aligned) This variant uses a traditional transformer embedding system with trainable token representations and left aligned, stacked sinusoidal position embeddings.

While we detect no significant difference in downstream performance between SCEPTR and its left-aligned variant (Fig. A.10), this may be because cases where the differences in their learned rule sets affects performance are rarely seen in our benchmarking data. The edit penalty profile of the left-aligned variant shows a similar falloff of indel penalties than TCR-BERT with higher penalties at the C than N-terminals (Fig. A.9b). As their is no clear biological rationale for this observation, these results suggest that SCEPTR's relative position encoding might result in a better-calibrated co-specificity ruleset. These preliminary findings add to the ongoing discussion around how to best encode residue position information in the protein language modelling domain [93].

Interestingly, the penalty falloff seen with SCEPTR (left-aligned) is sharper than that of TCR-BERT, whose indel penalties plateau past `M1`. As TCR-BERT is a substantially deeper model (12 self-attention layers, 12 heads each, embedding dimensionality 768), it might be partially able to internally un-learn the left-aligned-ness of the position information. If this is true, then position embedding choices are particularly important for training smaller, more efficient models.

A.3 Supplementary Figures/Tables

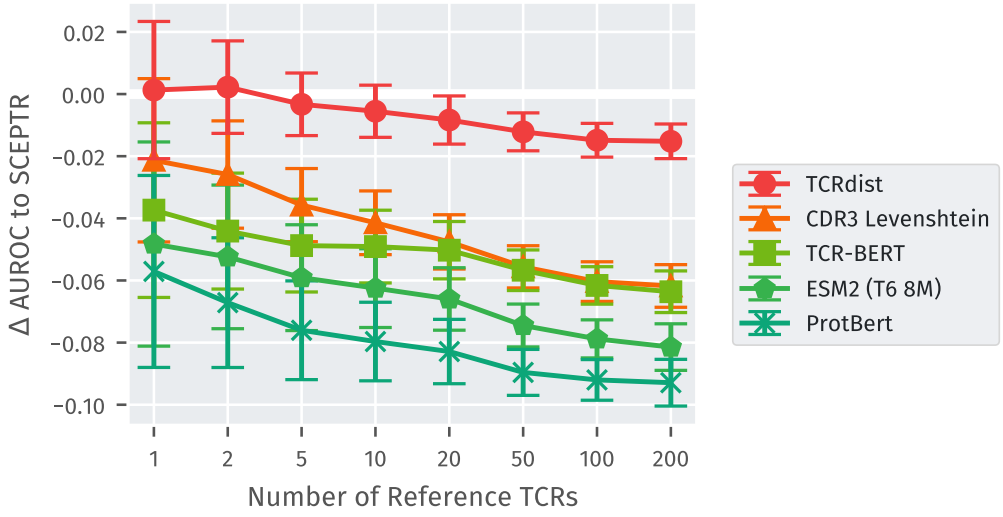


Figure A.1: **Statistical analysis of benchmark performance differences with respect to SCEPTR.** This is an accompanying plot to the benchmarking results shown in figure 4.1. The number of reference sequences varies along the x axis. The y axis shows the difference between the mean AUROC (ΔAUROC) of SCEPTR and other models. The error bars represent standard deviations, which were calculated across pMHCs and data splits. Reprinted from my own preprint [29] (licensed CCBY).

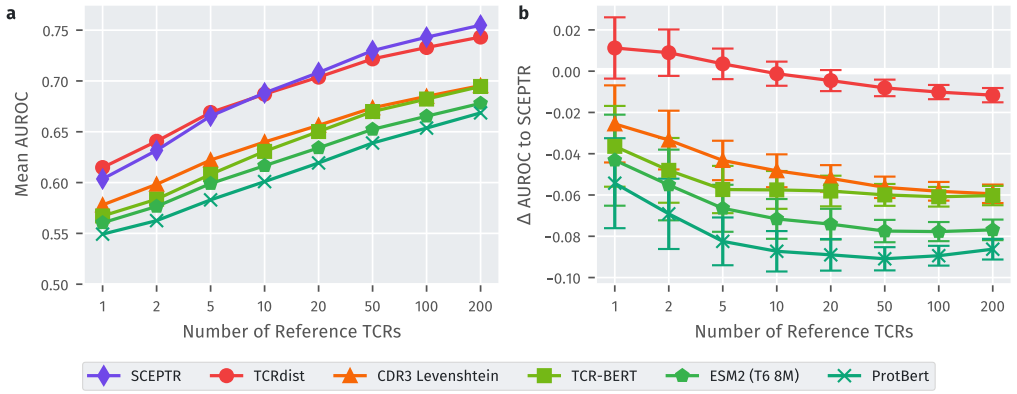


Figure A.2: **Benchmarking PLM embeddings on TCR specificity prediction with Montemurro et al.'s post-processed 10xGenomics dataset included.** This is a repeat of the benchmarking study from section 4.2.1 with a larger dataset of labelled TCRs. This includes six more sufficiently sampled pMHC specificities (with epitopes `ELAGIGILTV`, `GLCTLVAML`, `AVFDRKSDAK`, `IVTDFSVIK`, `RAKFKQLL`, `KLGGALQAK`). On both subplots, the number of reference sequences varies along the x axis. **a)** The y axis shows the models' AUROCs averaged across pMHCs. **b)** The y axis shows the distribution of ΔAUROC between SCEPTR and the other models (see Fig. A.1). The trends seen in section 4.2.1 are recapitulated. Reprinted from my own preprint [29] (licensed CCBY).

Table A.1: Per-epitope summary of the different models' performances from the nearest-neighbour prediction benchmarking (see section 4.2.1) with the number of reference TCRs k=200. The best AUROC per epitope is shown in bold.

epitope	SCEPTR	TCRdist	CDR3 Levenshtein	TCR-BERT	ESM2 (T6 8M)	ProtBert
`GILGFVFTL`	**0.911**	0.904	0.872	0.876	0.831	0.845
`NLVPMVATV`	**0.691**	0.648	0.632	0.655	0.652	0.629
`SPRWYFYYL`	**0.728**	0.695	0.610	0.637	0.604	0.575
`TFEYVSQPFLMDLE`	**0.976**	0.970	0.964	0.966	0.937	0.950
`TTDPSFLGRY`	0.708	**0.720**	0.600	0.576	0.579	0.564
`YLQPRTFLL`	**0.775**	0.762	0.743	0.698	0.697	0.669

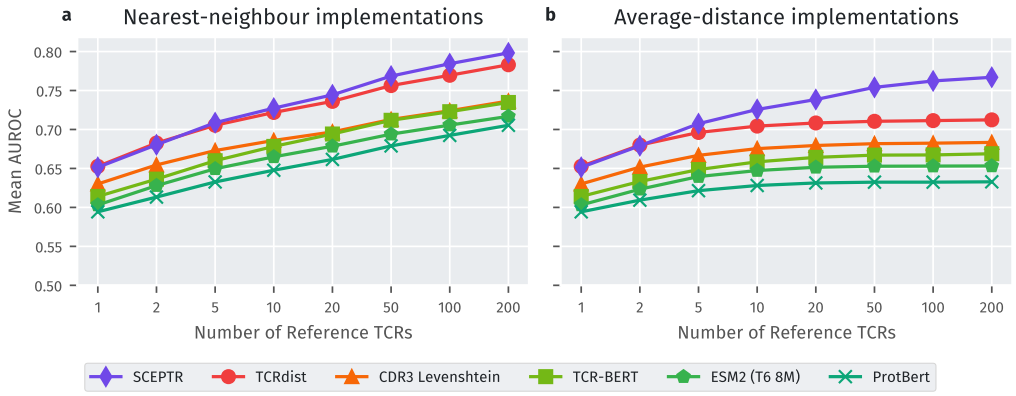


Figure A.3: **Nearest neighbour prediction is more performant than using the average distance to all references.** We repeated the benchmarking procedure used to produce figure 4.1d, but instead of making inferences based on the distance between a query TCR and its closest reference neighbour, we averaged its distance to all references. The results are shown in panel **b**, with panel **a** showing the original nearest neighbour prediction-based benchmarking results for comparison. The x axes show the number of reference sequences given to the model, while the y axes show average AUROC. SCEPTR shows state-of-the-art performance in both cases. All models perform better when applied through nearest neighbour prediction. Reprinted from my own preprint [29] (licensed CCBY).

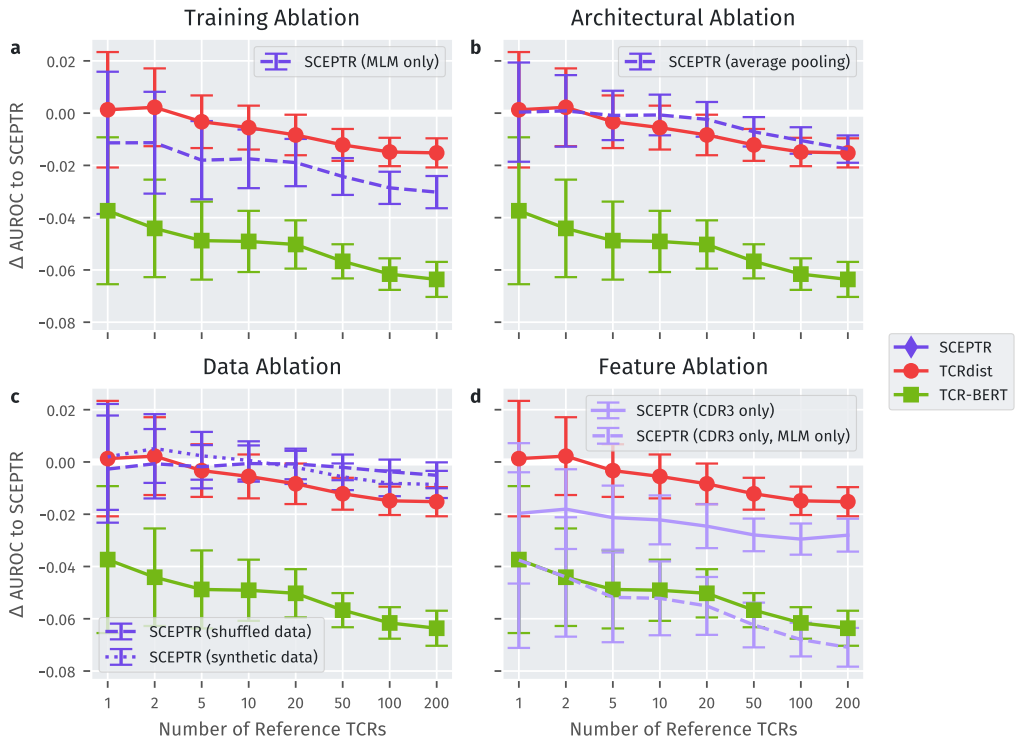


Figure A.4: **Statistical analysis of benchmark performance differences from the ablation studies.** This is an accompanying figure to the ablation study results shown in figure 4.3. The number of reference sequences varies along the x axis. The y axis shows the distributions of ΔAUROC to SCEPTR (see Fig. A.1). Reprinted from my own preprint [29] (licensed CCBY).

