LONDON'S GLOBAL UNIVERSITY

Multi-Instance Transfer Learning on T cell receptor LLMs for Cancer Prediction

Rudy C. Yuen [1]

Department of Computer Science
Faculty of Engineering Sciences

Supervisors

Professor Benny Chain, Yuta Nagano
Division of Infection & Immunology
Faculty of Medical Sciences

Professor John Shawe-Taylor
Department of Computer Science
Faculty of Engineering Sciences

Submission Date: 18th April, 2024

[1]**Disclaimer:** This report is submitted as part requirement for Master of Engineering in Mathematical Computation at UCL. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.

Acknowledgements

First and foremost, I would like to thank my family for their financial and mental support throughout my academic journey. It would not have been possible for me to become who I am now without all your nurture. In particular, I thank my grandfather for motivating me to complete this project.

My most sincere gratitude goes to my supervisors, Yuta Nagano, Benny Chain, and John Shawe-Taylor, for their endless support and teachings during the project. I cannot be more thankful for your patience in teaching me the immunology and machine learning concepts that I needed to complete this project, as well as for going through this manuscript and providing valuable insights into areas of improvement. Your support has made the journey of my project immensely enjoyable and fruitful.

I would also like to thank all my peers who provided insights and ideas into this project.

Lastly, I thank all the patients who voluntarily provided valuable data towards this research. Without your generosity, this project would not have been possible.

Abstract

In the early stages of cancer, T cells undergo proliferation as part of an immune response to eradicate cancer. Thus, tracking peripheral T cells in asymptomatic patients for early cancer prediction is possible. Previous attempts at constructing such a model represent T cell receptors (TCRs) numerically using physico-chemical properties and pairing with downstream classification algorithms. We argue that the accuracy can be improved when TCR representations are learnt using a pre-trained language model as language models create numerical representations for TCRs based on the amino acid sequence, whereas physico-chemical properties only encode TCRs based on individual amino acids.

This article classifies symptomatic lung cancer patients from the TRACERx dataset and control patients to demonstrate that the output embeddings from pre-trained large protein language models are more expressive than physico-chemical properties. Using these pre-trained language model embeddings, we can achieve high classification performance between cancer and non-cancer repertoires, achieving 100% AUC on the test and evaluation set using a simple two-layered multi-instanced downstream classifier. Our results also propose that embeddings from language models are more expressive than physico-chemical properties in cancer prediction.

Contents

Chapter 1

Executive Summary

T cells play an essential role in the adaptive immune system. Their receptors bind to foreign antigens, causing an immune response during an invasion. One response includes proliferation, where the T cell receptor (TCR) concentration in blood increases to eradicate invaders. Therefore, a conjecture is presented: the increased concentration of this receptor also increases the probability of sampling a cancer-targeting TCR. Thus, analysing T cells circulating in the blood can provide insight into the invaders in the human when the blood is sampled.

Given this conjecture, many researchers model TCRs to understand their specificity and aim to identify diseases in an asymptomatic patient. TCRs are proteins which can be expressed as a string of amino acids. Hence, to represent TCRs numerically, many research use amino acids' physico-chemical properties to encode TCRs numerically. They mainly use Atchley factors, a series of numbers that specifies the physico-chemical properties of amino acids, such as polarity, secondary structure, molecular volume, codon diversity and electrostatic charge. Other usable physico-chemical properties include Amino Acid Properties and Kidera Factors, which model different properties of amino acids.

We define symbolic encoding as a means of representing TCRs numerically. Symbolic encodings encode each amino acid in the TCR with a high-level meaning, such as with physico-chemical properties. However, we hypothesise that this symbolic means of representing T cells numerically can be improved through subsymbolic methods. Subsymbolic Methods refer to encoding TCRs using pre-trained language models' embedding spaces.

This hypothesis originates from the fact that physico-chemical encodings are not specific to TCRs. These encodings are generalised methods of encoding amino acids into numerical variants. Physico-chemical properties are also unable to consider neighbouring TCRs to create an embedding.

On the other hand, pre-trained language models generate a TCR-specific embedding space through prior information. The generated embeddings would also consider neighbouring amino acids within the same TCR. Hence, we believe that LLMs can assign spatial locality to similar TCRs, effectively reducing the difficulty of classifiers in learning about TCR repertoires and cancer.

We test our hypothesis by applying transfer learning on two large language models pre-trained on TCR sequences. For robustness, we did not fine-tune parameters within the two large language models; instead, we trained a simple two-layered downstream model using the language models' output embeddings to identify whether a patient has cancer. The two pre-trained large language models will be TCR-BERT, which is pre-trained on masked amino acid modelling on CDR3s, and SCEPTR, an in-house yet-to-be-published model significantly smaller than TCR-BERT. SCEPTR is trained on masked amino acid modelling followed by auto-contrastive learning.

Our data are T cell receptors sampled from a blood draw for both the control and positive classes. The control class comes from two different studies, and we only use data before experimental intervention; the positive class is the TRACERx Dataset, which contains TCR sequences sampled from patients clinically diagnosed with NSCLC Stage I to IIIa. These patients are clinically diagnosed through a body check after exhibiting symptoms.

Since the amount of T cell receptors in each blood draw is not fixed, we apply a multiple-instance learning approach to our learning problem. Our methodology involves one scoring layer, which scores each TCR and subsequently assigns sparse attention to each TCR to obtain weights for each TCR amino acid sequence. We compute a weighted sum using these weights to get a repertoire representing vector. A binary classification on this repertoire representing vector is utilised to find the probability of cancer. The proposed classifier has two linear layers, demonstrating that even such a simple model can identify cancer patients accurately.

The classifier is trained on both TCR alpha and beta chain CDR3 sequences, and we obtain good classification performance on the model that uses SCEPTR's embeddings. In line with our hypothesis, SCEPTR's downstream classifiers perform better than models that encode TCRs using their physico-chemical amino acid properties.

Since TCR-BERT's embedding space is too high-dimensional, we do not have enough patient samples to train TCR-BERT's downstream classifier to mitigate the risk of overfitting. Hence, we focused on SCEPTR's downstream model, which gave an impressive performance of a maximum AUC of 100% on both the test and evaluation set, with a maximum AUC of 97.8% on the training set.

We have also studied one of the best-performing downstream models that used SCEPTR's embeddings. This model identified public TCRB sequences and expanded them within the evaluation set.

To take our research forward, we proposed methods to fine-tune the language models so the embedding space will be arranged by TCR specificity. Several other extensions to this task have also been introduced to improve the accuracy of identifying cancer with TCRs, and better evaluate whether symbolic encodings are less expressive than subsymbolic encodings.

Chapter 2

Introduction

Cancer is caused by mutations that cause cells to divide uncontrollably, consequentially creating a tumour of malfunctioned cells that disrupt ordinary operations in the body. Although cancer cells are recognised as immunologically foreign, cancer is tough to eradicate naturally as they create a protective environment on themselves that suppresses immune responses [1] such as apoptosis resistance [2]. This disease is responsible for every sixth death worldwide [3], which demonstrates its destructiveness.

Identifying early-stage cancer has been shown to increase survival rates significantly [4]. Yet, tumours are small during the early stages of cancer and typically will not spread around the body. This causes the patient to exhibit minimal symptoms or be completely asymptomatic, making it hard for patients or physicians to spot it.

With technological advancements and research into machine learning (ML), it has been demonstrated that ML can learn complicated patterns in many fields, such as biology [5]. Given its cheap cost to deploy instead of laboratory experiments, there is plenty of ongoing research to apply ML to early cancer diagnosis [6].

The promise in the future of deploying ML in clinical settings has been demonstrated through its success in detecting cancer [7, 8, 9]. Current state-of-the-art Machine Learning techniques for cancer detection samples patients' data through non-invasive means, such as Liquid Biopsy [10, 11, 12] and Medical Image Segmentation [13, 14].

2.1 T Cells

Recently, the topic of using T cell receptors (TCRs) to detect cancer has been brought up [15]. T cells are immune cells that are part of the adaptive immune system; they orchestrate immunological responses to detect foreign agents such as viruses, bacteria, or cancer cells. Ordinarily, T cells circulate the body to seek a foreign antigen that their receptor can bind to [16]. These foreign antigens are often found on infected or diseased cells or foreign cells such as bacteria.

The majority of T cells carry antigen receptors made up of two chains, alpha and beta chains, on their receptors. Each chain contains 3 Complementarily-Determining Regions (CDR1, 2 and 3), which determine the T cell's specificity. TCRs are formed from a V(D)J Recombination Process, a stochastic process. This process generates many T cell receptors, consequently enabling them to gain a wide range of specificities, allowing the T cell system to be able to target all possible foreign antigens [17].

Once a TCR successfully binds to an antigen or a peptide presented on a human cell's Major Histocompatibility Complex, the T cell proliferates exponentially. This will consequentially increase the concentration of the TCR with the same specificity in blood, implying that these activated TCRs are more likely to be sampled [18].

Thus, it has been hypothesised that, by analysing the set of T cells circulating in the blood, we can gain an insight into the human's functional state at the time of the blood draw. For example, we can learn whether the patient has cancer or is under an incubation period for a disease [19].

The complexity of understanding the functionality of any specific TCR is due to cross-reactivity, which states that one TCR can target several antigens, and multiple TCRs can recognise one antigen. Generally, the TCR that binds to the same antigen between two individuals will not be identical [20]. However, there has been some evidence which suggests that these two TCRs should be similar in their amino acid sequences [20, 21].

In cancer, this process is further complicated as mutations in cancer cells are not specific enough to guarantee that the antigens hosted on cancer cells are identical. Since there are variations with the cancer cells' presenting antigen, the TCR that binds to it will, therefore, be different [21]. However, there may be similarities within these TCRs amongst different patients; there have been some research attempts in analysing TCRs computationally, demonstrating a possibility of using TCRs to identify cancer in an asymptomatic patient [15, 22, 23].

2.2 Representing T Cells Numerically

T cells can be sampled from any part of the body. Within each T cell, there are typically many TCRs, each comprising 3 Complementary Determining Regions (CDR): CDR1, CDR2, and CDR3. Each of these can be represented by amino acids, which are strings.

To utilise TCRs to infer the patient's condition using a mathematical model, past literature has attempted to represent TCRs amino acid sequences numerically using meaningful quantifiers [23, 24], which we refer to as a symbolic encoding in this article since each number in this encoding carries a human-understandable meaning, for example, pH. These symbolic encoding methods include Atchley Factors [25], which quantifies amino acids based on five physico-chemical properties, as well as other symbolic encoding methods, including Kidera factors [26] and amino acids properties [27].

Whilst symbolic means to quantify TCRs, such as Atchley Factors, can encapsulate amino acids' physical and chemical properties, we argue that they cannot inform the computational model with more profound information about the TCR. These physico-chemical properties do not incorporate any information specific to TCRs but properties that apply to all amino acids. Furthermore, using physico-chemical properties to encode TCRs does not capture any interactions and dependencies between amino acids, whereby the symbolic numerical representation for a particular amino acid will be the same irrespective of where it is placed within the sequence.

In contrast, large language models (LLMs) can overcome these problems. LLMs encode each amino acid depending on its structure; therefore, the embedding for one amino acid sequence depends on where each amino acid is placed in the sequence. This is typically referred to as context-based embedding.

Suppose we have an LLM trained with TCRs. In that case, it will capture intrinsic properties within the TCR, generating a context-based embedding specific to the TCR instead of a universal embedding across all amino acids. We denote representing TCRs numerically using LLM's output embeddings as a subsymbolic encoding since each value within the vector that represents the TCR does not carry a human-understandable meaning about the TCR.