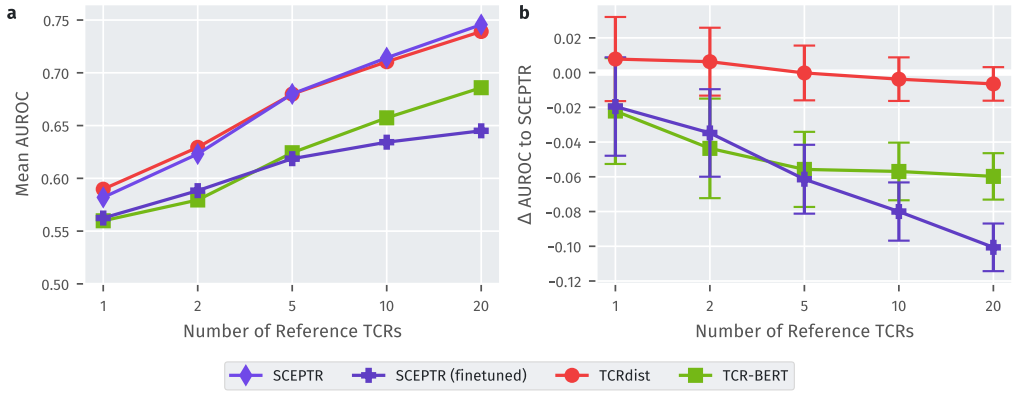


Figure A.5: **Benchmarking fine-tuned SCEPTR on TCR specificity prediction for unseen pMHCs.** This figure shows the result of benchmarking fine-tuned SCEPTR (see section 4.2.5) against TCRdist, TCR-BERT and the baseline SCEPTR model on pMHC targets unseen during SCEPTR's fine-tuning. Here, we use the benchmarking framework from section 4.2.1, with the prediction targets being all pMHCs other than the six seen during fine-tuning that have at least 120 known binders. The number of reference sequences k varies along the x axis. Here, we benchmark $k \in [1, 20]$ as we want to always have at least 100 positive test cases for each data split. **a)** The y axis shows the models' AUROCs averaged across pMHCs. **b)** The y axis shows the distribution of ΔAUROC (see Fig. A.1) between the baseline SCEPTR model and the others. Fine-tuned SCEPTR performs significantly worse compared to the baseline model. Reprinted from my own preprint [29] (licensed CCBY).

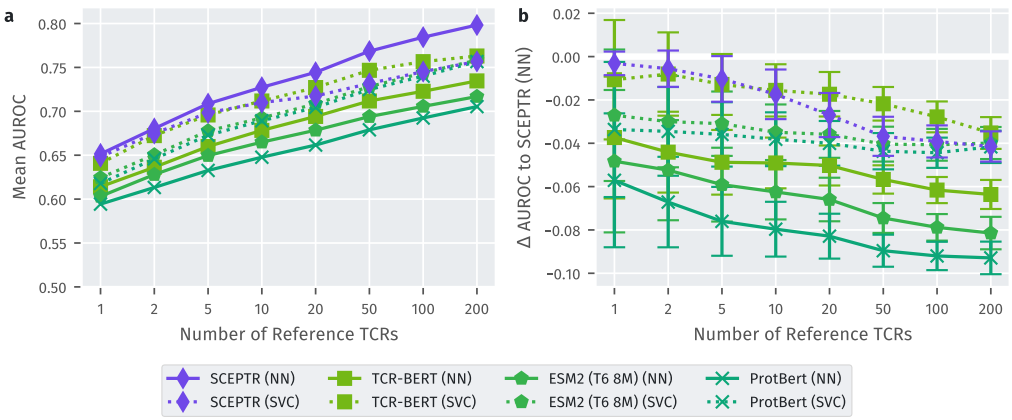


Figure A.6: **Benchmarking linear support vector classifiers trained on PLM features on TCR specificity prediction.** These plots compare the performances of different PLMs applied to few-shot TCR specificity prediction, either through nearest neighbour prediction (models marked as "NN" in the legend, see section 4.2.1) or using a linear support vector classifier trained atop their TCR featurisations (models marked as "SVC" in the legend, see section 4.2.4, methods 4.4.4). Benchmarking was done using the framework outlined in section 4.2.1. In both plots, the number of reference sequences varies along the x axis. **a)** The y axis shows the models' AUROCs averaged across pMHCs. **b)** The y axis shows the distribution of ΔAUROCs (see Fig. A.4) between SCEPTR (NN) and the other models. For all PLMs except SCEPTR, training a linear SVC atop the model's features improves performance. SCEPTR (NN) outperforms all methods, even the SVC trained atop its own features. Reprinted from my own preprint [29] (licensed CCBY).

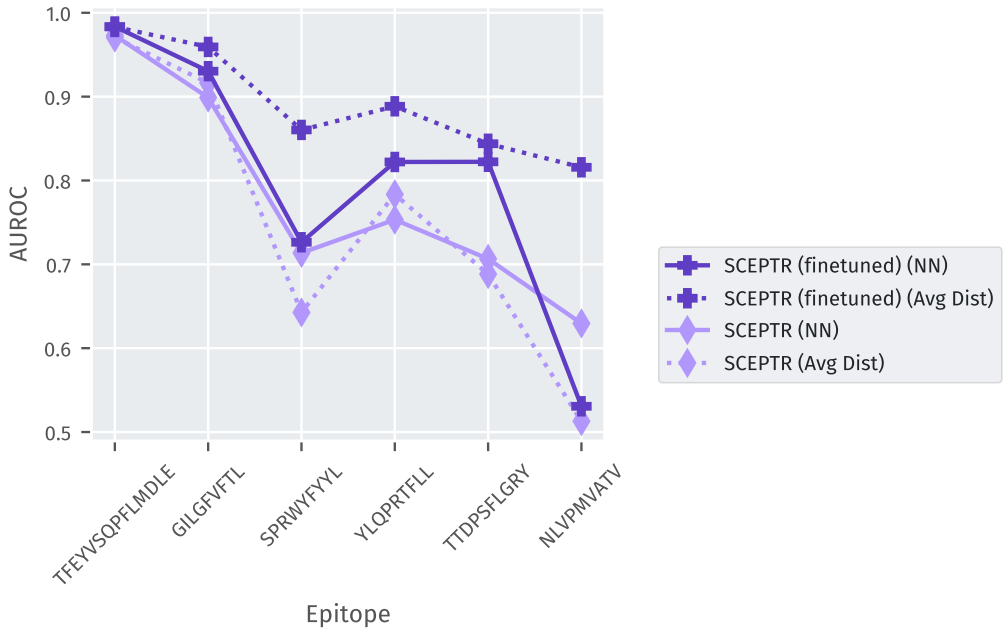


Figure A.7: **Comparing nearest neighbour and average distance implementations of the baseline and fine-tuned SCEPTR models.** This is an accompanying plot to figure 4.5. Here, we investigate the TCR specificity prediction performance of the baseline and fine-tuned versions of SCEPTR, implemented via the nearest neighbour (NN) or average distance (Avg dist) prediction frameworks (see section 4.2.1). For the baseline model, we see that the nearest neighbour implementation performs better on average, consistent with the results seen in figure A.3. In contrast, the average distance implementation of the fine-tuned model greatly outperforms its nearest neighbour counterpart. Our primary hypothesis as to why most models including the baseline SCEPTR model perform better through nearest neighbour prediction (Fig. A.3) is that each pMHC has multiple viable binding solutions comprised of TCRs with different primary sequence features, which means that averaging distance to all reference TCRs across binding solutions dilutes signal. The fact that the fine-tuned model no longer shows this property may hint at its ability to better resolve these distinct binding solutions into a single convex cluster. Reprinted from my own preprint [29] (licensed CCBY).

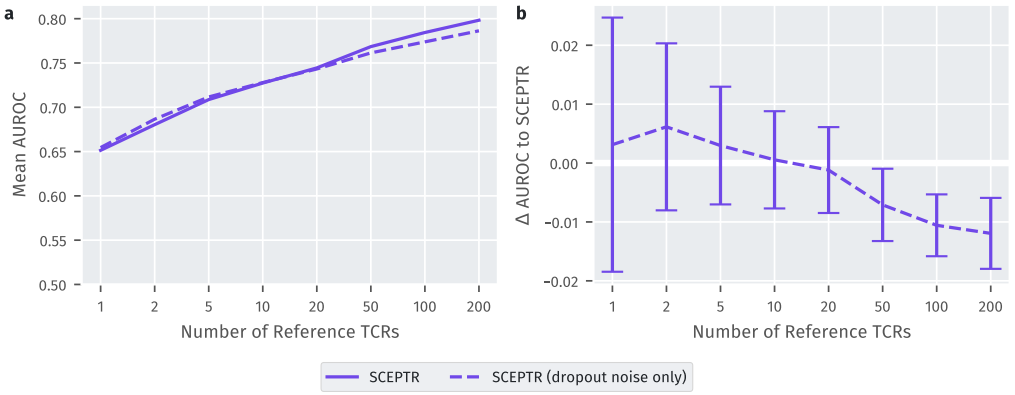


Figure A.8: **Residue and chain dropping provide an improved noise model during autocontrastive learning.** This plot summarises benchmarking results comparing SCEPTR with a variant (**SCEPTR (dropout noise only)**) which does not employ any extra noising operations when producing the two views of the same TCR during autocontrastive learning, solely relying on SCEPTR's internal dropout noise (see section 4.4.3). The benchmarking framework as outlined in section 4.2.1 is used. The number of reference sequences varies along the x axis. **a)** The y axis shows the models' AUROCs averaged across pMHCs. **b)** The y axis shows the distribution of ΔAUROCs (see Fig. A.1) from the dropout noise only variant to the baseline model. While overall performance is similar, the addition of out residue- and chain- dropping noise model improves downstream performance. Reprinted from my own preprint [29] (licensed CCBY).

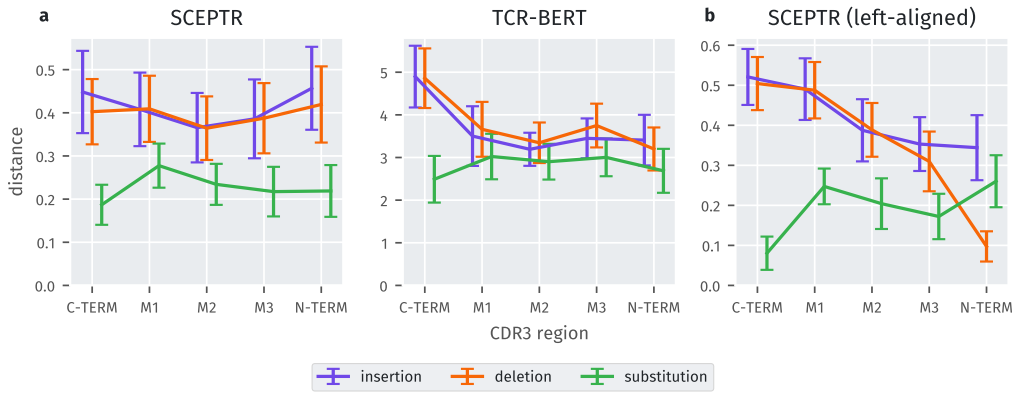


Figure A.9: **Investigating TCR co-specificity rules as learned by different PLMs.** Here we investigate TCR co-specificity rules as learned by various representation models by measuring the expected distance penalties incurred by single residue edits in different regions of the α and β CDR3 loops. We investigate three models: SCEPTR, TCR-BERT, and a SCEPTR variant which replaces its simplified initial embedding module with one that emulates the traditional transformer architecture, including a left-aligned position embedding system (see appendix A.2). The x axis shows different regions of the CDR3 divided into five bins, where `C-TERM` represents the first fifth of the loop counting from the C-terminal, `N-TERM` represents the last fifth of the loop on the N-terminal end, and the middle regions numbered from the C-terminal as shown. The y axis hows the expected distance penalty incurred by different types of single edits. The different lines show the expected penalty curves with respect to insertions (purple), deletions (orange) and substitutions (green). The error bars show the standard deviations. According to all models, substitutions on average incur a smaller distance penalty compared to indels. While SCEPTR uniformly penalises indels, both TCR-BERT and the left-aligned SCEPTR variant assign higher distance penalties to indels closer to the C-terminal. Reprinted from my own preprint [29] (licensed CCBY).

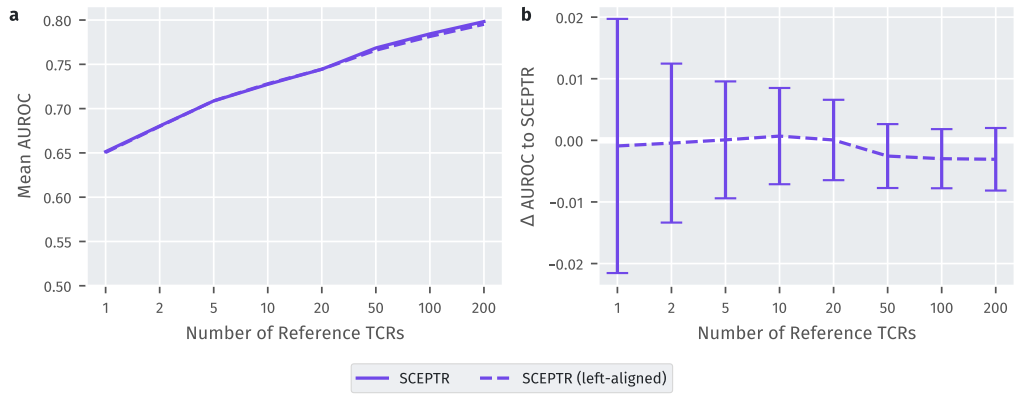


Figure A.10: **SCEPTR with its simplified embedder module performs similarly to a variant with an embedder module emulating the traditional transformer architecture.** Here we show the results of benchmarking SCEPTR against a variant which replaces SCEPTR's simplified embedder module (see methods 4.4.2) with an implementation emulating the traditional transformer architecture ("left-aligned" variant in plot, see appendix A.2). The benchmarking framework as outlined in section 4.2.1 is used. The number of reference sequences varies along the x axis. **a)** The y axis shows the models' AUROCs averaged across pMHCs. **b)** The y axis shows the distribution of ΔAUROCs (see Fig. A.1) from the left-aligned variant to the baseline model. We detect no significant difference in performance between the two models. Reprinted from my own preprint [29] (licensed CCBY).

Table A.2: The different studies that contributed TCR data to the training/validation and test splits for the supervised contrastive learning fine-tuning task. For datasets without a PubMed ID the table indicates the VDJdb GitHub issue number corresponding to the dataset inclusion.

epitope	training/validation	test
GILGFVFTL	PMID:28636592	PMID:12796775, PMID:18275829, PMID:28250417, PMID:28931605, PMID:7807026, PMID:28423320, PMID:28636589, PMID:27645996, PMID:29483513, PMID:29997621, PMID:34793243, VDJdbID:215
NLVPMVATV	PMID:28636592, VDJdbID:332	PMID:19542454, PMID:26429912, PMID:19864595, PMID:28423320, PMID:16237109, PMID:28636589, PMID:36711524, PMID:28623251, PMID:9971792, PMID:17709536, PMID:28934479, PMID:34793243, VDJdbID:252
SPRWYFYYL	PMID:33951417, PMID:35750048	PMID:33945786, PMID:34793243
TFEYVSQPFLMDLE	PMID:35750048	PMID:37030296
TTDPSFLGRY	PMID:35383307	PMID:35750048
YLQPRTFLL	PMID:35383307, PMID:34793243	PMID:34685626, PMID:37030296, PMID:33664060, PMID:33951417, PMID:35750048, VDJdbID:215

Appendix B

Deriving contrastive loss from a model of TCR-pMHC binding

The contrastive loss has recently gained popularity in the context of representation learning [103], and it takes the form:

$$\mathcal{L}_{\text{contrastive}}(f) \coloneqq \mathop{\mathbb{E}}_{\substack{(x,x^+)\sim p_{\text{pos}} \\ \{y_i\}_{i=1}^N \overset{\text{iid}}{\sim} p_{\text{data}}}} \left[-\log \frac{e^{f(x)^\top f(x^+)}}{e^{f(x)^\top f(x^+)} + \sum_i e^{f(x)^\top f(y_i)}} \right] \tag{B.1}$$

Here, $f : \mathcal{X} \to \mathcal{S}^{m-1}$ is some representation model mapping elements from observation space $\mathcal{X}$ to the m-dimensional unit hypersphere $\mathcal{S}^{m-1} \in \mathbb{R}^m$, p_{pos} is the joint distribution of positive data pairs, p_{data} is the background distribution of all data, and $N \in \mathbb{Z}_+$ is some fixed number of negative samples.

Wang and Isola have shown that at the limit of infinite data, optimising the contrastive loss is equivalent to jointly optimising "alignment" (Eq. B.2) and "uniformity" (Eq. B.3). Alignment describes how close together positive pairs are in the representation space, while uniformity is a measure of how fully a representation model uses this space. These two metrics together can be used to evaluate the quality of a representation mapping onto the unit hypersphere [103].

$$\text{Alignment}(f) \coloneqq \mathop{\mathbb{E}}_{(x,x^+)\sim p_{\text{pos}}} \left[\|f(x) - f(x^+)\|_2^\alpha \right], \quad \alpha > 0 \tag{B.2}$$

$$\text{Uniformity}(f) \coloneqq \log \mathop{\mathbb{E}}_{x,y \overset{\text{iid}}{\sim} p_{\text{data}}} \left[e^{-t\|f(x)-f(y)\|_2^2} \right], \quad t > 0 \tag{B.3}$$

Here I provide some insight on why the application of this form of contrastive learning to the TCR co-specificity problem is principally justified, as well as a more TCR-oriented interpretation of what model behaviour the contrastive loss function is really enforcing. I do this by first proposing a simple model of TCR co-specificity backed up by empirical evidence, then using maximum likelihood estimation (MLE) to end up with a loss formulation equivalent to Wang and Isola's alignment and uniformity, which in turn is asymptotically equivalent to the contrastive loss.

B.1 The model

Let $p_B(\tau)$ be the background probability that a TCR τ exists in an individual's TCR repertoire at any given time. Then, one way to model the probability p_{cosel} that two distinct TCRs τ_x and τ_y will both be selected against the same epitope stimulus within the same repertoire is:

$$p_{\text{cosel}}(\tau_x, \tau_y) = q(\tau_x, \tau_y) p_B(\tau_x) p_B(\tau_y) \tag{B.4}$$

Here, $p_B(\tau_x)p_B(\tau_y)$ is the joint background probability of the two TCRs, and $q(\tau_x, \tau_y)$ can be thought of as a *co-selection factor* which quantifies the functional similarity between the two TCRs.

It has previously been shown that the probability of observing co-specific TCR pairs with a sequence similarity up to some Levenshtein edit distance falls off exponentially with respect to the distance [22]. This motivates us to define q as the negative exponential of a function d with trainable parameters θ (Eq. B.5). Here we can imagine d as some 'functional distance' measure between two TCRs, which the model formulation asserts also has the same exponential fall-off property. I furthermore introduce s, a selection factor that captures the general ability of a TCR to be selected for any epitope stimulus such that $s(\tau)p_B(\tau) = p_S(\tau)$ is the general post-selection distribution of TCRs. The resulting expression for p_{cosel} in terms of θ is shown in equation B.6. Note that Z_θ here is a normalisation factor to ensure that p_{cosel} is a well-defined probability distribution over its domain of all possible pairs of TCRs (i.e. $\sum_{\tau_x,\tau_y} p_{\text{cosel}} = 1$).

$$q(\tau_x, \tau_y; \theta) := \frac{1}{Z_\theta} e^{-d(\tau_x,\tau_y;\theta)} s(\tau_x) s(\tau_y) \tag{B.5}$$

$$\implies p_{\text{cosel}}(\tau_x, \tau_y; \theta) = \frac{1}{Z_\theta} e^{-d(\tau_x,\tau_y;\theta)} p_S(\tau_x) p_S(\tau_y) \tag{B.6}$$

B.2 Maximum likelihood estimation

Assume we have a training set $\mathcal{T} := \{\boldsymbol{\tau}_i\}_{i=1}^{G}$ of G distinct sets $\boldsymbol{\tau}_i = \{\tau_{ij}\}_{j=1}^{N_i}$ of N_i epitope-restricted TCRs. Assume furthermore that the co-selection probability p_{cosel} of a set $\boldsymbol{\tau}_i$ of $N_i > 2$ TCRs can be approximated as the product of the pairwise co-selection probabilities $p_{\text{cosel}}(\tau_{ix}, \tau_{iy})$ over all pairs of TCRs in $\boldsymbol{\tau}_i$ (Eq. B.7). Substituting in equation B.6, the pseudo-likelihood approximation over $\mathcal{T}$ then becomes:

$$
\begin{aligned}
p_{\text{cosel}}(\mathcal{T};\theta) &= \prod_{i=1}^{G} p_{\text{cosel}}(\boldsymbol{\tau}_i;\theta) \\
&\approx \prod_{i=1}^{G} \left[\prod_{\tau_{ix},\tau_{iy}\in\boldsymbol{\tau}_i} p_{\text{cosel}}(\tau_{ix},\tau_{iy};\theta) \right] && \text{(B.7)} \\
&= \prod_{i=1}^{G} \left\{ \prod_{\tau_{ix},\tau_{iy}\in\boldsymbol{\tau}_i} \left[\frac{1}{Z_\theta} e^{-d(\tau_{ix},\tau_{iy};\theta)} p_S(\tau_{ix}) p_S(\tau_{iy}) \right] \right\} && \text{(B.8)}
\end{aligned}
$$