We propose using a pre-trained model instead of an LLM trained from scratch for cancer identification using TCRs. This is motivated by the observation that most previous works using TCRs to predict cancer do not pre-train their model with proteins or TCRs.

We assume that it is possible to create a meaningful subsymbolic encoding space for TCRs using LLMs, as it is a well-known fact that proteins exhibit a grammatical structure [28, 29, 30]. A similar study with T and B cell receptors has also used a subsymbolic encoding in one of their ensemble models to infer whether a patient has COVID-19, Lupus or HIV [31]. This subsymbolic encoding has achieved an 88.1% AUC and 71% accuracy in this multiclass classification problem, demonstrating that using subsymbolic encoding can be a promising methodology in a binary classification problem.

2.3 Research Aims

In this literature, we demonstrate with novelty that representing TCRs numerically subsymbolically through the use of TCR-BERT [32] and an in-house model SCEPTR, two different pre-trained large language models, can give a significantly better performance than symbolic encoding such as Kidera factors [26], Atchley factors [25] and Amino Acid Properties [27].

To show robustness, TCR-BERT and SCEPTR were never trained with disease specificity, and the dataset used to train them is unrelated to the dataset used in this literature. Therefore, the two encoding models' output embedding is unlikely to carry information about the disease specificity but just about the TCR. TCR-BERT was pre-trained on masked amino acid prediction on T cell CDR3 sequences. In contrast, SCEPTR is a BERT-based model pre-trained on V call sequences and CDR3 sequences from the alpha and beta chain using masked language modelling followed by auto-contrastive learning.

Since we argue that subsymbolic encoding provides a better quantification on T cells, we will freeze the encoding models and demonstrate that we can still obtain a higher performance score than symbolic encoding on a simple model. We measure classification performance by the Area Under the Receiver Operating Characteristic Curve (AUC).

2.4 Report Structure

This report is structured as follows:

In Chapter 3, we first demonstrate an identifiable difference between healthy patients' and cancer patients' TCR repertoires through an approach known as TCR Frequency Analysis. We then review current state-of-the-art approaches to predicting cancer by analysing TCR repertoires, and reviewing their weaknesses. We will also prove that proteins exhibit a grammatical structure and study TCR-BERT, a TCR Large Language Model.

Taking on Chapter 3's findings, we formulate a formal methodology in Chapter 4 that uses TCR Repertoires to predict cancer. We will also describe our experiment, which aims to show that subsymbolic encoding is more expressive than symbolic encoding. We then provide results from a simple model on each type of encoding.

In Chapter 5, we analyse the results obtained from the models in Chapter 4 and whether our results can shed light on our experimental hypothesis. Subsequently, we discuss weaknesses in our approach. We also look for areas of improvement in our current approach and provide guides on future work. We will also argue our models' robustness by interpreting the model and knowing to what extent the model has learnt this biological problem.

Chapter 6 concludes the report, summarising all our findings, limitations and future work.

The Executive Summary provides a technical summary with a more thorough project overview.

Chapter 3

Context & Related Works

To argue that it is possible to create a generalisable model that uses T cell receptors (TCRs) to predict cancer, we first demonstrate that amino acids, the building blocks of TCRs, exhibit a grammatical structure through literature that uses large language models to model proteins. This indicates that a linguistic approach to understanding TCRs is reasonable.

We then review literature using a motif-based approach, showing a distinguishable difference between healthy and cancer patients' TCR repertoires.

Subsequently, we study literature that uses machine learning to accomplish a similar task. All of these literature creates vectors that rigidly represent TCRs through physico-chemical properties. Although we argue that this representation is ineffective, we aim to learn about their approaches and how we could create a model that generalises this problem through language model embeddings.

Then, we will study transfer learning as an approach to classifying TCRs. This approach is promising because pre-trained models have already understood the underlying problem. We focus on TCR-BERT, a large language model pre-trained on TCR. Accompanied with what we argued about proteins exhibiting a language, this provides a comprehensive view of why using output embeddings from large language models to represent T cells numerically is a promising methodology for cancer prediction.

3.1 The Protein Language

Amino acids are the fundamental building block for TCRs. There are 20 different amino acids in the human body, and each amino acid is often referred to by 1 of the 26 English alphabets by biologists for convenience [28, 33]. For example, one TCR CDR3 amino acid sequence can be expressed as a string like 'CASRGLTGNYGYTF'.

A contiguous set of a few amino acids is referred to as a motif, which compares to a word in English. In [28], the authors used SentencePiece [34] to tokenise amino acids to find frequently occurring motifs in cancer patients' TCRs. It has been found that most motifs have sizes from 2 amino acids to 16, with most motifs being seven amino acids long. Although proteins operate in a three-dimensional space [35, 36] with contrast that languages are one-dimensional, there have been successful previous works that discard this three-dimensional structure by using conventional large language models to model the protein language.

For example, ProGen was trained in [29]. ProGen is a large language model that is trained to generate synthetic proteins. Upon fine-tuning, the generated proteins showed similar catalytic efficiencies as natural proteins. Yet, the generated protein showed only 34.1% similarity against their natural counterparts. This shows that proteins have an underlying language-like structure, as proteins can be synthetically generated through ProGen, a language model.

On the other hand, ProGPT-2 was proposed in [30]. ProGPT-2 is a deep unsupervised protein language model. This model can generate proteins with natural amino acid propensities but are distantly related to the natural ones sampled. The generated protein's topology is also not captured in databases that store naturally occurring protein structures. This further proves that proteins exhibit a language-like structure that is learnable by LLMs.

3.2 TCR Sequence Frequency Analysis

T cells mediate immune responses through binding onto foreign antigens that their receptor permits [16]. T cell receptors (TCRs) are produced in the thymus through a stochastic process called the Variable (V) Diversity (D) and Joining (J) Recombination. This produces a large population of distinct TCRs, enabling them to target all possible foreign antigens [17]. Each human hosts different sets of TCRs, and it has been shown that under the same immunological stimulus by the same antigen, the TCRs within different humans that target the same antigen will be different yet similar [20, 21].

When a foreign antigen is bound successfully, the T cell will proliferate, increasing the population of T cells of that specificity in the body. Proliferated T cells also differentiate to acquire various functions. Differentiated CD4 T cells are primarily involved in amplifying other types of immune cells, whereas differentiated CD8 T cells kill the target cells they recognise [18]. Although proliferation and differentiation mainly occur in tissues, T cells migrate around the body in blood to seek these foreign cells that may have spread around the body [37].

Hence, we can sample and measure peripheral TCRs stimulated by antigen exposure, telling us about the type of attack the body is under when the blood is sampled. TCRs have been hypothesised to be an indicator of the human's immune system's current state, such as whether the body is under any attack or if there are cancer cells inside the body [19].

To provide evidence for this hypothesis, studies analyse differences in TCRs' motifs occurrences within a TCR repertoire for patients under the same immunological stimulus against a control group. Since TCRs exhibit a grammatical structure as they comprise amino acids, it is sensible to analyse this difference linguistically through analysing motifs. This approach is often referred to as TCR Frequency Analysis.

One such study is [38], which measures the difference in the frequency of different CD4+ TCRs in mice that are immunised against *Mycobacterium tuberculosis* and those that are not. It has been discovered that SVM, a supervised learning technique and Hierarchical Clustering, an unsupervised learning technique, would both yield 100% efficiency in categorising TCR repertoires between immunised and unimmunised mice. This shows a significant difference between the two populations'

TCR repertoires. Although this study is not related to cancer, this study shows that there is a causal link between immune stimulus and a change in motif frequency within TCR repertoires.

Such studies imply that it might be possible to use TCRs to predict cancer since cancer stimulates an immune response. To further investigate the difference in motifs within TCR repertoires from cancer patients and control subjects, SentencePiece, a language-independent sub-word tokenizer [34] was used in [28] to generate tokens from TCR sequences. These tokens are created based on their occurrence frequency inside the repertoire; the tokenizer under the study was trained in an environment with minimal restrictions to identify tokens that can best represent the TCR sequence of any length. Sixteen motifs are identified within the TRACERx Lung Cancer Dataset [39] which are enriched in cancer patients compared to healthy patients.

However, TCR frequency analysis cannot conclude a generalisable relationship for cancer detection. The T cell system is a highly variable system within each individual; therefore, using TCR frequency analysis to generate a look-up table is ineffective. The differences in each human's T cell system further highlight the ineffectiveness of this approach, where this difference can originate from factors such as ethnicity, immune history and HLA types [20, 40].

Whilst TCR frequency analysis can demonstrate that there is a difference between cancer patients and control patients' TCR repertoires, it is analogous to comment on the use of TCR frequency analysis to classify whether a patient has cancer as classifying whether an email is workplace-appropriate by reviewing the sender's sitting posture.

3.3 Deep Learning on Physico-Chemical Properties

Since TCR Frequency Analysis is insufficient to predict cancer, deep learning methods have been used to predict cancer with TCRs. Deep learning is used as it has been demonstrated to capture complicated relationships [41, 42], which is what we need to uncover relationships between TCR repertoires and cancer. Amongst all current works, TCRs have been encoded using physico-chemical properties to the best of our knowledge.

These physico-chemical properties are derived from the amino acid index [43]. This index contains at least 566 amino acid indices that one can choose from, yet many are highly correlated and contain redundant information [24]. There have been several efforts to reduce the dimensionality of the amino acid index so we can use fewer numbers to represent amino acids whilst maintaining the majority of the information that best represents them. For example, the Kidera factor is a table of 10 features extracted from 188 amino acid indices [26], and the Atchley factors are a table of 5 factors using a series of techniques [25].

Using these methodologies to represent amino acids numerically, DeepCAT is one of the earliest studies to predict cancer using TCR repertoires. The authors used Convolutional Neural Networks to predict cancer. They have achieved an AUC of > 0.95 [15] after representing TCRs numerically with the Amino Acid Index [43]. The authors of the article found this AUC statistic to be unexpected. We believe that this could be explained through the training paradigm; it can be seen from Figure 3.1 that the sampling methodology of TCR sequences from cancer and healthy patients are different, which suggests a chance that the model has overfitted to classify between the two sampling methods, rather than the true relationship between TCR repertoires from cancer patients and control patients.

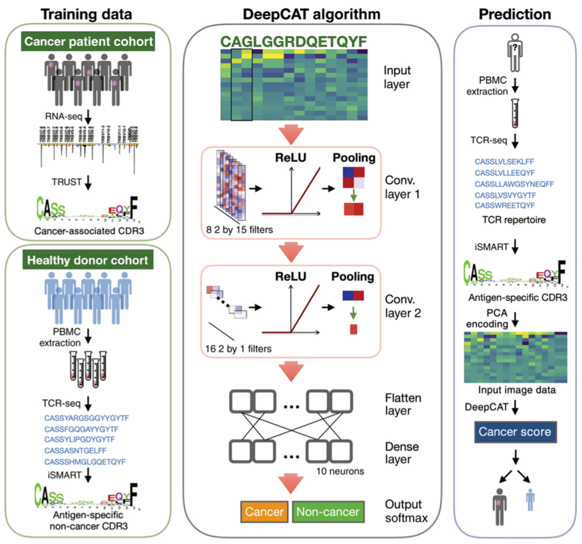


Figure 3.1: DeepCAT Training Paradigm [15]

On the other hand, in [22], the authors used blood TCRs from 146 patients, where they are either pathologically confirmed with Stage 1 lung cancer or non-cancer controls to train a Logistic Regression model. The authors used 28 samples in the training set and the remaining as the test set. The trained model had a significant discrepancy in the training and testing AUC, where they are 0.8 and 0.91, respectively. The training AUC of 0.8 for a binary classification problem is relatively low, so the reported model's training and testing data split might have placed the easier samples in the test set, or the model attained this high test AUC by chance. It is not convincing that this logistic regression model can generalise the problem since we would expect an optimally fitted model to have a similar, if not better, train AUC than test AUC.