Evaluating the negative log likelihood, we get:

$$
\begin{aligned}
-\log p_{\text{cosel}}(\mathcal{T};\theta) &\approx -\sum_{i=1}^{G} \left\{ \sum_{\tau_{ix},\tau_{iy}\in\boldsymbol{\tau}_i} \log \left[\frac{1}{Z_\theta} e^{-d(\tau_{ix},\tau_{iy};\theta)} p_S(\tau_{ix}) p_S(\tau_{iy}) \right] \right\} \\
&= \left[\sum_{i=1}^{G} \sum_{\tau_{ix},\tau_{iy}\in\boldsymbol{\tau}_i} d(\tau_{ix},\tau_{iy};\theta) \right] + N_p \log Z_\theta + \text{const.} && \text{(B.9)}
\end{aligned}
$$

where $N_p = \sum_{i=1}^{G} {N_i}^2$ is the total number of epitope-matched TCR pairs in $\mathcal{T}$, and terms unrelated to θ are summarised as "const.". Dividing everything by N_p, ignoring the constant term with respect to θ, and taking the limit to infinite training data, we get an expression for a TCR co-selection loss function $\mathcal{L}_{\text{cosel}}$:

$$
\begin{aligned}
\mathcal{L}_{\text{cosel}} &:= \frac{1}{N_p} \left[\sum_{i=1}^{G} \sum_{\tau_{ix},\tau_{iy}\in\boldsymbol{\tau}_i} d(\tau_{ix},\tau_{iy};\theta) \right] + \log Z_\theta \\
&\overset{\lim_{G,N_i\to\infty}}{=} \mathop{\mathbb{E}}_{(\tau_x,\tau_x^+)\sim p_{\text{cosel}}} \left[d(\tau_x,\tau_x^+;\theta) \right] + \log Z_\theta && \text{(B.10)}
\end{aligned}
$$

By recalling that Z_θ is a normalisation factor defined in the scope of equation B.6, we can express its value in terms of d:

$$\begin{aligned}\sum_{\tau_x,\tau_y} p_{\text{cosel}}(\tau_x,\tau_y) = 1 &= \frac{1}{Z_\theta}\sum_{\tau_x,\tau_y} e^{-d(\tau_x,\tau_y;\theta)} p_S(\tau_x)p_S(\tau_y) \\ &= \frac{1}{Z_\theta}\mathop{\mathbb{E}}_{\tau_x,\tau_y\overset{\text{iid}}{\sim}p_S}\left[e^{-d(\tau_x,\tau_y;\theta)}\right] \\ \implies Z_\theta &= \mathop{\mathbb{E}}_{\tau_x,\tau_y\overset{\text{iid}}{\sim}p_S}\left[e^{-d(\tau_x,\tau_y;\theta)}\right] \end{aligned} \tag{B.11}$$

This allows us to express $\mathcal{L}_{\text{cosel}}$ completely in terms of d:

$$\mathcal{L}_{\text{cosel}} = \mathop{\mathbb{E}}_{(\tau_x,\tau_x^+)\sim p_{\text{cosel}}}\left[d(\tau_x,\tau_x^+;\theta)\right] + \log \mathop{\mathbb{E}}_{\tau_y,\tau_z\overset{\text{iid}}{\sim}p_S}\left[e^{-d(\tau_y,\tau_z;\theta)}\right] \tag{B.12}$$

B.3 Relationship to alignment and uniformity

The first and second terms in our final expression for $\mathcal{L}_{\text{cosel}}$ (Eq. B.12) look very similar to Wang and Isola's alignment (Eq. B.2) and uniformity (Eq. B.3) respectively. In fact, they are both direct analogues. Below are some observations from my derivations of TCR alignment (Eq. B.13) and uniformity (Eq. B.14):

$$\text{Alignment}_{\text{TCR}}(d) := \mathop{\mathbb{E}}_{(\tau_x,\tau_x^+)\sim p_{\text{cosel}}}\left[d(\tau_x,\tau_x^+;\theta)\right] \tag{B.13}$$

$$\text{Uniformity}_{\text{TCR}}(d) := \log \mathop{\mathbb{E}}_{\tau_x,\tau_y\overset{\text{iid}}{\sim}p_S}\left[e^{-d(\tau_x,\tau_y;\theta)}\right] \tag{B.14}$$

The positive distribution The distribution p_{cosel} is the TCR analogue to the positive-pair distribution p_{pos} in the general ML setting, specifically in this context of TCR epitope restriction.

The data distribution The distribution p_S is the TCR analogue to p_{data} in the general ML setting, as both represent a form of background distribution of the data. However, note how p_S is not necessarily p_B, reflecting how some TCRs may generally have a lesser ability to respond to pMHCs in general. On one hand, the naïve but most direct way to estimate p_S

may be to sample from the union of epitope-specific TCR sets across all (or as many available) pMHCs:

$$p_S(\tau) = \mathop{\mathbb{E}}_{\pi}\left[p_{S|\Pi}(\tau|\pi)\right] \tag{B.15}$$

where $p_{S|\Pi}(\cdot|\pi)$ is the binder distribution to a particular pMHC π. However, considering that the number of pMHCs for which we have specificity-annotated TCR data is extremely limited, the best empirical estimation of p_S may still be to sample from the peripheral blood of healthy individuals.

Alignment and uniformity hyper-parameters Any transformations to the distance (α from equation B.2, t from equation B.3) should appear equivalently on both the alignment and uniformity metrics to keep them on an equivalent scale. If we want a negative exponential relationship between co-selection and distance, then this corresponds to $\alpha = 1$ and $t = 1$. Opting for a Gaussian kernel (as Wang and Isola do in their formulation of uniformity) means that the distance should be squared in both the alignment and uniformity terms.

Alignment and uniformity weighting The MLE framework suggests that alignment and uniformity should be optimised with equal weighting (i.e. one unit of alignment is equivalent to one unit of uniformity).

Generality to arbitrary metrics My derivation suggests that we can generalise the idea of alignment and uniformity from euclidean distances between unit-hypersphere embeddings to an arbitrary notion of TCR distance.

B.4 Relationship to contrastive loss

Since cosine similarity[1] is monotonically negative to distance, let us define d with respect to some representation model f such that:

$$d(x, y) = -f(x)^\top f(y) \tag{B.16}$$

Then, we can rearrange the terms in the expression for $\mathcal{L}_{\text{contrastive}}$ (Eq. B.1) and show that at the limit of infinite negative samples, it is equivalent to

[1]Recall that the model in question f maps to the unit hypersphere, so a dot product between model outputs is equivalent to the cosine similarity between them.

our co-selection loss. This is similar to what Wang and Isola show in terms of their alignment and uniformity formulations, but in this case the limit is exact.

$$\begin{aligned}
\mathcal{L}_{\text{contrastive}} &= \mathop{\mathbb{E}}_{\substack{(x,x^+)\sim p_{\text{pos}} \\ \{y_i\}_{i=1}^N \overset{\text{iid}}{\sim} p_{\text{data}}}} \left[-\log \frac{e^{-d(x,x^+)}}{e^{-d(x,x^+)} + \sum_i e^{-d(x,y_i)}} \right] \\
&= \mathop{\mathbb{E}}_{\substack{(x,x^+)\sim p_{\text{pos}} \\ \{y_i\}_{i=1}^N \overset{\text{iid}}{\sim} p_{\text{data}}}} \left\{ d(x,x^+) + \log \left[e^{-d(x,x^+)} + \sum_i e^{-d(x,y_i)} \right] \right\} \\
&\overset{\lim}{=}_{N\to\infty} \mathop{\mathbb{E}}_{(x,x^+)\sim p_{\text{pos}}} \left\{ d(x,x^+) + \log \left[N \times \mathop{\mathbb{E}}_{y\sim p_{\text{data}}} \left\{ e^{-d(x,y)} \right\} \right] \right\} \\
&= \mathop{\mathbb{E}}_{(x,x^+)\sim p_{\text{pos}}} \left[d(x,x^+) \right] + \log \mathop{\mathbb{E}}_{x,y \overset{\text{iid}}{\sim} p_{\text{data}}} \left[e^{-d(x,y)} \right] + \log N \qquad \text{(B.17)}
\end{aligned}$$

In other words, we can interpret contrastive loss as a within-batch empirical estimation of the co-selection loss with a $\log N$ term added on.

Appendix C

BANANAS: Approximating background distance likelihood

In equation 5.7 of chapter 5, I make the following approximation[1]:

$$\begin{aligned} P(\Delta = \delta | Z = 0) &= \sum_{\tau \in \mathcal{T}} \sum_{\tau' \in \mathcal{T}} p_B(\tau) p_{S|\Pi}(\tau' | \pi) I_{d(\tau,\tau')=\delta} \\ &\approx p_C[p_B](\delta) \end{aligned} \tag{C.1}$$

Let us see how to arrive at this approximation, and what assumptions are necessary in order for it to be valid.

Without loss of generality, let us re-express the distribution $p_{S|\Pi}(\cdot|\pi)$ as follows:

$$p_{S|\Pi}(\tau|\pi) = q_\pi(\tau) p_B(\tau), \quad \text{subject to } \mathop{\mathbb{E}}_{\tau \sim p_B}[q_\pi(\tau)] = 1 \tag{C.2}$$

where q_π can be thought of as a *selection factor* which quantifies the relative propensity of a TCR τ to engage with the target pMHC of interest π. The constraint that the expectation of q_π over the background TCR distribution is equal to one comes from the fact that the right-hand side of equation C.2 needs to be a well-defined probability distribution.

Plugging this re-expression back into equation C.1, we get:

$$\sum_{\tau \in \mathcal{T}} \sum_{\tau' \in \mathcal{T}} p_B(\tau) p_{S|\Pi}(\tau' | \pi) I_{d(\tau,\tau')=\delta} = \sum_{\tau' \in \mathcal{T}} p_B(\tau') q_\pi(\tau') \sum_{\tau \in \mathcal{T}} p_B(\tau) I_{d(\tau,\tau')=\delta} \tag{C.3}$$

$$= \mathop{\mathbb{E}}_{\tau' \sim p_B} \left[q_\pi(\tau') n_\delta(\tau') \right] \tag{C.4}$$

[1] For a reminder of the notation used, please refer back to chapter 5.

where:

$$n_\delta(\tau) \coloneqq \sum_{\tau' \in \mathcal{T}} p_B(\tau') I_{d(\tau,\tau')=\delta} \tag{C.5}$$

is the total probability mass of all TCRs that are a certain distance δ away from some anchor TCR τ.

Imagine that we have a random variable in TCR sequence space $T \sim p_B$. Then, $q_\pi(T)$ and $n_\delta(T)$ are both also random variables. If we assume that $q_\pi(T)$ and $n_\delta(T)$ are *independent* random variables, then we get:

$$\begin{aligned}
\mathbb{E}\left[q_\pi(T) n_\delta(T)\right] &= \mathbb{E}\left[q_\pi(T)\right] \mathbb{E}\left[n_\delta(T)\right] && \text{(C.6)}\\
&= \mathbb{E}\left[n_\delta(T)\right] && \text{(C.7)}\\
&= \sum_{\tau \in \mathcal{T}} p_B(\tau) \sum_{\tau' \in \mathcal{T}} p_B(\tau') I_{d(\tau,\tau')=\delta} && \text{(C.8)}\\
&= p_C[p_B](\delta) && \text{(C.9)}
\end{aligned}$$

where line C.6 results from the assumption of independence, line C.7 follows from the constraint on q_π in equation C.2, line C.8 expands n_δ according to its definition, and the final line follows from the definition of $p_C[\cdot](\cdot)$.

In other words, the approximation in equation C.1 is exact if the propensity of a TCR to engage the pMHC π is independent of the probabilities of generation of its neighbours at distance δ. With smaller values of δ, $n_\delta(T)$ is roughly proportional to $p_B(T)$, since the generation probabilities of TCRs that are similar in sequence tend to be close to one another. There is currently no evidence to suggest any relationship between the generation probability of a TCR and its ability to bind specific pMHCs. Therefore, at small values of δ – which is practically the only regime in which any real information about the specificity of a query TCR can be gained – the assumption that $q_\pi(T)$ and $n_\delta(T)$ are independent should be a reasonable one, and the approximation should be justified.

Bibliography

[1] Murphy M Kenneth and Weaver Casey. *Janeway's Immunobiology: Tenth International Student Edition with Registration Card*. en. Google-Books-ID: uiabzgEACAAJ. W.W. Norton & Company, June 2022. ISBN: 978-0-393-88491-3.

[2] Hongbo Chi, Marion Pepper, and Paul G. Thomas. "Principles and therapeutic applications of adaptive immunity". English. In: *Cell* 187.9 (Apr. 2024). Publisher: Elsevier, pp. 2052–2078. ISSN: 0092-8674, 1097-4172. DOI: 10.1016/j.cell.2024.03.037. URL: https://www.cell.com/cell/abstract/S0092-8674(24)00353-2 (visited on 07/08/2024).

[3] Yuta Nagano and Benjamin Chain. "tidytcells: standardizer for TR/MH nomenclature". English. In: *Frontiers in Immunology* 14 (Oct. 2023). Publisher: Frontiers. ISSN: 1664-3224. DOI: 10.3389/fimmu.2023.1276106. URL: https://www.frontiersin.org/journals/immunology/articles/10.3389/fimmu.2023.1276106/full (visited on 04/17/2024).

[4] M. M. Davis and P. J. Bjorkman. "T-cell antigen receptor genes and T-cell recognition". eng. In: *Nature* 334.6181 (Aug. 1988), pp. 395–402. ISSN: 0028-0836. DOI: 10.1038/334395a0.

[5] Marie-Paule Lefranc and Gérard Lefranc. *The T Cell Receptor FactsBook*. English. London, UK: Academic Press, 2001. ISBN: 978-0-12-441352-8.

[6] Marie-Paule Lefranc. "Immunoglobulin and T Cell Receptor Genes: IMGT® and the Birth and Rise of Immunoinformatics". In: *Frontiers in Immunology* 5 (2014). ISSN: 1664-3224. URL: https://www.frontiersin.org/articles/10.3389/fimmu.2014.00022 (visited on 09/27/2023).

[7] Anand Murugan et al. "Statistical inference of the generation probability of T-cell receptors from sequence repertoires". In: *Proceedings of the National Academy of Sciences* 109.40 (Oct. 2012). Publisher: Proceedings of the National Academy of Sciences, pp. 16161–16166. DOI: 10.1073/pnas.1212755109. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1212755109 (visited on 04/12/2024).

[8] Thierry Mora and Aleksandra M. Walczak. "Quantifying lymphocyte receptor diversity". In: *Systems Immunology*. Num Pages: 16. CRC Press, 2019. ISBN: 978-1-315-11984-7.

[9] Martina Milighetti et al. *Intra- and inter-chain contacts determine TCR specificity: applying protein co-evolution methods to TCR$\alpha\beta$ pairing*. en. Pages: 2024.05.24.595718 Section: New Results. May 2024. DOI: 10.1101/2024.05.24.595718. URL: https://www.biorxiv.org/content/10.1101/2024.05.24.595718v1 (visited on 06/28/2024).

[10] Andreas Mayer et al. "How a well-adapted immune system is organized". In: *Proceedings of the National Academy of Sciences* 112.19 (May 2015). Publisher: Proceedings of the National Academy of Sciences, pp. 5950–5955. DOI: 10.1073/pnas.1421827112. URL: https://www.pnas.org/doi/10.1073/pnas.1421827112 (visited on 07/08/2024).

[11] Wilfred Ndifon et al. "Chromatin conformation governs T-cell receptor Jβ gene segment usage". In: *Proceedings of the National Academy of Sciences* 109.39 (Sept. 2012). Publisher: Proceedings of the National Academy of Sciences, pp. 15865–15870. DOI: 10.1073/pnas.1203916109. URL: https://www.pnas.org/doi/10.1073/pnas.1203916109 (visited on 07/04/2024).

[12] Zachary Sethna et al. "Population variability in the generation and selection of T-cell repertoires". en. In: *PLOS Computational Biology* 16.12 (Dec. 2020). Publisher: Public Library of Science, e1008394. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1008394. URL: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008394 (visited on 07/19/2024).

[13] Michael Widrich et al. *Modern Hopfield Networks and Attention for Immune Repertoire Classification*. en. Pages: 2020.04.12.038158 Section: New Results. Aug. 2020. DOI: 10.1101/2020.04.12.038158. URL: https://www.biorxiv.org/content/10.1101/2020.04.12.038158v3 (visited on 12/12/2023).

[14] Maxim E. Zaslavsky et al. *Disease diagnostics using machine learning of immune receptors*. en. Pages: 2022.04.26.489314 Section: New Results. Jan. 2023. DOI: 10.1101/2022.04.26.489314. URL: https://www.biorxiv.org/content/10.1101/2022.04.26.489314v2 (visited on 02/09/2023).

[15] Yu Zhao et al. "DeepAIR: A deep learning framework for effective integration of sequence and 3D structure to enable adaptive immune receptor analysis". In: *Science Advances* 9.32 (Aug. 2023). Publisher: American Association for the Advancement of Science, eabo5128. DOI: 10.1126/sciadv.abo5128. URL: https://www.science.org/doi/10.1126/sciadv.abo5128 (visited on 06/20/2024).

[16] Daria Beshnova et al. "De novo prediction of cancer-associated T cell receptors for noninvasive cancer detection". In: *Science Translational Medicine* 12.557 (Aug. 2020). Publisher: American Association for the Advancement of Science, eaaz3738. DOI: 10.1126/scitranslmed.aaz3738. URL: https://www.science.org/doi/full/10.1126/scitranslmed.aaz3738 (visited on 06/07/2022).

[17] Min Li et al. "Lung cancer-associated T cell repertoire as potential biomarker for early detection of stage I lung cancer". en. In: *Lung Cancer* 162 (Dec. 2021), pp. 16–22. ISSN: 0169-5002. DOI: 10.1016/j.lungcan.2021.09.017. URL: https://www.sciencedirect.com/science/article/pii/S0169500221005535 (visited on 06/06/2022).

[18] Huaichao Luo et al. "Characteristics and significance of peripheral blood T-cell receptor repertoire features in patients with indeterminate lung nodules". en. In: *Signal Transduction and Targeted Therapy* 7.1 (Oct. 2022). Number: 1 Publisher: Nature Publishing Group, pp. 1–4. ISSN: 2059-3635. DOI: 10.1038/s41392-022-01169-7. URL: https://www.nature.com/articles/s41392-022-01169-7 (visited on 11/24/2022).

[19] TRACERx consortium et al. "Spatial heterogeneity of the T cell receptor repertoire reflects the mutational landscape in lung cancer". en. In: *Nature Medicine* 25.10 (Oct. 2019), pp. 1549–1559. ISSN: 1078-8956, 1546-170X. DOI: 10.1038/s41591-019-0592-2. URL: http://www.nature.com/articles/s41591-019-0592-2 (visited on 05/19/2022).

[20] Dan Hudson et al. "Can we predict T cell specificity with digital biology and machine learning?" en. In: *Nature Reviews Immunology* (Feb. 2023). Publisher: Nature Publishing Group, pp. 1–11. ISSN: 1474-1741. DOI: 10.1038/s41577-023-00835-3. URL: https://

www.nature.com/articles/s41577-023-00835-3 (visited on 02/09/2023).

[21] Grant Lythe et al. "How many TCR clonotypes does a body maintain?" In: *Journal of Theoretical Biology* 389 (Jan. 2016), pp. 214–224. ISSN: 0022-5193. DOI: 10.1016/j.jtbi.2015.10.016. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4678146/ (visited on 07/04/2024).

[22] Andreas Mayer and Curtis G. Callan. "Measures of epitope binding degeneracy from T cell receptor repertoires". In: *Proceedings of the National Academy of Sciences* 120.4 (Jan. 2023). Publisher: Proceedings of the National Academy of Sciences, e2213264120. DOI: 10.1073/pnas.2213264120. URL: https://www.pnas.org/doi/10.1073/pnas.2213264120 (visited on 03/30/2023).

[23] Dmitry V Bagaev et al. "VDJdb in 2019: database extension, new analysis infrastructure and a T-cell receptor motif compendium". In: *Nucleic Acids Research* 48.D1 (Jan. 2020), pp. D1057–D1062. ISSN: 0305-1048. DOI: 10.1093/nar/gkz874. URL: https://doi.org/10.1093/nar/gkz874 (visited on 06/06/2022).

[24] *IEDB.org: Free epitope database and prediction resource*. en. URL: http://www.iedb.org (visited on 03/30/2023).

[25] Anastasia A. Minervina et al. "SARS-CoV-2 antigen exposure history shapes phenotypes and specificity of memory CD8+ T cells". en. In: *Nature Immunology* 23.5 (May 2022). Number: 5 Publisher: Nature Publishing Group, pp. 781–790. ISSN: 1529-2916. DOI: 10.1038/s41590-022-01184-4. URL: https://www.nature.com/articles/s41590-022-01184-4 (visited on 11/21/2022).

[26] Anna Weber, Aurélien Pélissier, and María Rodríguez Martínez. *T cell receptor binding prediction: A machine learning revolution*. arXiv:2312.16594 [q-bio]. Dec. 2023. DOI: 10.48550/arXiv.2312.16594. URL: http://arxiv.org/abs/2312.16594 (visited on 01/11/2024).

[27] Ashish Vaswani et al. *Attention Is All You Need*. en. Number: arXiv:1706.03762 arXiv:1706.03762 [cs]. Dec. 2017. URL: http://arxiv.org/abs/1706.03762 (visited on 05/19/2022).

[28] Taku Kudo and John Richardson. *SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing*. arXiv:1808.06226 [cs]. Aug. 2018. DOI: 10.48550/arXiv.1808.06226. URL: http://arxiv.org/abs/1808.06226 (visited on 02/09/2023).

[29] Yuta Nagano et al. *Contrastive learning of T cell receptor representations*. arXiv:2406.06397 [cs, q-bio] version: 1. June 2024. DOI: 10.48550/arXiv.2406.06397. URL: http://arxiv.org/abs/2406.06397 (visited on 06/25/2024).