In [24], the authors attempted to understand the difference between TCR CDR3 motifs from a tumour tissue and a healthy tissue within the same organ from the same patient. Through the analysis of X-ray crystallographic structures of human TCRs bound to peptide-MHC hosted on cancer cells, the authors located the amino acids within the TCR beta chain CDR3s involved in binding. Subsequently, the authors encoded four amino acids within TCRs at a time using the Atchley factors [25] and studied the importance of each factor within the Atchley factors for identifying cancer using a logistic regression model. The trained logistic regression model had a k-fold cross-validation classification accuracy of 93% and 94% for colorectal and breast cancer, respectively. This paper demonstrates that using these physico-chemical properties, such as Atchley factors, without any transformation could uncover certain information regarding TCR repertoires.

3.3.1 MINN-SA

With the appreciation that any machine learning algorithms used to tackle medical problems would need to be explainable, MINN-SA [23] has been designed to improve the explainability of models that are used to predict cancer with TCRs by using sparse attention to isolate the TCRs that might have a cancer specificity from those that are unrelated.

The authors represented each amino acid within a TCR CDR3 sequence numerically using the Atchley factors [25], where they obtain an Atchley Matrix for the CDR3 sequence, with each column representing one amino acid within the CDR3 sequence. They transform this matrix using a pre-trained autoencoder model, TESSA [44], to create a 30-dimensional vector for each TCR sequence.

MINN-SA is a multiple-instance learning algorithm because TCR datasets are arranged in repertoires, where each repertoire contains a different amount of TCRs. Within the repertoire, we will have a label to indicate whether the repertoire comes from a cancer or a healthy patient rather than whether each TCR targets cancer antigens. This dataset structure is due to the practical impossibility of labelling the specificity of individual TCRs as TCRs are known for their cross-reactivity, which states that TCRs can potentially recognise more than one antigen [45].

MINN-SA tackled this multiple-instance learning problem by using a weighted sum across all TCRs within the repertoire to create a repertoire representing vector, also known as the bag representing vector. The weights for this sum are computed using the Sparsemax activation function [46], which is a variant of softmax; instead of having small probabilities for irrelevant instances, sparsemax assigns 0. Algorithm 4.2.1 is a pseudocode for the sparsemax algorithm.

The benefit of using sparsemax over softmax has been fully demonstrated within the weighted sum. Softmax cannot output a zero probability, where this non-zero probability for irrelevant TCRs will obscure the signal in the weighted sum process. On the other hand, since Sparsemax can output a zero probability, it amplifies the signal for relevant TCRs within the bag-representing vector, as irrelevant TCRs will not be involved within the sum. Hence, all vectors in the weighted sum will be related to the classification label.

Using sparsemax increases the sparsity of the probability distribution generated, increases the computation speed, and improves explainability. This is due to sparsemax's ability to distinguish the instances that matter and those that do not. In a trained model, it is expected that TCRs related to cancer will be assigned a non-zero probability, whereas the irrelevant instances will be assigned a zero probability. This increases the interpretability of the model, as we can now see which instances are deemed important by the model. Therefore, we can explain what the model has learnt. We can also track if the model's knowledge aligns with what we know about the problem.

In MINN-SA, TESSA processes the Atchley matrices for each CDR3 sequence into 30-dimensional vectors, which are then passed into the deep network to obtain the components within the weighted sum. The processed vectors and weights from sparsemax will be used to compute a weighted sum. The result from the sum creates a bag-representing vector, which is then classified by a classifying network to form a probability of whether the patient has cancer.

MINN-SA gave 0.744 and 0.818 in AUC as the median of 10-fold cross-validation on ten types of cancer in balanced and imbalanced data scenarios, respectively. This AUC performance might originate from the authors' lack of data in this study. Therefore, MINN-SA is prone to overfitting, as the deep network that creates the high-level feature vector contains much more trainable parameters than the number of labels.

Although it is not convincing that MINN-SA has generalised the relationship between cancer and non-cancer patients by evaluating the AUC, many features within MINN-SA stand out. We believe using sparsemax and the weighted sum is an important aspect that could be learnt from MINN-SA.

3.4 Transfer Learning

To highlight the importance of using transfer learning, let's consider a newborn, 'Timmy', a destined radiologist. For Timmy to progress in his destined career, he must learn Biology, earn a medical degree, and pass his residency to progress in his destined career. If we ignore licensing, it is not impossible for Timmy to be a radiologist by teaching him the sorts of MRIs that correspond to cancer since birth without first teaching him biology or studying medicine in university. If we assume that eventually, Timmy can distinguish the MRIs that are from a cancer patient and those that are not, it is not convincing that he has learnt enough to become a radiologist. It is probably true that he might not understand why one medical image correlates with a symptom.

This underpins a critical problem with all models seen in the previous section. Models such as DeepCAT [15] have used randomly initialised models with no knowledge of TCRs and are directly trained to predict cancer using TCR sequences. Although these models have demonstrated good AUC results, this can be improved by taking a more gradual approach to training models. This can be done by providing sufficient background knowledge of the problem to the model before training the model in the actual knowledge we would wish the model to learn.

Furthermore, these works have been seen to use physico-chemical properties such as the Atchley Factors [25] in two different articles: [23, 24], and the AAIndex [43] in [15] to represent TCRs numerically. This is arguably not an effective way for a model to learn about the structure of the TCR, as physico-chemical properties cannot capture TCR-specific information and neighbouring amino acids within the same CDR3 sequence.

3.4.1 Domain Mismatch

Although MINN-SA used TESSA [47], an encoder specialised in encoding Atchley matrices from CDR3 amino acid sequences onto a 30-dimensional vector, we argue that the use of TESSA is an instance of domain mismatch. TESSA is a convolutional neural network that processes CDR3 sequences like images. This design overlooks the language-like structure of amino acid sequences, which exhibit language-like patterns instead of visual images.

This mismatch becomes particularly problematic in tumour environments. The authors highlighted a critical issue where the predictive power of TESSA diminishes due to the homogenisation of T cell functional patterns in cancerous contexts. This renders TESSA less effective at distinguishing tumour-targeting TCRs within its encoding space. The underlying cause may have stemmed from TESSA's failure to account for TCR sequences' sequential, language-like structure, which is critical for accurately interpreting TCR functional implications in complex biological settings like tumours.

Hence, we argue that using TESSA to detect cancer is a domain mismatch. Domain mismatch occurs when we apply a model to a domain in which the model has not been trained or is weak. We have seen that TESSA is weak in tumour environments, such as cancer. Hence, applying TESSA to identify whether a patient has cancer in MINN-SA demonstrates this fundamental concept in transfer learning.

3.4.2 Pre-Trained Models

To address problems such as domain mismatch and insufficient background knowledge of TCRs and cancer, we propose using pre-trained large language models (LLMs). LLMs can understand language-like properties of sequences and can offer insight into the complex relationships within TCRs. Leveraging LLM's ability to understand intrinsic language-like properties of amino acids, a more thorough understanding of problems to do with TCR sequences can be acquired. Using LLMs to model TCRs can also avoid problems such as the inability to generalise in particular scenarios like cancer in TESSA.

Pre-trained models (PTM) such as ChatGPT [48] are models already trained with data from a broad domain. PTMs will often have a broad understanding, but they are not specialised towards solving one particular task. Whilst they can be used on specific tasks, they benefit from fine-tuning, which uses data from a smaller domain and specialises a PTM. An example of this specialisation process is fine-tuning a large language model like ChatGPT to classify whether an email is spam.

It has been seen that correct training paradigms will make fine-tuned models perform better than PTMs [49, 50, 51]. It has also been demonstrated that large-scale fine-tuned PTMs often perform better than models that are trained from scratch [52, 53, 54].

In a medical setting, LLaMA [55], a large language model, has been fine-tuned to create ChatDoctor [56]. It has been shown that ChatDoctor's understanding of patient inquiries have significantly improved, and provides more accurate consultations than the pre-trained LLaMA. The authors have also compared ChatDoctor to ChatGPT, where ChatGPT is a stronger PTM than LLaMA in most aspects [57]. After fine-tuning, ChatDoctor yields a better BERTScore than ChatGPT in responding to patient queries, underpinning the importance of fine-tuning.

Since we have seen that TCRs exhibit a grammatical structure in section 3.1, we can apply language models pre-trained on TCRs to generate an encoding for them. This approach can be promising as LLMs can create vectorised representations of the input based on the ordering of tokens and their surrounding tokens, including ones positioned further away. This provides a solid promise to our proposal of using pre-trained TCR language models in cancer prediction, as they can encode tokens with information based on the surrounding amino acids, which is what physico-chemical encodings cannot.

3.4.3 TCR-BERT

TCR-BERT is one of the earliest TCR pre-trained LLMs, which is then fine-tuned to solve TCR sequencing problems [32]. TCR-BERT is a lightly modified BERT model pre-trained on unlabelled TCR sequences. The model has approximately 57 million parameters, where the modifications onto the conventional BERT model [58] allowed TCR-BERT to leverage unlabelled TCR sequences effectively so it can learn from the vast diversity and complex binding dynamics of TCRs to antigens.

TCR-BERT had been pre-trained using a two-step process to provide the model with a more gradual process of understanding TCR problems. The two-step process is as below, and weights for both models are available on HuggingFace with links provided:

1. **Step 1: Masked Amino Acid Prediction.**
   `https://huggingface.co/wukevin/tcr-bert-mlm-only`
   Upon initialisation, TCR-BERT is trained on a large dataset of unlabelled TCR CDR3 sequences. The training involved a masked language modelling process where 15% of amino acids in each sequence is randomly masked and TCR-BERT is trained to predict these hidden

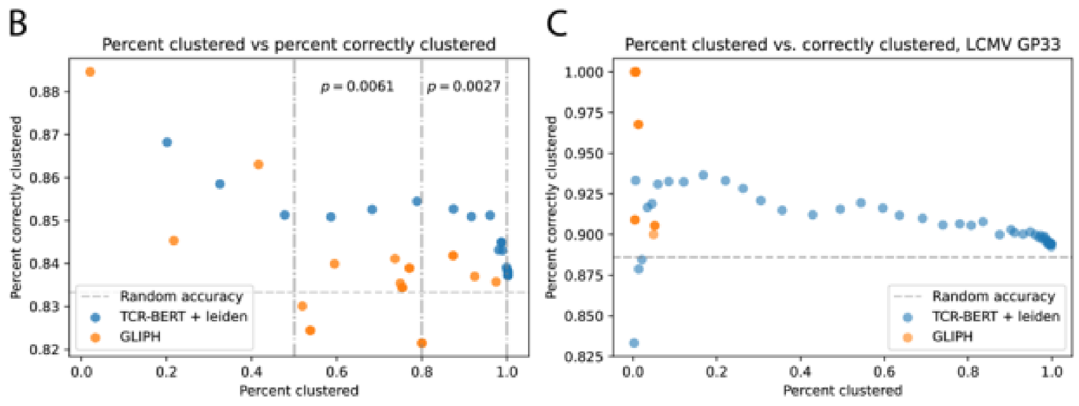


Figure 3.2: TCR-BERT compared against GLIPH [32]

amino acids based on the unmasked 85%. This allowed the model to learn the underlying semantics of naturally occurring TCR sequences.