[30] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. en. Number: arXiv:1810.04805 arXiv:1810.04805 [cs]. May 2019. URL: http://arxiv.org/abs/1810.04805 (visited on 05/21/2022).

[31] Alec Radford et al. "Improving Language Understanding by Generative Pre- Training". en. In: (June 2018).

[32] Long Ouyang et al. *Training language models to follow instructions with human feedback*. arXiv:2203.02155 [cs]. Mar. 2022. DOI: 10.48550/arXiv.2203.02155. URL: http://arxiv.org/abs/2203.02155 (visited on 07/18/2024).

[33] Yuntao Bai et al. *Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback*. arXiv:2204.05862 [cs]. Apr. 2022. DOI: 10.48550/arXiv.2204.05862. URL: http://arxiv.org/abs/2204.05862 (visited on 07/18/2024).

[34] Gemini Team et al. *Gemini: A Family of Highly Capable Multimodal Models*. en. arXiv:2312.11805 [cs]. June 2024. URL: http://arxiv.org/abs/2312.11805 (visited on 07/18/2024).

[35] Rajul Parikh et al. "Understanding and using sensitivity, specificity and predictive values". In: *Indian Journal of Ophthalmology* 56.1 (2008), pp. 45–50. ISSN: 0301-4738. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2636062/ (visited on 07/20/2024).

[36] David Powers. "Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation". In: *Mach. Learn. Technol.* 2 (Jan. 2008).

[37] Tom Fawcett. "An introduction to ROC analysis". In: *Pattern Recognition Letters*. ROC Analysis in Pattern Recognition 27.8 (June 2006), pp. 861–874. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2005.10.010. URL: https://www.sciencedirect.com/science/article/pii/S016786550500303X (visited on 07/20/2024).

[38] Sofie Gielis et al. "Detection of Enriched T Cell Epitope Specificity in Full T Cell Receptor Sequence Repertoires". In: *Frontiers in Immunology* 10 (2019). ISSN: 1664-3224. URL: https://www.frontiersin.org/articles/10.3389/fimmu.2019.02820 (visited on 11/28/2023).

[39] Yao Tong et al. "SETE: Sequence-based Ensemble learning approach for TCR Epitope binding prediction". In: *Computational Biology and Chemistry* 87 (Aug. 2020), p. 107281. ISSN: 1476-9271. DOI: 10.1016/j.compbiolchem.2020.107281. URL: https://www.sciencedirect.com/science/article/pii/S1476927120303194 (visited on 12/11/2023).

[40] Emmi Jokinen et al. "Predicting recognition between T cell receptors and epitopes with TCRGP". en. In: *PLOS Computational Biology* 17.3 (Mar. 2021). Publisher: Public Library of Science, e1008814. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1008814. URL: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008814 (visited on 12/10/2023).

[41] David S Fischer et al. "Predicting antigen specificity of single T cells based on TCR CDR3 regions". In: *Molecular Systems Biology* 16.8 (Aug. 2020). Publisher: John Wiley & Sons, Ltd, e9416. ISSN: 1744-4292. DOI: 10.15252/msb.20199416. URL: https://www.embopress.org/doi/full/10.15252/msb.20199416 (visited on 12/10/2023).

[42] John-William Sidhom et al. "DeepTCR is a deep learning framework for revealing sequence concepts within T-cell repertoires". en. In: *Nature Communications* 12.1 (Mar. 2021). Number: 1 Publisher: Nature Publishing Group, p. 1605. ISSN: 2041-1723. DOI: 10.1038/s41467-021-21879-w. URL: https://www.nature.com/articles/s41467-021-21879-w (visited on 06/23/2022).

[43] Emmi Jokinen et al. "TCRconv: predicting recognition between T cell receptors and epitopes using contextualized motifs". In: *Bioinformatics* 39.1 (Jan. 2023), btac788. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btac788. URL: https://doi.org/10.1093/bioinformatics/btac788 (visited on 12/10/2023).

[44] Kevin Wu et al. *TCR-BERT: learning the grammar of T-cell receptors for flexible antigen-xbinding analyses*. en. preprint. Bioinformatics, Nov. 2021. DOI: 10.1101/2021.11.18.469186. URL: http://biorxiv.org/lookup/doi/10.1101/2021.11.18.469186 (visited on 05/21/2022).

[45] Giancarlo Croce et al. "Deep learning predictions of TCR-epitope interactions reveal epitope-specific chains in dual alpha T cells". en. In: *Nature Communications* 15.1 (Apr. 2024). Publisher: Nature Publishing Group, p. 3211. ISSN: 2041-1723. DOI: 10.1038/s41467-

024-47461-8. URL: https://www.nature.com/articles/s41467-024-47461-8 (visited on 06/20/2024).

[46] Pieter Meysman et al. "Benchmarking solutions to the T-cell receptor epitope prediction problem: IMMREP22 workshop report". en. In: *ImmunoInformatics* 9 (Mar. 2023), p. 100024. ISSN: 2667-1190. DOI: 10.1016/j.immuno.2023.100024. URL: https://www.sciencedirect.com/science/article/pii/S2667119023000046 (visited on 02/20/2023).

[47] Lihua Deng et al. "Performance comparison of TCR-pMHC prediction tools reveals a strong data dependency". In: *Frontiers in Immunology* 14 (2023). ISSN: 1664-3224. URL: https://www.frontiersin.org/articles/10.3389/fimmu.2023.1128326 (visited on 11/27/2023).

[48] Vanessa Isabell Jurtz et al. *NetTCR: sequence-based prediction of TCR binding to peptide- MHC complexes using convolutional neural networks*. en. Pages: 433706 Section: New Results. Oct. 2018. DOI: 10.1101/433706. URL: https://www.biorxiv.org/content/10.1101/433706v1 (visited on 12/07/2023).

[49] Alessandro Montemurro et al. "NetTCR-2.0 enables accurate prediction of TCR-peptide binding by using paired TCRα and β sequence data". en. In: *Communications Biology* 4.1 (Sept. 2021). Number: 1 Publisher: Nature Publishing Group, pp. 1–13. ISSN: 2399-3642. DOI: 10.1038/s42003-021-02610-3. URL: https://www.nature.com/articles/s42003-021-02610-3 (visited on 12/07/2023).

[50] Alan M. Luu et al. "Predicting TCR-Epitope Binding Specificity Using Deep Metric Learning and Multimodal Learning". In: *Genes* 12.4 (Apr. 2021), p. 572. ISSN: 2073-4425. DOI: 10.3390/genes12040572. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8071129/ (visited on 12/06/2023).

[51] Anna Weber, Jannis Born, and María Rodriguez Martínez. "TITAN: T-cell receptor specificity prediction with bimodal attention networks". In: *Bioinformatics* 37.Supplement_1 (July 2021), pp. i237–i244. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btab294. URL: https://doi.org/10.1093/bioinformatics/btab294 (visited on 02/22/2023).

[52] Pieter Moris et al. "Current challenges for unseen-epitope TCR interaction prediction and a new perspective derived from image classification". In: *Briefings in Bioinformatics* 22.4 (July 2021), bbaa318. ISSN: 1477-4054. DOI: 10.1093/bib/bbaa318. URL: https://doi.org/10.1093/bib/bbaa318 (visited on 02/22/2023).

[53] Tianshi Lu et al. "Deep learning-based prediction of the T cell receptor–antigen binding specificity". en. In: *Nature Machine Intelligence* 3.10 (Oct. 2021), pp. 864–875. ISSN: 2522-5839. DOI: 10.1038/s42256-021-00383-2. URL: https://www.nature.com/articles/s42256-021-00383-2 (visited on 05/21/2022).

[54] Xingang Peng et al. "Characterizing the interaction conformation between T-cell receptors and epitopes with deep learning". en. In: *Nature Machine Intelligence* 5.4 (Apr. 2023). Number: 4 Publisher: Nature Publishing Group, pp. 395–407. ISSN: 2522-5839. DOI: 10.1038/s42256-023-00634-4. URL: https://www.nature.com/articles/s42256-023-00634-4 (visited on 11/28/2023).

[55] Yoram Louzoun. "Prediction of Specific TCR-Peptide Binding From Large Dictionaries of TCR-Peptide Pairs". en. In: *Frontiers in Immunology* 11 (2020), p. 10.

[56] Ido Springer, Nili Tickotsky, and Yoram Louzoun. "Contribution of T Cell Receptor Alpha and Beta CDR3, MHC Typing, V and J Genes to Peptide Binding Prediction". en. In: *Frontiers in Immunology* 12 (Apr. 2021), p. 664514. ISSN: 1664-3224. DOI: 10.3389/fimmu.2021.664514. URL: https://www.frontiersin.org/articles/10.3389/fimmu.2021.664514/full (visited on 05/21/2022).

[57] Yuepeng Jiang, Miaozhe Huo, and Shuai Cheng Li. "TEINet: a deep learning framework for prediction of TCR–epitope binding specificity". In: *Briefings in Bioinformatics* 24.2 (Mar. 2023), bbad086. ISSN: 1477-4054. DOI: 10.1093/bib/bbad086. URL: https://doi.org/10.1093/bib/bbad086 (visited on 12/06/2023).

[58] Michael Cai et al. "ATM-TCR: TCR-Epitope Binding Affinity Prediction Using a Multi-Head Self-Attention Model". In: *Frontiers in Immunology* 13 (2022). ISSN: 1664-3224. URL: https://www.frontiersin.org/articles/10.3389/fimmu.2022.893247 (visited on 12/07/2023).

[59] Yicheng Gao et al. "Pan-Peptide Meta Learning for T-cell receptor–antigen binding recognition". en. In: *Nature Machine Intelligence* 5.3 (Mar. 2023). Number: 3 Publisher: Nature Publishing Group, pp. 236–249. ISSN: 2522-5839. DOI: 10.1038/s42256-023-00619-3. URL: https://www.nature.com/articles/s42256-023-00619-3 (visited on 12/08/2023).

[60] Bjørn P. Y. Kwee et al. *STAPLER: Efficient learning of TCR-peptide specificity prediction from full-length TCR-peptide data*. en. Pages: 2023.04.25.538237 Section: New Results. Apr. 2023. DOI: 10.1101/2023.04.25.

538237. URL: https://www.biorxiv.org/content/10.1101/2023.04.25.538237v1 (visited on 01/06/2024).

[61] Barthelemy Meynard-Piganeau et al. *TULIP — a Transformer based Unsupervised Language model for Interacting Peptides and T-cell receptors that generalizes to unseen epitopes*. en. Pages: 2023.07.19.549669 Section: New Results. July 2023. DOI: 10.1101/2023.07.19.549669. URL: https://www.biorxiv.org/content/10.1101/2023.07.19.549669v1 (visited on 01/05/2024).

[62] My-Diem Nguyen Pham et al. "epiTCR: a highly sensitive predictor for TCR–peptide binding". In: *Bioinformatics* 39.5 (May 2023), btad284. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btad284. URL: https://doi.org/10.1093/bioinformatics/btad284 (visited on 12/08/2023).