The dataset used for this step consists of 88,403 predominantly human TCR sequences from α and β chains only, collected from the public datasets: VDJdb [59] and PIRD [60]. This covers a wide range of known and unknown antigen specificity and phenotypes, such as the HLA alleles.

2. **Step 2: Antigen Classification.**
`https://huggingface.co/wukevin/tcr-bert`
Taking the model from the previous step, TCR-BERT is further trained on 4,365 β chains' CDR3 amino acid sequences to predict the antigens they bind to.

It is promising to say that the model understands the underlying problem it is tackling with, such as antigen prediction, as TCR-BERT is trained through a gradual process. The authors verified this hypothesis by pairing TCR-BERT with downstream machine learning algorithms to tackle supervised and unsupervised learning problems.

In the supervised learning scenario, the authors compared TCR-BERT's ability to predict antigen binding against other state-of-the-art algorithms when paired with a Convolutional Neural Network (CNN) and Evolutionary Scale Modelling (ESM) [61]. Under 26 different training instances where various amounts of antigens are used to train the paired model, TCR-BERT showed an improvement in 25 out of 26 instances when paired with a CNN and showed an improvement in 26 out of 26 instances when paired with a downstream ESM when compared against other state-of-the-art algorithms.

TCR-BERT has also shown a significant performance improvement compared to other algorithms in predicting the TCR beta chain that binds to the human NP177 antigen. TCR-BERT had an AUC of 0.338 whereas the second best performing AUC is 0.299, achieved by TAPE [62]. TCR-BERT has also improved performance in predicting antigen binding when given paired TCR CDR3 alpha and beta sequence, showing an AUC of 0.608, where the second best performing AUC is 0.541, achieved by a baseline CNN.

TCR-BERT was also evaluated on unsupervised learning tasks. It was tasked to cluster patient

TCR sequences upon pairing with the Leiden algorithm [63]. As a comparison, GLIPH was used, which is a specially designed clustering algorithm for TCRs [33].

TCR-BERT demonstrated a more consistent performance in correctly clustering TCR sequences across all percentage proteins clustered, as seen in Figure 3.2. GLIPH was only able to cluster a maximum of 14% of TCRs, whereas TCR-BERT can cluster all TCRs, with an accuracy of slightly lower than 90% as shown in graph C in Figure 3.2. This demonstrates that TCR-BERT can transform the data so that downstream algorithms can better capture patterns within the data.

With its robustness and performance across both supervised and unsupervised learning tasks, TCR-BERT promises that applying transfer learning to itself to predict cancer by pairing it with downstream algorithms can give better results than other state-of-the-art algorithms.

Chapter 4

Methodology & Results

Having established the promise that transfer learning can assist models in understanding a specialised domain of expertise by first learning a broader domain, we aim to utilise pre-trained TCR language models to predict cancer by pairing them with a downstream classifier.

This chapter demonstrates how the model architecture, training algorithm and training paradigm are designed. In particular, we focus on two types of encodings: 'symbolic' and 'subsymbolic' encoding. The unconventional usage of these two phrases is inspired by 'symbolic AI' and 'subsymbolic AI'. These phrases are created due to a lack of terminology that best distinguishes the two representation spaces to the best of our knowledge. We define them as follows:

Definition 1 (Symbolic Encoding) *Symbolic encoding is a method that represents nonnumerical inputs by assigning meaningful values to each symbol. These values have an understandable meaning towards it.*

Definition 2 (Subsymbolic Encoding) *Subsymbolic encoding is a method that assigns each symbol within a nonnumerical input with values that do not carry a human-understandable meaning. This method could be through a pre-trained model (PTM).*

In this chapter, we aim to provide a layout of our experiment. Our experimental hypothesis is that transfer learning, which refers to using subsymbolic encodings from a TCR PTM, is a more effective means of training downstream classifiers to classify cancer than symbolic encodings.

We will use four symbolic encodings to provide evidence for our hypothesis and compare their performances with two subsymbolic encodings. The two subsymbolic encodings are from TCR-BERT and an in-house pre-trained model SCEPTR, which will be introduced in section 4.1.3.

An overview of the pipeline has been attached graphically as Figure 4.1 for convenience. Note that when we use different encodings, we will only change the code that performs the action 'Numerically Representing Sequences' within Figure 4.1, and subsequently, the number of neurons in the neural network.

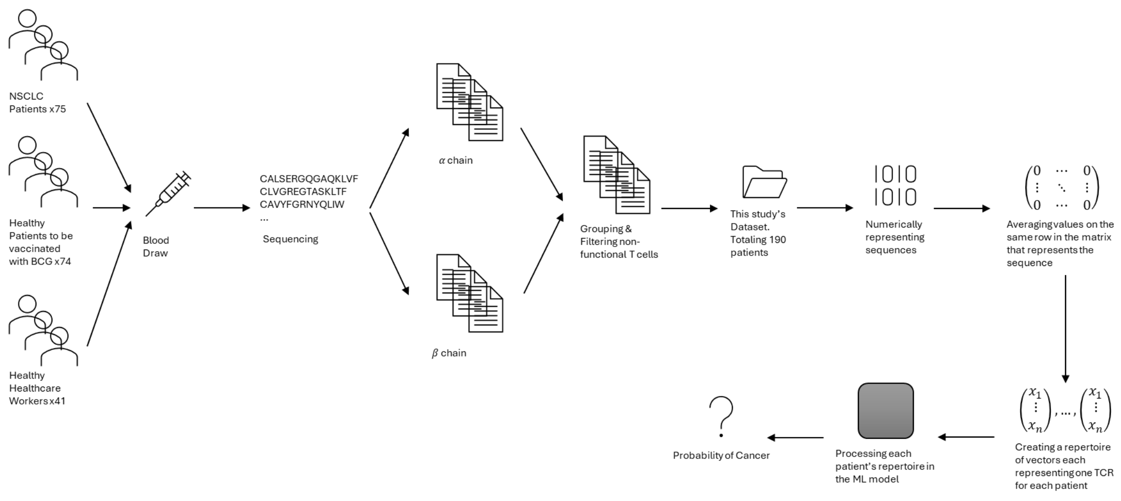

NSCLC Patients x75
Healthy Patients to be vaccinated with BCG x74
Healthy Healthcare Workers x41
Blood Draw
CALSERGQGAQKLVF
CLVGREGTASKLTF
CAVYFGRNYQLIW
...
Sequencing
α chain
β chain
Grouping & Filtering non-functional T cells
This study's Dataset. Totaling 190 patients
1010
1010
Numerically representing sequences
Averaging values on the same row in the matrix that represents the sequence
Creating a repertoire of vectors each representing one TCR for each patient
Processing each patient's repertoire in the ML model
?
Probability of Cancer

Figure 4.1: Data Pipeline

4.1 Data Processing

This section describes the data that we are using. We will introduce the training data, how we clean it, and how to represent them numerically. We will also briefly discuss the in-house model SCEPTR, which is a yet-to-be-published model developed within the Chain's Lab. Note that we do not claim the originality of SCEPTR.

4.1.1 Training data

We use three different datasets to train our model. These three datasets are stored within the Chain's Lab Research Data Storage and are not publicly available. TCRs are sequenced using the same sequencing methodology in the three datasets to avoid a sampling bias in this study. We suggest referring to the articles for each dataset to learn more about the sampling methodology.

The dataset containing all cancer patients will be the Lung Tracking Cancer Evolution through therapy (Rx) dataset, also known as the Lung TRACERx, referred to as Tx [39, 64]. Data sampling has been conducted in hospitals in London, Leicester, Manchester, Aberdeen, Birmingham and Cardiff with 842 patients primarily diagnosed with Non-Small Cell Lung Cancer (NSCLC) over an accrual period of 4 years [64]. We focus on early-stage lung cancer patients to increase the practicability, as the model would be best prepared to classify whether an asymptotic patient has lung cancer.

The control data is extracted from two different studies. One of the datasets was collected to study healthcare workers during the first wave of the COVID-19 epidemic [65], and another dataset was collected to investigate the effect of *Bacillus Calmette Guerin* (BCG) vaccination which is to provide human protection against *M. tuberculosis* [66].

To maintain fairness, in that the control data is not obscured with T cell receptors with a BCG or a COVID-19 specificity, we will only use data from individuals not infected with COVID-19 or have not received the BCG vaccine. All experimental subjects within the control set have been self-declared as healthy.

Whilst with appreciation that using TCR repertoires sampled from tumours for the positive set is going to increase the accuracy as it is easier for the model to distinguish between the two classes, we use only peripheral blood mononuclear cell (PBMC) data for both classes to avoid introducing a bias to the model. Furthermore, it is unethical to invasively sample TCR repertoires from a healthy patient's lung since PBMC data can be sampled from a blood draw. Therefore, using PBMC data for both classes is ethical and unbiased. Note that when we use the phrase 'PBMC', we refer to only T cells whilst appreciating that PBMC includes more cells than T cells.

This study was conducted without ethical approval, as all files used do not include personal identifiers for the patients involved. All files contain only the sequences of TCRs in the blood draw, which cannot be reverse-engineered to derive the patient's identity.

4.1.2 Data Cleaning

We clean the data by removing all non-functional TCR sequences using TidyTCells (TT) [67], a data-cleaning library for TCR repertoires. We use TT to find non-functional TCRs for removal and standardise TCR annotations to be IGMT-compliant. To ensure that SCEPTR and TCR-BERT, the two pre-trained models, perform properly as both are trained on alpha and beta chains, we also remove TCRs that host a gamma or delta chain.

After cleaning, we are left with 230 files for the control set and 149 for the cancer set, totalling 379. The amount of T cell receptors within each file is not fixed; each file contains TCR sequences for either the alpha or beta chain for a particular patient, so there are two files from the same patient, one for their alpha chain TCRs and another for their beta chain TCRs. It should be noted that these alpha and beta chains are not paired.

To help the model better analyse the patient's repertoire, we concatenate the alpha chain file with the beta chain file for the same patient. This is because beta chains are thought to contain more information regarding the epitope specificity of a TCR and are more diverse due to its VDJ recombination process [68]. Therefore, if we concatenate the two files, we allow the model to have a more thorough view of the patient's bodily conditions and their repertoire of TCRs, thereby making a well-rounded decision as to whether the patient has cancer.

After concatenating, we are left with 115 files for the control set and 75 files for the cancer set, each corresponding to the whole TCR repertoire from the same patient. Note that one of the pre-trained models, SCEPTR, can process V call and CDR3 sequences; therefore, we include the V call genes in the files used to train SCEPTR's downstream classifier.

4.1.3 SCEPTR

When this article was written, SCEPTR was an in-house, yet-to-be-published model within the Chain Labs[1]. Similarly to TCR-BERT, SCEPTR is a pre-trained BERT-based language model on TCRs but has 153 thousand parameters as opposed to TCR-BERT's 57 million.

SCEPTR has been pre-trained with unlabelled paired chain TCRs from the dataset provided in [69]. This enables SCEPTR to process paired or unpaired alpha and beta chains with V call and CDR3 sequences to create an embedding. Since SCEPTR is not trained with the data used in this study, this avoids data leakage to the downstream model. Using this dataset, SCEPTR was first

[1]This article does not claim originality in SCEPTR.

trained on masked amino acid modelling and subsequently contrastive modelling to assign spatial locality for similar input data. It is trained to encode TCRs onto a 64-dimensional hyperspace.

4.1.4 Symbolic Encodings

Three physico-chemical encodings alongside 1 ‘control’ encoding will be used, collectively called symbolic encodings. All four symbolic encodings are tab-separated value (TSV) files with several numbers per amino acid indicating the amino acid’s physico-chemical properties. We refer to each of these values for an amino acid as a feature.