[63] Emilio Dorigatti et al. *Predicting T Cell Receptor Functionality against Mutant Epitopes*. en. May 2023. DOI: 10.1101/2023.05.10.540189. URL: http://biorxiv.org/lookup/doi/10.1101/2023.05.10.540189 (visited on 05/14/2024).

[64] Philip Bradley. "Structure-based prediction of T cell receptor:peptide-MHC interactions". In: *eLife* 12 (Jan. 2023). Ed. by Michael L Dustin, Tadatsugu Taniguchi, and Michael L Dustin. Publisher: eLife Sciences Publications, Ltd, e82813. ISSN: 2050-084X. DOI: 10.7554/eLife.82813. URL: https://doi.org/10.7554/eLife.82813 (visited on 04/19/2023).

[65] Xingcheng Lin et al. "Rapid assessment of T-cell receptor specificity of the immune repertoire". en. In: *Nature Computational Science* 1.5 (May 2021). Number: 5 Publisher: Nature Publishing Group, pp. 362–373. ISSN: 2662-8457. DOI: 10.1038/s43588-021-00076-1. URL: https://www.nature.com/articles/s43588-021-00076-1 (visited on 12/07/2023).

[66] Zhaochun Xu et al. "DLpTCR: an ensemble deep learning framework for predicting immunogenic peptide recognized by T cell receptor". In: *Briefings in Bioinformatics* 22.6 (Nov. 2021), bbab335. ISSN: 1477-4054. DOI: 10.1093/bib/bbab335. URL: https://doi.org/10.1093/bib/bbab335 (visited on 11/28/2023).

[67] Filippo Grazioli et al. "On TCR binding predictors failing to generalize to unseen peptides". In: *Frontiers in Immunology* 13 (2022). ISSN: 1664-3224. URL: https://www.frontiersin.org/articles/10.3389/fimmu.2022.1014256 (visited on 11/28/2023).

[68] Pradyot Dash et al. "Quantifiable predictive features define epitope-specific T cell receptor repertoires". en. In: *Nature* 547.7661 (July 2017). Number: 7661 Publisher: Nature Publishing Group, pp. 89–93. ISSN: 1476-4687. DOI: 10.1038/nature22383. URL: https://www.nature.com/articles/nature22383 (visited on 05/19/2022).

[69] Hongyi Zhang et al. "Investigation of Antigen-Specific T-Cell Receptor Clusters in Human Cancers". In: *Clinical Cancer Research* 26.6 (Mar. 2020), pp. 1359–1371. ISSN: 1078-0432. DOI: 10.1158/1078-0432.CCR-19-3249. URL: https://doi.org/10.1158/1078-0432.CCR-19-3249 (visited on 12/06/2023).

[70] Ryan Ehrlich et al. "SwarmTCR: a computational approach to predict the specificity of T cell receptors". en. In: *BMC Bioinformatics* 22.1 (Dec. 2021). Number: 1 Publisher: BioMed Central, pp. 1–14. ISSN: 1471-2105. DOI: 10.1186/s12859-021-04335-w. URL: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-021-04335-w (visited on 12/05/2023).

[71] William D. Chronister et al. "TCRMatch: Predicting T-Cell Receptor Specificity Based on Sequence Similarity to Previously Characterized Receptors". In: *Frontiers in Immunology* 12 (2021). ISSN: 1664-3224. URL: https://www.frontiersin.org/articles/10.3389/fimmu.2021.640725 (visited on 12/06/2023).

[72] Jacob Glanville et al. "Identifying specificity groups in the T cell receptor repertoire". en. In: *Nature* 547.7661 (July 2017), pp. 94–98. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature22976. URL: http://www.nature.com/articles/nature22976 (visited on 05/21/2022).

[73] S Henikoff and J G Henikoff. "Amino acid substitution matrices from protein blocks." In: *Proceedings of the National Academy of Sciences of the United States of America* 89.22 (Nov. 1992), pp. 10915–10919. ISSN: 0027-8424. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC50453/ (visited on 04/17/2024).

[74] Hongyi Zhang, Xiaowei Zhan, and Bo Li. "GIANA allows computationally-efficient TCR clustering and multi-disease repertoire classification by isometric transformation". en. In: *Nature Communications* 12.1 (Aug. 2021). Number: 1 Publisher: Nature Publishing Group, p. 4699. ISSN: 2041-1723. DOI: 10.1038/s41467-021-25006-7. URL: https://www.nature.com/articles/s41467-021-25006-7 (visited on 06/07/2023).

[75] Sebastiaan Valkiers et al. "ClusTCR: a python interface for rapid clustering of large sets of CDR3 sequences with unknown antigen specificity". In: *Bioinformatics* 37.24 (Dec. 2021), pp. 4865–4867. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btab446. URL: https://doi.org/10.1093/bioinformatics/btab446 (visited on 12/05/2023).

[76] Kristen E. Pauken et al. "TCR-sequencing in cancer and autoimmunity: barcodes and beyond". eng. In: *Trends in Immunology* 43.3 (Mar. 2022), pp. 180–194. ISSN: 1471-4981. DOI: 10.1016/j.it.2022.01.002.

[77] John-William Sidhom et al. "Deep learning reveals predictive sequence concepts within immune repertoires to immunotherapy". eng. In: *Science Advances* 8.37 (Sept. 2022), eabq5089. ISSN: 2375-2548. DOI: 10.1126/sciadv.abq5089.

[78] Aneesh Chandran et al. "Rapid synchronous type 1 IFN and virus-specific T cell responses characterize first wave non-severe SARS-CoV-2 infections". en. In: *Cell Reports Medicine* 3.3 (Mar. 2022), p. 100557. ISSN: 2666-3791. DOI: 10.1016/j.xcrm.2022.100557. URL: https://www.sciencedirect.com/science/article/pii/S2666379122000647 (visited on 06/23/2022).

[79] Hannah Kockelbergh et al. "Utility of Bulk T-Cell Receptor Repertoire Sequencing Analysis in Understanding Immune Responses to COVID-19". eng. In: *Diagnostics (Basel, Switzerland)* 12.5 (May 2022), p. 1222. ISSN: 2075-4418. DOI: 10.3390/diagnostics12051222.

[80] Martina Milighetti et al. "Large clones of pre-existing T cells drive early immunity against SARS-COV-2 and LCMV infection". eng. In: *iScience* 26.6 (June 2023), p. 106937. ISSN: 2589-0042. DOI: 10.1016/j.isci.2023.106937.

[81] Leo Swadling et al. "Pre-existing polymerase-specific T cells expand in abortive seronegative SARS-CoV-2". eng. In: *Nature* 601.7891 (Jan. 2022), pp. 110–117. ISSN: 1476-4687. DOI: 10.1038/s41586-021-04186-8.

[82] Koshlan Mayer-Blackwell et al. "TCR meta-clonotypes for biomarker discovery with tcrdist3 enabled identification of public, HLA-restricted clusters of SARS-CoV-2 TCRs". In: *eLife* 10 (Nov. 2021). Ed. by Benny Chain et al. Publisher: eLife Sciences Publications, Ltd, e68605. ISSN: 2050-084X. DOI: 10.7554/eLife.68605. URL: https://doi.org/10.7554/eLife.68605 (visited on 06/27/2024).

[83] Mikhail V. Pogorelyy et al. "Resolving SARS-CoV-2 CD4+ T cell specificity via reverse epitope discovery". English. In: *Cell Reports Medicine* 3.8 (Aug. 2022). Publisher: Elsevier. ISSN: 2666-3791. DOI: 10.1016/j.xcrm.2022.100697. URL: https://www.cell.com/cell-reports-medicine/abstract/S2666-3791(22)00233-6 (visited on 10/18/2023).

[84] *AIRR Community — AIRR Standards 1.4 documentation*. URL: https://docs.airr-community.org/en/stable/index.html (visited on 07/04/2023).

[85] Véronique Giudicelli, Denys Chaume, and Marie-Paule Lefranc. "IMGT/GENE-DB: a comprehensive database for human and mouse immunoglobulin and T cell receptor genes". In: *Nucleic Acids Research* 33.suppl_1 (Jan. 2005), pp. D256–D261. ISSN: 0305-1048. DOI: 10.1093/nar/gki010. URL: https://doi.org/10.1093/nar/gki010 (visited on 09/27/2023).

[86] Connor S. Dobson et al. "Antigen identification and high-throughput interaction mapping by reprogramming viral entry". en. In: *Nature Methods* 19.4 (Apr. 2022). Publisher: Nature Publishing Group, pp. 449–460. ISSN: 1548-7105. DOI: 10.1038/s41592-022-01436-z. URL: https://www.nature.com/articles/s41592-022-01436-z (visited on 07/08/2024).

[87] Alok V. Joglekar et al. "T cell antigen discovery via signaling and antigen-presenting bifunctional receptors". en. In: *Nature Methods* 16.2 (Feb. 2019). Publisher: Nature Publishing Group, pp. 191–198. ISSN: 1548-7105. DOI: 10.1038/s41592-018-0304-8. URL: https://www.nature.com/articles/s41592-018-0304-8 (visited on 07/08/2024).

[88] Romi Goldner Kabeli et al. "Self-supervised learning of T cell receptor sequences exposes core properties for T cell membership". In: *Science Advances* 10.17 (Apr. 2024). Publisher: American Association for the Advancement of Science, eadk4670. DOI: 10.1126/sciadv.adk4670. URL: https://www.science.org/doi/10.1126/sciadv.adk4670 (visited on 07/08/2024).

[89] Tom B. Brown et al. *Language Models are Few-Shot Learners*. arXiv:2005.14165 [cs]. July 2020. DOI: 10.48550/arXiv.2005.14165. URL: http://arxiv.org/abs/2005.14165 (visited on 04/27/2023).

[90] Alexander Rives et al. "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences". In: *Proceedings of the National Academy of Sciences* 118.15 (Apr. 2021). Publisher: Proceedings of the National Academy of Sciences, e2016239118. DOI: 10.1073/pnas.2016239118. URL: https://www.pnas.org/doi/10.1073/pnas.2016239118 (visited on 07/08/2024).

[91] Zeming Lin et al. "Evolutionary-scale prediction of atomic-level protein structure with a language model". In: *Science* 379.6637 (Mar. 2023). Publisher: American Association for the Advancement of Science, pp. 1123–1130. DOI: 10.1126/science.ade2574. URL: https://www.science.org/doi/full/10.1126/science.ade2574 (visited on 04/06/2024).

[92] Ahmed Elnaggar et al. "ProtTrans: Toward Understanding the Language of Life Through Self-Supervised Learning". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.10 (Oct. 2022). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 7112–7127. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2021.3095381. URL: https://ieeexplore.ieee.org/document/9477085 (visited on 04/06/2024).

[93] Ahmed Elnaggar et al. *Ankh: Optimized Protein Language Model Unlocks General-Purpose Modelling*. arXiv:2301.06568 [cs, q-bio]. Jan. 2023. DOI: 10.48550/arXiv.2301.06568. URL: http://arxiv.org/abs/2301.06568 (visited on 05/22/2024).