For the physico-chemical encodings, we use the Atchley Factors [25], Kidera Factors [26] and Amino Acid Properties [27]. They represent one amino acid with 5, 10 and 14 features, respectively. We extracted the encodings from the following GitHub repository:
`https://github.com/vadimnazarov/kidera-atchley`.

We scale the values linearly for each feature using the minimum-maximum scaling technique defined below to reduce the difficulty for the model to learn due to the values’ range. After scaling, all values are between 0 and 1.

$$x_{i,\text{scaled}} = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}$$

We also use a random encoding as a ‘control’ encoding. We use five features pre-generated from a random uniform distribution from 0 to 1 for each amino acid. We use five features in this encoding since the minimum number of features per amino acid in the physico-chemical encodings is 5.

If we assume that physico-chemical encodings provide a good and learnable embedding space, we would assume that all three physico-chemical encodings will outperform the random encoding, as the random encoding’s embedding space should be challenging to learn since it is random. On the contrary, if any of the three physico-chemical encodings have a similar or worse performance than the random encoding, then this shows that the particular physico-chemical encoding is not a practical representation space.

We create a vector representation for each TCR CDR3 sequence from each symbolic encoding by first generating an embedding for each amino acid in the TCR sequence. Then, we average the list of embedding vectors, creating one vector representing the whole TCR CDR3 sequence. We take an average across the embeddings’ representation to maintain fairness since we also take a feature-wise average from TCR-BERT’s subsymbolic encoding, which will be covered in the next section.

4.1.5 Subsymbolic Encoding

In this section, we will introduce large language models (LLMs) as we have seen that our two symbolic encodings, TCR-BERT and SCEPTR in section 3.4.3 and 4.1.3 respectively, are both LLMs. We then discuss how and where we will be extracting our embedding from.

Although LLMs such as ChatGPT [48] seem to be taking in strings as input, the neural network itself does not take in a string. It takes in a vectorised expression of the string instead. Typically, a string is passed into a tokenizer, which parses the string into a series of tokens, which can be words, multiple words or sections of words [70].

In bioinformatics, it is often true that one amino acid is tokenized into one token, which can then be parsed into a one-hot vector where the non-zero value within this one-hot vector indicates the amino acid. This is no exception for SCEPTR and TCR-BERT.

The token will then be passed into a transformer-based structure, such as BERT [58], to turn the tokens into an embedding. This embedding refers to a vector inside a high-dimensional vector space. This transformer aims to assign spacial locality to similar inputs [70]. These embeddings are then passed to the neural network for processing, where this processing could be a classification task such as masked language modelling.

In conventional natural language LLMs, it has been observed that it can be complicated to teach one pre-trained language model to understand another language [71, 72, 73]. Yet, through active forgetting, a technique that involves actively resetting the output embedding layers and retraining the whole neural network, it has been shown that PTMs give a better performance than adding extra embedding layers at the end of the network [74] or training the LLM for some extra epochs on the target language without amendments to the neural network configuration.

This demonstrates that through a good quality of pretraining, models can capture adequate information about the broader domain in upstream layers, increasing its plasticity and robustness across similar domains [75, 76, 77, 78].

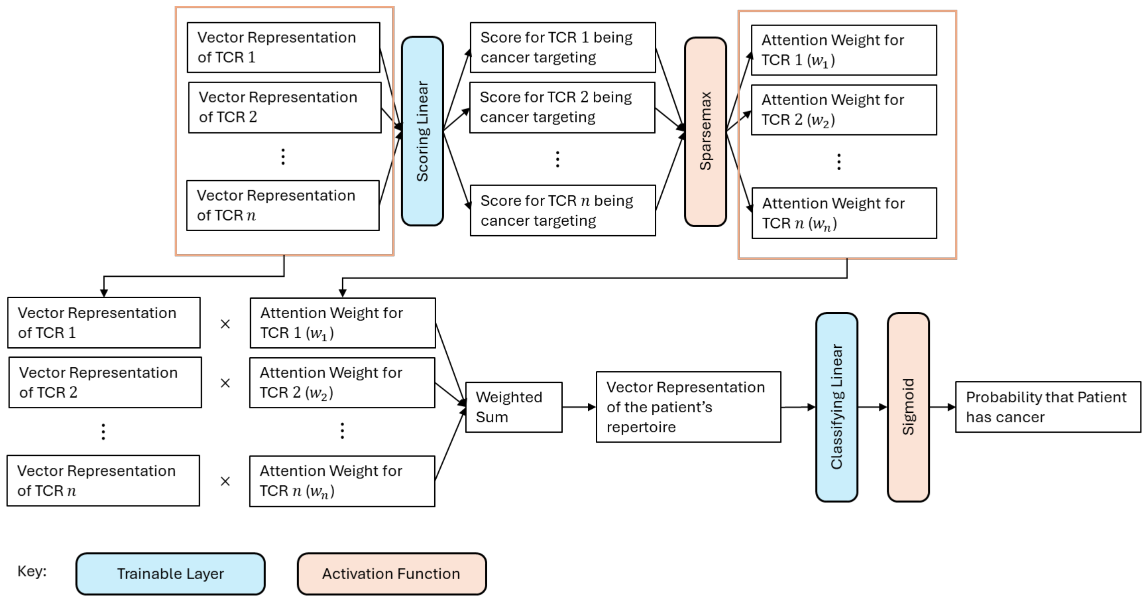
There is much research that supports this claim, where it has been suggested that earlier layers within neural networks are not only applicable for one dataset but seem to be transferable across problems within a similar domain as it is learning a low level of information [79]. This is particularly evident within Convolutional Neural Networks [80, 81]. A similar observation is expected within Large Language Models [82, 83], but is challenging to be solidly proven within Large Language Models, as they are one of the least interpretable classes of deep learning models [84].

On the subject of cancer prediction using TCRs, we are looking for an output embedding that can be beneficial for our prediction task - having established that earlier layers of PTMs contain a broad set of information, which could, therefore, help downstream layers to learn better as it assigns a meaningful geometric hyperspace to the input, we will look into using earlier layers, or the layers that are to deal with processing the language of TCRs to pair with our downstream algorithms.

We will be using two different LLMs to generate output embeddings for our task: the first is SCEPTR, as previously introduced in section 4.1.3. SCEPTR is a pre-trained model, which has been trained on data provided in [69], and therefore is an independent dataset to Tx. SCEPTR has been trained to generate a 64-dimensional output embedding based on a TCR's V call and CDR3 sequence. SCEPTR comes with a package that computes the vector representation of TCRs using the pre-trained model. The vector representation of each TCR is calculated using the `<cls>` pooling method, where the output embedding is the embedding of the `<cls>` token.

The second model we use is TCR-BERT, reviewed previously in section 3.4.3. We will use the masked amino acid modelling pre-trained variant of TCR-BERT, which can process alpha and beta chain CDR3 sequences. This can provide more information about the patient's TCR repertoire to downstream models than the other variant that can only process beta chains. TCR-BERT is trained on VDJdb, independent of Tx and the two control datasets.

TCR-BERT has 12 blocks of BERT [32], outputting a 768-dimensional feature space. Since the



Vector Representation of TCR 1
Vector Representation of TCR 2
Vector Representation of TCR n
Scoring Linear
Score for TCR 1 being cancer targeting
Score for TCR 2 being cancer targeting
Score for TCR n being cancer targeting
Sparsemax
Attention Weight for TCR 1 (w_1)
Attention Weight for TCR 2 (w_2)
Attention Weight for TCR n (w_n)
Weighted Sum
Vector Representation of the patient's repertoire
Classifying Linear
Sigmoid
Probability that Patient has cancer
Key:
Trainable Layer
Activation Function

Figure 4.2: Downstream Model for Cancer Classification

BERT structure is highly efficient in language modelling, we will extract only the BERT structures within the network to pair with our downstream algorithm.

Conventionally, we should extract the embedding of the `<cls>` token to extract the embedding from a BERT structure, similarly to SCEPTR. However, TCR-BERT is a modified BERT model that tackles classification rather than protein generation, so TCR-BERT does not output a stop token. Instead, each amino acid sequence is represented in a 768 by n matrix, where n is the number of amino acids in the TCR CDR3 sequence. We create a vector representation of each TCR by averaging row-wise in this 768 by n matrix to obtain a 768-dimensional vector representing the CDR3 sequence.

During our training, we will freeze the parameters in both large language models, therefore making them untrainable.

4.2 Model Design

Since we hypothesise that subsymbolic encodings are more efficient than symbolic encodings, we will use a simple model with minimal parameters and demonstrate that it can infer better when it takes in subsymbolic encodings as opposed to symbolic encodings, even using a simple model and without fine-tuning the upstream model during training.

In this section, we first describe the downstream model for the two methods of subsymbolic encodings and argue that using this same model architecture for the symbolic encodings is a fair means of comparison to demonstrate that symbolic encodings are not as expressive. An illustration for the model has been placed as Figure 4.2.

4.2.1 Subsymbolic Encoding Downstream Model

The problem of classifying whether a patient has cancer using TCRs is a Multi-Instance Learning problem. Within each TCR repertoire (often referred to as ‘a bag’ in the context of Multi-Instance Learning), we have a collection of TCRs from the patient and a label for whether this patient has cancer. However, we do not have a label for whether each TCR has a cancer specificity.

This further motivates the importance of a simple model. During forward propagation, the input to each layer is stored to compute the gradients in the backpropagation step. These inputs are stored in GPU memory and are not deleted until backpropagation.

In the context of using TCRs to predict cancer, since we do not have a fixed amount of TCRs in each TCR repertoire, the number of TCRs per repertoire could be large. This means forward propagating a whole TCR repertoire down a deep model could incur many of these inputs to be stored within the network. Since this accumulation can only be cleared out by taking one update step, this leads to an explosion in GPU memory usage, causing complications when training the model due to computational constraints.

Creating a model for this task is difficult as there may only be a few TCR instances in a cancer bag specific to cancer. Moreover, the number of TCRs in a bag is not a determined value, and it is impossible to sort them meaningfully.

This highlights the importance of using a weighted sum, which is what MINN-SA [23] used for their work. It will be desirable if the weights add up to 1, as this avoids creating a bag-representing vector with a significantly smaller or larger magnitude than the output embeddings. Similarly to MINN-SA, we rely on a weighted sum generated by a sparsemax activation layer [46]. The sparsemax algorithm is defined as Algorithm 4.2.1, where τ is defined to be a threshold function and all values smaller than $\tau(\cdot)$ will be assigned value 0. The sparsemax Function is non-smooth but is differentiable except at a few points. Details can be seen in [46].

Algorithm 1 Sparsemax Algorithm [46]

```
1: Input: Vector $z \in \mathbb{R}^n$
2: Define: $z^{(i)}$ as the $i$th value in Vector $z$
3: Sort $z$ as $z^{(1)} \geq z^{(2)} \geq \cdots \geq z^{(n)}$
4: Find $k(z) := \max \left\{ k \in [K] \,\middle|\, 1 + kz^{(k)} > \sum_{j \leq k} z^{(j)} \right\}$
5: Define $\tau(z) = \left( \sum_{j \leq k(z)} z^{(j)} - 1 \right) / k(z)$
6: Output: probability vector $p$ such that $p_i = max(z_i - \tau(z), 0)$
```

As opposed to a softmax activation function, which is defined as:

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_{k=0}^{n} \exp(x_k)}$$

We use a sparsemax as it outputs a sparse probability distribution. A sparse probability distribution is more desirable than a softmax since we know that only a small amount of x_ks correspond to cancer, where we define x_i to be a TCR instance and e_i to be its embedding.