[94] Kevin E. Wu, Howard Chang, and James Zou. *ProteinCLIP: enhancing protein language models with natural language*. en. May 2024. DOI: 10.1101/2024.05.14.594226. URL: http://biorxiv.org/lookup/doi/10.1101/2024.05.14.594226 (visited on 05/24/2024).

[95] Francesca-Zhoufan Li et al. *Feature Reuse and Scaling: Understanding Transfer Learning with Protein Language Models*. en. Feb. 2024. DOI: 10.1101/2024.02.05.578959. URL: http://biorxiv.org/lookup/doi/10.1101/2024.02.05.578959 (visited on 05/22/2024).

[96] Meng Wang et al. "Language model-based B cell receptor sequence embeddings can effectively encode receptor specificity". In: *Nucleic Acids Research* 52.2 (Jan. 2024), pp. 548–557. ISSN: 0305-1048. DOI: 10.1093/nar/gkad1128. URL: https://doi.org/10.1093/nar/gkad1128 (visited on 07/08/2024).

[97] Justin Barton et al. *A generative foundation model for antibody sequence understanding*. en. May 2024. DOI: 10.1101/2024.05.22.594943. URL: http://biorxiv.org/lookup/doi/10.1101/2024.05.22.594943 (visited on 05/31/2024).

[98] 10x Genomics. *A New Way of Exploring Immunity - Linking Highly Multiplexed Antigen Recognition to Immune Repertoire and Phenotype*. Mar. 2020.

[99] Wen Zhang et al. "A framework for highly multiplexed dextramer mapping and prediction of T cell receptor sequences to antigen specificity". In: *Science Advances* 7.20 (May 2021). Publisher: American Association for the Advancement of Science, eabf5835. DOI: 10.1126/sciadv.abf5835. URL: https://www.science.org/doi/10.1126/sciadv.abf5835 (visited on 12/11/2023).

[100] Alessandro Montemurro et al. "Benchmarking data-driven filtering for denoising of TCRpMHC single-cell data". en. In: *Scientific Reports* 13.1 (Sept. 2023). Publisher: Nature Publishing Group, p. 16147. ISSN: 2045-2322. DOI: 10.1038/s41598-023-43048-3. URL: https://www.nature.com/articles/s41598-023-43048-3 (visited on 04/25/2024).

[101] James Henderson et al. *Limits on Inferring T-cell Specificity from Partial Information*. arXiv:2404.12565 [cond-mat, q-bio]. Apr. 2024. DOI: 10.48550/arXiv.2404.12565. URL: http://arxiv.org/abs/2404.12565 (visited on 04/29/2024).

[102] Andreas Tiffeau-Mayer. *Unbiased estimation of sampling variance for Simpson's diversity index*. arXiv:2310.03439 [cond-mat, q-bio]. Feb. 2024. DOI: 10.48550/arXiv.2310.03439. URL: http://arxiv.org/abs/2310.03439 (visited on 07/08/2024).

[103] Tongzhou Wang and Phillip Isola. *Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere*. arXiv:2005.10242 [cs, stat]. Aug. 2022. DOI: 10.48550/arXiv.2005.10242. URL: http://arxiv.org/abs/2005.10242 (visited on 10/24/2022).

[104] Prannay Khosla et al. *Supervised Contrastive Learning*. arXiv:2004.11362 [cs, stat]. Mar. 2021. DOI: 10.48550/arXiv.2004.11362. URL: http://arxiv.org/abs/2004.11362 (visited on 06/16/2023).

[105] Felix Drost, Lennard Schiefelbein, and Benjamin Schubert. *meTCRs - Learning a metric for T-cell receptors*. en. Pages: 2022.10.24.513533 Section: New Results. Oct. 2022. DOI: 10.1101/2022.10.24.513533. URL: https://www.biorxiv.org/content/10.1101/2022.10.24.513533v1 (visited on 06/08/2024).

[106] Margarita Pertseva et al. *TCR clustering by contrastive learning on antigen specificity*. en. Pages: 2024.04.04.587695 Section: New Results. Apr. 2024. DOI: 10.1101/2024.04.04.587695. URL: https://www.biorxiv.org/content/10.1101/2024.04.04.587695v1 (visited on 06/08/2024).

[107] Tianyu Gao, Xingcheng Yao, and Danqi Chen. *SimCSE: Simple Contrastive Learning of Sentence Embeddings*. arXiv:2104.08821 [cs]. May 2022. DOI: 10.48550/arXiv.2104.08821. URL: http://arxiv.org/abs/2104.08821 (visited on 04/17/2024).

[108] Yiming Fang, Xuejun Liu, and Hui Liu. "Attention-aware contrastive learning for predicting T cell receptor–antigen binding specificity". In: *Briefings in Bioinformatics* 23.6 (Nov. 2022), bbac378. ISSN: 1477-4054. DOI: 10.1093/bib/bbac378. URL: https://doi.org/10.1093/bib/bbac378 (visited on 06/08/2024).

[109] Bohan Li et al. "On the Sentence Embeddings from Pre-trained Language Models". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber et al. Online: Association for Computational Linguistics, Nov. 2020, pp. 9119–9130. DOI: 10.18653/v1/2020.emnlp-main.733. URL: https://aclanthology.org/2020.emnlp-main.733 (visited on 04/25/2024).

[110] Jesse D. Bloom et al. "Protein stability promotes evolvability". In: *Proceedings of the National Academy of Sciences* 103.15 (Apr. 2006). Publisher: Proceedings of the National Academy of Sciences, pp. 5869–5874. DOI: 10.1073/pnas.0510098103. URL: https://www.pnas.org/doi/10.1073/pnas.0510098103 (visited on 07/08/2024).

[111] Martin Weigt et al. "Identification of direct residue contacts in protein–protein interaction by message passing". In: *Proceedings of the National Academy of Sciences* 106.1 (Jan. 2009). Publisher: Proceedings of the National Academy of Sciences, pp. 67–72. DOI: 10.1073/pnas.0805923106. URL: https://www.pnas.org/doi/10.1073/pnas.0805923106 (visited on 07/08/2024).

[112] John Jumper et al. "Highly accurate protein structure prediction with AlphaFold". en. In: *Nature* 596.7873 (Aug. 2021). Publisher: Nature Publishing Group, pp. 583–589. ISSN: 1476-4687. DOI: 10.1038/s41586-021-03819-2. URL: https://www.nature.com/articles/s41586-021-03819-2 (visited on 07/08/2024).

[113] Paul G. Thomas and Jeremy Chase Crawford. "Selected before selection: A case for inherent antigen bias in the T-cell receptor repertoire". In: *Current Opinion in Systems Biology* 18 (Dec. 2019), pp. 36–43. ISSN: 2452-3100. DOI: 10.1016/j.coisb.2019.10.007. URL: https://www.sciencedirect.com/science/article/pii/S245231001930054X (visited on 07/08/2024).

[114] Yuval Elhanati et al. "Quantifying selection in immune receptor repertoires". In: *Proceedings of the National Academy of Sciences* 111.27 (July 2014). Publisher: Proceedings of the National Academy of Sciences, pp. 9875–9880. DOI: 10.1073/pnas.1409572111. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1409572111 (visited on 07/08/2024).

[115] Zachary Sethna et al. "OLGA: fast computation of generation probabilities of B- and T-cell receptor amino acid sequences and motifs". In: *Bioinformatics* 35.17 (Sept. 2019), pp. 2974–2981. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz035. URL: https://doi.org/10.1093/bioinformatics/btz035 (visited on 07/08/2024).

[116] Ceder Dens et al. "The pitfalls of negative data bias for the T-cell epitope specificity challenge". en. In: *Nature Machine Intelligence* 5.10 (Oct. 2023). Publisher: Nature Publishing Group, pp. 1060–1062. ISSN: 2522-5839. DOI: 10.1038/s42256-023-00727-0. URL: https://www.nature.com/articles/s42256-023-00727-0 (visited on 03/18/2024).

[117] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A unified embedding for face recognition and clustering". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. ISSN: 1063-6919. June 2015, pp. 815–823. DOI: 10.1109/CVPR.2015.7298682.

[118] Weiyang Liu et al. "SphereFace: Deep Hypersphere Embedding for Face Recognition". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. ISSN: 1063-6919. July 2017, pp. 6738–6746. DOI: 10.1109/CVPR.2017.713.

[119] Sheng Chen et al. *MobileFaceNets: Efficient CNNs for Accurate Real-Time Face Verification on Mobile Devices*. arXiv:1804.07573 [cs]. June 2018. DOI: 10.48550/arXiv.1804.07573. URL: http://arxiv.org/abs/1804.07573 (visited on 02/16/2023).

[120] Jiankang Deng et al. "ArcFace: Additive Angular Margin Loss for Deep Face Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021). arXiv:1801.07698 [cs], pp. 1–1. ISSN: 0162-8828, 2160-9292, 1939-3539. DOI: 10.1109/TPAMI.2021.3087709. URL: http://arxiv.org/abs/1801.07698 (visited on 02/16/2023).

[121] Hidetaka Tanno et al. "Determinants governing T cell receptor α/β-chain pairing in repertoire formation of identical twins". In: *Proceedings of the National Academy of Sciences* 117.1 (Jan. 2020). Publisher: Proceedings of the National Academy of Sciences, pp. 532–540. DOI: 10.1073/pnas.1915008117. URL: https://www.pnas.org/doi/10.1073/pnas.1915008117 (visited on 11/02/2022).

[122] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. arXiv:1412.6980 [cs]. Jan. 2017. DOI: 10.48550/arXiv.1412.6980. URL: http://arxiv.org/abs/1412.6980 (visited on 04/17/2024).

[123] Niclas Thomas et al. "Tracking global changes induced in the CD4 T-cell receptor repertoire by immunization with a complex antigen using short stretches of CDR3 protein sequence". In: *Bioinformatics* 30.22 (Nov. 2014), pp. 3181–3188. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btu523. URL: https://doi.org/10.1093/bioinformatics/btu523 (visited on 10/24/2022).

[124] Diederik P. Kingma and Max Welling. "An Introduction to Variational Autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4 (2019). arXiv:1906.02691 [cs, stat], pp. 307–392. ISSN: 1935-8237, 1935-8245. DOI: 10.1561/2200000056. URL: http://arxiv.org/abs/1906.02691 (visited on 07/18/2024).

[125] Jiacheng Xu and Greg Durrett. *Spherical Latent Spaces for Stable Variational Autoencoders*. arXiv:1808.10805 [cs]. Oct. 2018. DOI: 10.48550/arXiv.1808.10805. URL: http://arxiv.org/abs/1808.10805 (visited on 06/18/2024).

[126] James M Heather et al. "Stitchr: stitching coding TCR nucleotide sequences from V/J/CDR3 information". In: *Nucleic Acids Research* 50.12 (July 2022), e68. ISSN: 0305-1048. DOI: 10.1093/nar/gkac190. URL: https://doi.org/10.1093/nar/gkac190 (visited on 04/27/2024).