If a softmax function generates the weights, all e_i will be assigned some non-zero weighting regardless of how small it is, which obscures the signal in the weighted sum and, therefore, makes it hard for the downstream classifier to learn a relationship.

To use the sparsemax function, we must assign a scalar score to each input vector. To make sure that our function is simple, we implement a linear function, where the score, for instance i will be computed as:

$$s_i = \langle w_s, e_i \rangle + b_s$$

where we define w_s as the scoring weights and b_s as the scoring bias. This layer is named the scoring layer, and this function assumes that all TCRs with the same specificity will point strongly to direction w_s^*, which we call the optimal direction.

In the discussions section, we will evaluate the cosine similarity of w_s^*s from different training instances of the same model architecture to evaluate whether the upstream model can put TCRs with the same specificity in a close distance despite not knowing this information from training.

After we obtain s_i, we can obtain p_i, the weighting outputted by the sparsemax function, for instance i. We can think of p_i as the probability that x_i is a TCR with a cancer specificity. Subsequently, we can compute the bag-representing vector $b_X = \sum p_i e_i$. This is a weighted sum of the embedding vectors e_i with p_i being the weights. This can then be classified with a logistic regression model $f(b_X) = \sigma(\langle w_c, b_X \rangle + b_c)$ to compute the probability of cancer where σ is the sigmoid activation function.

The model can be visually illustrated as Figure 4.2. In particular, we assert that the model that generates the output embedding will have all parameters frozen and, therefore, will not update during gradient descent. We believe that doing so will demonstrate the robustness of the subsymbolic encoding.

4.2.2 Symbolic Encoding Downstream Model

To design a symbolic encoding downstream model that adequately compares the difference between the quality of the embedding space for each encoding method, we must not introduce further information into the encoded vector through kernels nor increase the difficulty of the learning algorithm by training an unnecessary number of parameters.

We also believe that using an autoencoder like TESSA [44] is not a good idea as TESSA discards the linguistic structure of the input and is weak at embedding TCRs from a tumour environment, such as a cancer patient.

Hence, we will use the raw representation space to embed the input. After numerically representing each amino acid as a vector, we average the vectors to obtain one vector representing the TCR CDR3 sequence for fairness since we do the same operation for TCR-BERT. Note that all values within the vector will be in the range of 0 to 1 to ensure the input range does not affect the difficulty of the learning algorithm.

To avoid increasing the difficulty of the learning algorithm whilst ensuring that we evaluate subsymbolic and symbolic encodings fairly, we believe that the symbolic encoding's downstream model should share the same model architecture as the subsymbolic encoding's downstream model. This ensures that the difference between the performance of the models trained from the two encodings originates in the expressivity of the encoding methods.

As the amount of features within symbolic encodings is significantly lower than subsymbolic encodings, we create table 4.1 to demonstrate the trainable parameters in our model for each encoding

Encoding Method	Encoding Dimension	Trainable Parameters
Atchley & Random	5	12
Kidera	10	22
Amino Acid Properties	14	30
TCR-BERT	768	1538
SCEPTR	64	130

Table 4.1: Number of Parameters in downstream model

means. Note that there are two trainable linear layers that both output a scalar; hence, the number of trainable parameters can be modelled as $2(n + 1)$.

4.3 Experimental Layout

We provide background information to our experiment here, such as hyperparameters and the training environment.

4.3.1 Training Environment

The code has been deployed in Python 3.11 using PyTorch 2.2.0 with CUDA 12.1 acceleration. Versions for other libraries can be found in the GitHub repository where the code has been deposited. The training has been completed on PCs with i9-12900K processors, 128 GB RAM and NVIDIA GeForce RTX 3090 Ti GPUs, which have 24GB GRAM.

Due to resource limitations, it is impossible to correctly report the time taken for each training process as if they were run on a computer without background tasks. This is because all PCs used in this study are shared across students, meaning that someone could be running an intense background task whilst we are running our training. This makes the recorded run time inaccurate.

4.3.2 Hyperparameters

Since TCR-BERT has a significantly higher output embedding dimension than the other encodings, we will use L2-Regularisation to reduce the complexity of TCR-BERT's downstream model. This can be imposed by PyTorch's optimizer's `weight_decay` parameter. We use 0.25 as our L2-Regularisation strength. Note that models trained for all other encodings do not use L2-regularisation.

However, we are still expecting TCR-BERT's downstream model to overfit. This is because TCR-BERT's downstream model has 1538 parameters, whilst we have only 190 labels to train it. This makes the training problem for TCR-BERT's downstream model over-determined. Whilst it is possible to downscale TCR-BERT's feature space through the use of an auto-encoder or the Johnson-Lindenstrauss Lemma, we are not doing so as we are looking to compare the raw embedding space as discussed in section 4.2.2 to maintain fairness.

To explore the best model for each encoding, we train the model for 50 epochs, an extensive amount for 190 bags. We deem this amount of epochs comprehensive as we have observed plateauing in training loss for most training instances across all embedding means except for TCR-BERT, where it overfitted.

We will then select the best model from the checkpoints generated from the 50 epochs. The best model is selected based on the highest test set AUC in the training instance. The test loss at that epoch should not increase compared to the previous few epochs, as this is a sign of overfitting.

The learning rate is 0.001 for all six embedding methods on the Adam optimiser [85]. This relatively small learning rate should allow the model to take enough steps throughout the 50 epochs to converge. Other hyperparameters in the Adam optimiser are as PyTorch's default.

One limitation is that we are unable to perform a Grid Search to find the best set of hyperparameters in this task as the training takes at minimum 36 hours for each encoding when we use an 80%-20% train and test data split, rendering it not feasible to perform such an extensive search. The problem with the time taken is more severe in training TCR-BERT's downstream model, where it took at least 72 hours to complete the 50 training epochs.

To mitigate problems with class imbalance, we upscale the calculated Binary cross-entropy loss. For example, if 25% of our data is in the positive class and the remaining is in the negative class, we scale the Binary Cross Entropy loss up by $\frac{1}{0.25} = 4$ times if we are computing the loss for the positive class and $\frac{1}{0.75} = \frac{4}{3}$ otherwise. This is done to encourage the model to descent more on the minority class than the majority class, so the model will not classifying everything to the majority class. Note that the losses that we will present are not upscaled.

4.4 Results

Encoding	Train BCE Loss	Train Accuracy	Train AUC
Atchley	0.686 - 0.696 (μ: 0.691)	0.395 - 0.750 (μ: 0.583)	0.223 - 0.877 (μ: 0.663)
Kidera	0.670 - 0.690 (μ: 0.682)	0.664 - 0.836 (μ: 0.732)	0.876 - 0.951 (μ: 0.909)
AA Properties	0.570 - 0.691 (μ: 0.664)	0.566 - 0.849 (μ: 0.716)	0.648 - 0.912 (μ: 0.784)
Random	0.687 - 0.692 (μ: 0.689)	0.612 - 0.836 (μ: 0.716)	0.778 - 0.889 (μ: 0.843)
TCR-BERT	0.105 - 0.353 (μ: 0.221)	0.836 - 0.980 (μ: 0.914)	0.925 - 0.995 (μ: 0.967)
SCEPTR	0.273 - 0.601 (μ: 0.396)	0.824 - 0.949 (μ: 0.886)	0.882 - 0.978 (μ: 0.948)

Table 4.2: Results for the best-performing checkpoint on the train set

Encoding	Test BCE Loss	Test Accuracy	Test AUC
Atchley	0.684 - 0.694 (μ: 0.690)	0.395 - 0.763 (μ: 0.674)	0.157 - 0.939 (μ: 0.728)
Kidera	0.663 - 0.690 (μ: 0.681)	0.632 - 0.921 (μ: 0.763)	0.901 - 0.975 (μ: 0.941)
AA Properties	0.595 - 0.692 (μ: 0.669)	0.526 - 0.842 (μ: 0.711)	0.768 - 0.892 (μ: 0.855)
Random	0.686 - 0.691 (μ: 0.688)	0.605 - 0.842 (μ: 0.753)	0.889 - 0.960 (μ: 0.927)
TCR-BERT	0.366 - 0.579 (μ: 0.497)	0.737 - 0.842 (μ: 0.795)	0.820 - 0.946 (μ: 0.900)
SCEPTR	0.254 - 0.606 (μ: 0.406)	0.771 - 0.971 (μ: 0.877)	0.901 - 1.000 (μ: 0.950)

Table 4.3: Results for the best-performing checkpoint on the test set

Unless otherwise specified, we repeat the same training process five times for each encoding. Across these repeats, we will use different training and testing data splits and model parameter initialization. Note that our results are statistics from the best checkpoint, where the best checkpoints have the best test AUCs across the 50 epochs. We have also manually checked that these chosen checkpoints' test AUC is approximately similar to the train AUC and is not ascending in the test loss in that particular epoch.

Table 4.2 and Table 4.3 summarise the performance of the best checkpoints for each encoding's downstream model's training instance. Details are in Appendix A, where we attached the training and testing loss graphs, AUC graphs and the confusion matrices for each encoding method. Note that we used the threshold of 0.5 to compute the confusion matrix and accuracy statistics.

4.4.1 Evaluation Set

Splitting the data into train-validation-test sets is often considered good practice in machine learning. The training set is used to train the model, and the validation set is used to validate the model during training. The test set is often withheld to observe the different models' performance on some mutually unseen data after training. We refer to this partitioning as train-test-evaluation instead of train-validation-test. This is because we performed a 2-way train-test split to avoid losing too much data before, and the phrase 'test' in a 2-way split has the same functional meaning as 'validation' in a 3-way split in our context.

Since we do not have many repertoires, we only create an evaluation set from our data to evaluate the best-performing encoding to avoid further reducing the knowledge that the models can learn from.

We have observed that SCEPTR's embeddings allow its downstream models to attain a robust test set AUC. Therefore, we create an evaluation set by manually withholding 10% of the data, equivalent to 19 patients' data. We then train SCEPTR's downstream model again, from scratch, with the remaining 90% of the data.

Table 4.2 and 4.3 presents SCEPTR's performance after withholding the data. We repeat the training ten times instead of five, where each training instance splits the remaining 90% of the data using an 80%-20% train-test split.

These 19 patients are composed of 10 control patients, where we extract five patients from each of the two control datasets. The remaining are cancer patients. For convenience, the patients extracted from each dataset have consecutive patient IDs. Since the patient ID and the difficulty of classifying the patient are not correlated, the methodology in choosing repertoires is not based on prior knowledge of its difficulty in classifying. On the contrary, some files withheld to serve as the evaluation data have only CDR3 sequences and do not contain the V calls. This introduces extra difficulty in classification as the TCR encodings are not created from a thorough view of the chain.

We test the downstream models on the evaluation data by taking the best checkpoint from each of the ten repeats. The evaluation set's AUC curves and confusion matrix across the ten trained models have been attached as Figure A.37 and A.38 respectively.

For convenience, the summary of the evaluation set's BCE Loss, accuracy and AUC values for the ten training instances' best checkpoints are summarised as table 4.4.

Encoding	BCE Loss	Accuracy	AUC
SCEPTR	0.215 - 0.545 (μ: 0.326)	0.842 - 1.000 (μ: 0.947)	0.922 - 1.000 (μ: 0.988)

Table 4.4: Results for SCEPTR's downstream model's performance on the evaluation set

Note that data leakage can not happen during training. This is because TCR repertoires are cleaned independently, and the evaluation data has been manually withheld, so it does not exist in the computer that trained the downstream model.

Chapter 5

Discussion

In this chapter, we discuss the robustness of our approach, with a specific focus on SCEPTR as the subsymbolic encoding method. We also evaluate the weaknesses of all symbolic encodings and the limitations of our investigation.

An outline of potential future works for this project will be provided. Future works will be conducted in three directions: increasing the accuracy of cancer prediction using TCRs, extending this investigation between the expressiveness of subsymbolic encoding and symbolic encoding, and increasing the robustness of our study.

5.1 Achievements

Through our experiments, we have demonstrated that symbolic encodings are likely to be a less expressive encoding space than subsymbolic encodings. In this section, we provide three primary evidences for this belief.

The hypothesis that TCR-BERT will overfit can be shown when we compare TCR-BERT's train loss in table 4.2 against its test loss in table 4.3. This is because the train losses in TCR-BERT are significantly lower than the test loss, a sign of overfitting. In section 5.3, we will propose methods to mitigate this issue.

5.1.1 Symbolic Encoding's Lack of Expressivity

We have stated in section 2.2 that physico-chemical encodings are theoretically not as expressive as embeddings from an LLM as LLMs output context-based embeddings, where this embedding has been made with consideration to neighbouring amino acids, which cannot be achieved through encoding with physico-chemical properties.

Three pieces of evidence will be provided to support this claim. They collectively reason why physico-chemical encodings are less expressive than subsymbolic encodings, supporting the above-mentioned claim.

Convergence of Training Loss

We can see the train and test loss curves for the downstream model that encodes TCRs using Atchley factors, Kidera factors, AA Properties and Random Encodings in Figure A.13, A.19, A.25 and A.31 respectively. We notice that only one training instance that used AA Properties as the encoding method showed a sign of learning, demonstrated by its continuous decrease in training loss. This instance of the training resulted in a final train loss of 0.570. All other training instances that encoded TCRs using symbolic means have failed to decrease their training loss throughout epochs and have converged quickly.

We believe that this is not due to a small learning rate, as we can see that some training instances showed a sharp decrease in loss and plateaued quickly. If the problem is with a small learning rate, all training instances should gradually decrease their training loss rather than plateauing.

When we compare this AA Properties' downstream model's train loss with TCR-BERT's or SCEPTR's downstream model's training loss, we notice that the change in loss in AA Properties' downstream model is insignificant. All of SCEPTR's downstream model's training instances have succeeded in decreasing their train loss, and SCEPTR's best downstream model's train loss was 0.276. A similar observation can be made with TCR-BERT in table 4.2.

This demonstrates some easily learnable patterns within SCEPTR's and TCR-BERT's output embedding since most of their downstream models' training instances were able to fit towards it eventually. However, in symbolic encodings, we observe that even if there is a pattern that distinguishes cancer patients and non-cancer patients, this pattern can be complicated for the models to learn and observe.

This problem with train loss is furthered if we consider a classifier which outputs 0.5 all the time. This classifier will have a binary cross-entropy loss of the following:

$$-\frac{1}{n}\sum_{i}^{n} y_i \ln(p_i) + (1-y_i)\ln(1-p_i) = -\frac{1}{n}\sum_{i}^{n} y_i \ln(0.5) + (1-y_i)\ln(0.5) = -\frac{1}{n}\sum_{i}^{n} \ln(0.5) \approx 0.693$$

Note that this expression is invariant to class imbalance since y_i is cancelled. We see that the smallest train BCE loss for the four symbolic encodings is 0.570, and the smallest test BCE loss is 0.595, which is much closer to 0.693 than the train loss of 0.273 and test loss of 0.254 achieved by SCEPTR's downstream model. This suggests that the probabilities outputted from symbolic encodings' downstream models are always close to 0.5, showing that the downstream model is always uncertain about its decision-making.

AUC

Whilst metrics such as the accuracy and confusion matrices are indicative of the model's performance, it requires a selection of a threshold value whereby all values outputted by the model below the threshold value will be assigned a label 0 and label one if the values are above the threshold.

In the plots we showed in Appendix A, all figures relating to the accuracy or inferencing on TCR repertoires use the threshold 0.5 as a comparison. However, the AUC measures the model's performance when we vary this threshold, which mitigates this issue. Therefore, the AUC is more credible when we do not know the optimal threshold.

The test AUCs of the subsymbolic and symbolic encodings are in table 4.3. Ignoring all TCR-BERT's downstream models, which suffered from overfitting, causing a low AUC, we can see that SCEPTR's downstream model can give an extraordinary AUC statistic of at maximum 100% with a mean of 95%. Note that these AUCs are recorded after reserving 10% of the data to serve as the evaluation set, which should increase the difficulty of SCEPTR's downstream model to learn the data properly.

On the contrary, the mean AUCs for the symbolic encodings are 72.8%, 94.1%, 85.5% and 92.7% for Atchley Factors, Kidera Factors, AA Properties, and Random Encodings, respectively. Although Kidera Factors can reach a mean test AUC similar to SCEPTR, it cannot get the perfect test set AUC that SCEPTR can achieve in multiple instances.

Given these arguments, it can be concluded that SCEPTR's embedding space is more expressive than most symbolic encodings.

Difference with Random Encodings

Random encodings are used as the control encoding, where this encoding carries no meaning whatsoever, unlike the physico-chemical encodings. We would therefore expect the random encodings' downstream classifier to perform worse than the physico-chemical encodings' downstream classifier.

However, we observe that all training statistics, such as the loss, accuracy and AUC for random encoding's downstream models, are approximately similar to the downstream model trained with TCRs encoded with physico-chemical encodings in table 4.2 and table 4.3. The random encodings' downstream model occasionally performs better than the physico-chemical properties' downstream model in accuracy and AUC. For example, random encodings' downstream model had higher mean accuracy and AUC than AA Properties' and Atchley Factor's downstream model in both the train and test set.

This provides evidence that random encodings are approximately as expressive as the physico-chemical encodings since the difference in performance of their downstream models is minimal. This also suggests that the models are learning sets of individual shared TCRs, where the embedding method does not matter.

On the other hand, we can see a significant difference between the performance of SCEPTR's and symbolic encodings' downstream models in the test set. This observation cannot be made when we compare the performance of physico-chemical properties and random encodings' downstream models. This demonstrates that subsymbolic encodings are far more expressive than symbolic encodings.

5.1.2 Novelty

In this study, other than demonstrating with novelty that symbolic encodings are not as expressive as subsymbolic encodings, we have also attained far better AUC statistics than state-of-the-art methodologies in cancer prediction by using SCEPTR's encodings.

We have seen in [23] that the current state-of-the-art AUCs for NSCLC is approximately 62.5% and 65% [1] in Lung Adenocarcinoma (LUAD) and Lung Squamous Cell Carcinoma (LUSC), re-

[1]The authors for MINN-SA did not provide the exact numbers, and these numbers are only an estimation from reading Figure 5.1's black line, which indicates MINN-SA's performance.

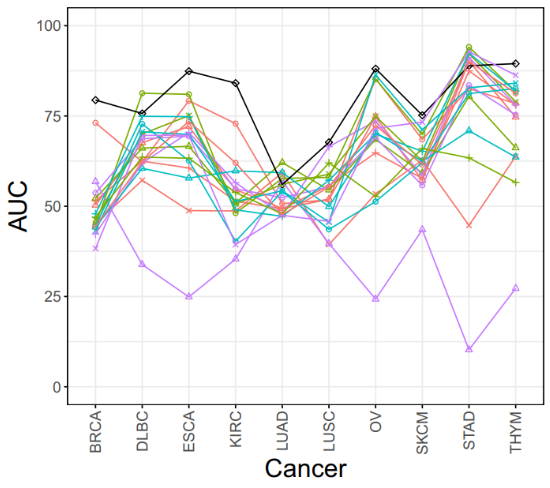


Figure 5.1: AUC Results for 10 Cancer Types on imbalanced dataset [23]

spectively. Note that the AUCs in Figure 5.1 did not include DeepCAT [15] and the Logistic Regression approach [22] as the authors of MINN-SA did not find these articles reproducible [23].

However, even if we were to include DeepCAT and the logistic regression approach's AUCs, whereby they achieved 95% and 91% AUC, respectively, SCEPTR's downstream model's average AUC performance of 98.8% on the evaluation set (Table 4.4) is still the best.

Whilst it can be argued that MINN-SA, DeepCAT and the logistic regression approach used different datasets to train their model, given that our model distinguished whether a patient has cancer from PBMC data, it is unlikely that our data is easier to classify than the data used in any of the three studies as mentioned earlier. This is because some articles, such as MINN-SA, classified tumour tissue TCR repertoires against PBMC TCR repertoires from control patients, which is biased and is an easier task than comparing PBMC data from cancer and healthy patients.

Thus, we believe that using pre-trained TCR language models as the encoding method would be a good direction in researching methodologies for predicting whether a patient is diseased.

5.2 Interpretability

We have seen success over SCEPTR's downstream model, which has demonstrated state-of-the-art AUC statistics as opposed to current models. In this section, we aim to uncover what SCEPTR's downstream model has learned by looking at the probability distribution that the sparsemax layer generated. We will focus only on the best checkpoint out of the ten repeats we have done, which is the 5th repeat, which attained 100% AUC on both the test and evaluation set, demonstrated in Figure A.37. This perfect score indicates that the 5th repeat's model had best learnt the underlying relationship between TCRs and cancer when compared against the other repeats.

We will then look at how ill-posed the problem is by looking at how different the weights for each training instance are against other training instances through the cosine similarity and Euclidean distance.

5.2.1 Predictive T Cell Receptors

A property of the sparsemax layer is that it assigns weights sparsely; therefore, only a small fraction of TCRs will be assigned weights. We can interpret the model's understanding of which TCRs correspond to cancer by interpreting which TCRs have been assigned a non-zero weight. Subsequently, we can gather these TCRs for each correctly classified patient, attempt to understand what the model has learnt, and evaluate whether this knowledge aligns with what we know in biology.

We attempt to understand which V call, J call and CDR3s have been seen repeatedly within different patients in the evaluation set with the best training instance that used SCEPTR's encodings, where this 'best training instance' is hereby referred to as 'the model'. In particular, we are most interested in whether we see the same V call, J call and CDR3 sequence within different true positives.

The model correctly classified 18 out of 19 patients in the evaluation set, whereas it incorrectly classified one control patient. In the nine correctly classified true positives, the model assigned non-zero weights to 32 alpha chains and 292 beta chains TCRs.

Although the evaluation set has more beta chains than alpha chains, we believe it is not the root cause of the model picking up more beta chains for true positives. This is because in table A.4, which records the occurrence of the non-zero weighted TCRs V and J call chains in true negatives, we notice that the model picked up 180 alpha chains and 102 beta chains. Suppose the hypothesis that there are more beta chains in the evaluation set, so the model picks up more beta chains in the true positives, is true. The model should assign more non-zero weights to beta chains than alpha chains in the true negative samples.

Furthermore, there are 464300 alpha chains and 680238 beta chains TCRs in the nine true positives. It can be seen that the ratio of nonzero weighted alpha chains against beta chains is unmatched with the total amount of alpha and beta chains. This demonstrates that the model decides that there are more signals to the positive label in the beta chain than in the alpha chain. This is in line with what we know in biology, as beta chains are thought to carry more information about the specificity of the TCR due to a more diverse set of beta chains that can be generated from the VDJ recombination.

Table A.2 illustrates the unique V and J calls within these non-zero weighted alpha and beta chains and their occurrence frequencies. We can see that TRBV2 is the only beta V call that the model has picked up, and it has also picked up a collection of J calls repeatedly.

We check whether these V and J call genes could be from cancer-targeting TCRs with the use of VDJdb [59]. For simplicity, we only focus on TRBV2 and the most frequently picked up J call gene TRBJ2-2. Note that VDJdb does not incorporate data from our datasets.

Among all cancer-targeting TCRs with a high confidence registered in VDJdb, we observe 135 TCRs have a TRBV2 gene and 294 TCRs have a TRBJ2-2 gene. Of these, 25 TCRs have TRBV2 and TRBJ2-2 as their beta chain V and J call genes.

We gathered the CDR3 sequences seen in multiple patients and placed them in Table A.3. It can be observed that multiple rows have identical CDR3 sequences. This is because the weighting assigned by sparsemax for the same CDR3 in different patients will differ.

In total, SCEPTR's downstream model picked up seven CDR3 chains seen across several patients. Six CDR3 chains are from the beta chain, and one is from the alpha chain. We can see that these six beta chains are all formed from the TRBV2 V call and either TRBJ1-2 or TRBJ2-2 J calls. We attempted to search whether these 7 CDR3 chains are recorded in VDJdb but in vain. Since our dataset and VDJdb are independent, the nonexistence of these CDR3 chains in VDJdb does not show that these CDR3 chains are incorrectly picked up, as CDR3 chains are known to be highly variable.

The CDR3 chain 'CASRGLTGNYGYTF' has been seen in 4 patients. Amongst these four patients, this chain had been expanded in 3. The phrase 'expanded' refers to the scenario where the same chain has been spotted more than once within the same blood draw. This is caused by T cell proliferation during clonal selection. We also observe that the CDR3 chain 'CASSFRGGADEGYTF' is highly expanded within the two patients with this chain in their blood draw. This CDR3 chain has been seen 16 and 32 times within the two patients.

On the contrary, the model picked up more signals from the alpha chain in the true negatives. The model did not give weighting to the same beta chain CDR3s within different patients. The CDR3 sequences that the model picked up are, firstly, alpha chains and, secondly, not as expanded as what we saw in the true positives (Table A.5, A.6). The most expanded CDR3 sequence that the model picked up was an alpha chain with CDR3 sequence 'CAVGGYNKLIF', with eight occurrences within the same patient.

All in all, we can see that even when the model is not informed about the duplicate counts of the TCR during training, the model still picks up expanded TCRs for cancer patients. In the non-zero weighted TCRs in true positives, the model prioritises beta chains, which were thought to carry more signals than alpha chains.

The model assigns non-zero weights to a large selection of TCRs in the true negatives, demonstrating no signal for the model to pick up. These two factors demonstrate robustness with SCEPTR's embedding space and the downstream model.

5.2.2 Component Similarity

To evaluate whether SCEPTR assigns all cancer-targeting TCRs into one close region, we evaluate how similar the scoring layer's weights are against every other repeat of SCEPTR's downstream model. The scoring layer is the layer that assigns a score to each TCR for assigning weights by sparsemax. We also evaluate the similarity of the classifying layer across different training instances.

The intuition for doing so is that if SCEPTR assigns all cancer-targeting TCRs to one close region, then all models that are trained on SCEPTR's embedding space will have the scoring layer pointing in the same direction since there is only one region that has the features that help the model understand the relationship. Hence, we should expect the distance between different models' scoring layers' weights to be close if this is the case.

That said, SCEPTR was never informed about the TCR's specificity during its training; it should not be expected that SCEPTR assigns spatial locality by specificity.

Sparsemax assigns weights by seeing how large the score is. Only the instances close to the scoring layer's weights will be assigned high scores. Therefore, we would assume that the scoring layer is similar to the classifying layer. Using this argument, we would expect the classifying layers' weights to be similar to other training instances.

We quantify this similarity through the cosine similarity and the Euclidean distance. The cosine similarity measures the angle between two different points, whereby -1 indicates that they are pointing in totally different directions, and 1 indicates that they are identical. Cosine similarity is modelled as below:

$$\text{cosine similarity} = \frac{A \cdot B}{||A||\ ||B||}$$

The Euclidean distance measures the distance between two different points, and the Euclidean distance is also referred to as the L2 norm between two vectors. We normalise each vector to have an L2-norm of 1 before computing the L2 norm between two vectors for convenience in visualising this distance since the magnitude would influence how the matrix of Euclidean distances is created. The L2 Norm is defined as below:

$$\text{L2 Norm} = ||A - B||_2^2$$

Using these two formulas, we obtain Figure A.39, which measures the similarity between different training instances on the scoring layer, Figure A.40, which measures the same but for the classifying layer and Figure A.41 which measures the similarity between the same training instances' scoring and classifying layer. Note that we have deliberately created a NaN in the diagonal for better visualisation.

Suppose our hypothesis, as mentioned earlier, that SCEPTR assigns cancer-targeting TCRs to one close region, is true. In that case, we should expect the Euclidean distances of different training instances' classifying layers and scoring layers to be close to 0 and cosine similarity to be close to 1. Yet, the similarity between different training instances' classifying and scoring layers is low in Figure A.39 and A.40. In the same training instance, the similarity between the scoring and classifying layer is also low. This suggests that SCEPTR does not assign spatial locality to TCRs by its specificity in its embedding hyperspace.

5.3 Limitations

Although we have seen success in demonstrating subsymbolic encodings' superiority in expressivity as opposed to symbolic encodings, there are several key points to address where we believe further research is needed to further establish our observation.

5.3.1 Data Limitations

Although data from 190 patients is not considered a small number in computational biology as clinical data is complex to collect, it is insufficient to say that the model has generalised the relationship between TCR repertoires and cancer. It has been seen that individuals with the same ethnicity will have a similar immune system as opposed to individuals from different ethnic backgrounds [20, 40, 86, 87]. Our data could be biased in the sense that our data is collected from patients with the same ethnicity, as data collection has been performed in the UK only. Thus, most patients in this study are expected to have similar ethnic backgrounds.

Furthermore, we have seen that TCR-BERT has overfitted as there are too many parameters to train given the amount of labelled data. We propose two methods which could help overcome this: imposing a larger L2 penalty, where the best L2 penalty can be chosen from a K-Fold Cross Validation with Grid Search. The second method introduces more data, which can also help the downstream model for SCEPTR to understand the relationship between TCRs and cancer better.

5.3.2 Quality of Embedding Space

As seen in Section 5.2.2, different downstream classifier instances that use SCEPTR's encodings give significantly different models. For example, some of the models' weights are almost orthogonal to each other, as seen in Figure A.39 and A.40. This demonstrates that SCEPTR's embedding space does not fully capture disease specificity.

Although this is an expected phenomenon as the dataset used to train SCEPTR has not labelled TCR specificity [69], it would be more desirable for a TCR embedding space to assign spatial locality based on specificity. This applies to identifying whether a patient is diseased and other TCR problems, such as pairing alpha and beta chains and pairing TCRs with antigens.

Yet, this limitation does not reduce the performance of SCEPTR's downstream model, as we have seen impressive results in classifying whether a patient has cancer.

5.3.3 Computational Resources

Since this is a student's project, we ran our code under a computer cluster shared amongst all students. This means there are certain restrictions on how extensive our experiments can be. Therefore, we are unable to complete a grid search on the best set of hyperparameters, as we are unable to occupy multiple computers with the exact specification as mentioned previously in section 4.3.1 to run the training for six different encoding types, especially considering TCR-BERT takes a large amount of GPU memory and time to run.

It would be more desirable if each encoding method had its own set of hyperparameters so that the choice of hyperparameters does not influence the model's performance, which relates to the encoding space's expressivity.

We believe that this problem is particularly predominant within TCR-BERT. Its downstream model requires regularisation through the `weight_decay` parameter. However, the best value for this hyperparameter has not been cross-validated, and the l2-penalty used, 0.25, is only a rough guess. It is possible that TCR-BERT could perform better than SCEPTR as the encoding model; however, this will be verified in future works. We will propose methods to compare TCR-BERT's and SCEPTR's expressivity in section 5.4.4.

5.4 Future Works

We conclude our findings with a proposal for future works that could be done to further this research. We focus on three aspects: increasing the robustness of our methodology to compare symbolic and subsymbolic encodings, improving the classification scores and the interpretability of the models. We will also provide ideas on how to tackle TCR-BERT overfitting.

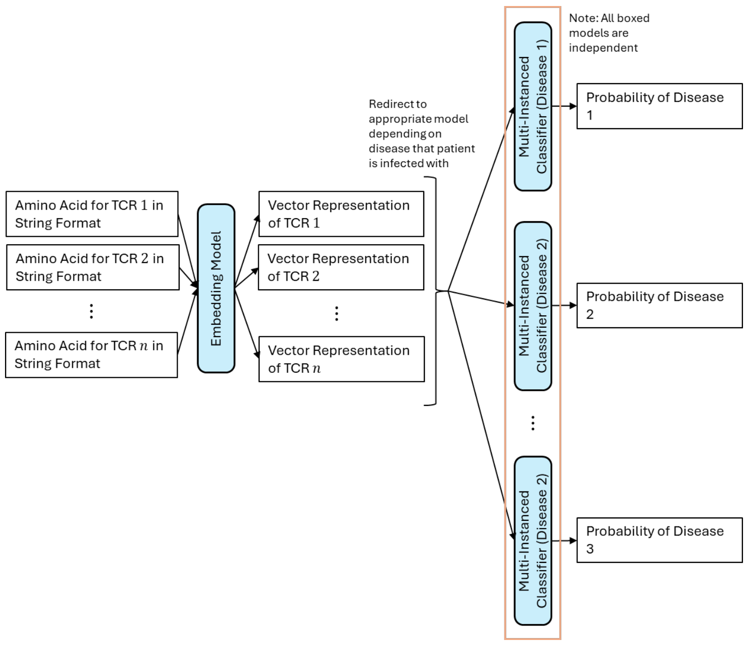

Amino Acid for TCR 1 in String Format
Amino Acid for TCR 2 in String Format
⋮
Amino Acid for TCR n in String Format
Embedding Model
Vector Representation of TCR 1
Vector Representation of TCR 2
⋮
Vector Representation of TCR n
Redirect to appropriate model depending on disease that patient is infected with
Multi-Instanced Classifier (Disease 1)
Multi-Instanced Classifier (Disease 2)
⋮
Multi-Instanced Classifier (Disease 2)
Note: All boxed models are independent
Probability of Disease 1
Probability of Disease 2
Probability of Disease 3

Figure 5.2: Fine Tuning Regime for Encoding Models

5.4.1 Fine Tuning Encoding Models

We have not fine-tuned TCR-BERT or SCEPTR during our experiments to create a more desirable output embedding space. This is due to demonstrating robustness with encoding models and GPU memory limitations. These embedding models are a generalised method of representing TCRs numerically whilst respecting neighbouring TCRs, which is not something that symbolic encodings can do. They are not specially designed for disease identification. We believe that fine-tuning these models towards disease identification can mitigate the problems with embedding space quality as discussed in section 5.3.2.

During our experiments, we attempted to fine-tune models naively by making all parameters in the encoding trainable, and we observed an explosion in GPU memory consumed. This memory cannot be released before one backpropagation step, and this problem is particularly prevalent with TCR-BERT as it has many parameters. This causes a large amount of gradients to be accumulated during the training step.

This problem has also been observed with conventional LLM fine-tuning, such as the 65 billion parameter LLM Llama requires over 780GB memory to fine-tune [55]. We believe that using fine-tuning tricks such as Low-Rank Adaptation (LoRA) [88] or Quantised LoRA [89] can significantly reduce the memory footprint required. Yet, the amount of TCR instances within one repertoire can be large, meaning that the memory footprint cannot be estimated unless we know the number of instances within the largest repertoire.

Whilst we conjecture that being able to fine-tune the model can improve the accuracy obtained, we also argue that the amount of data required to fine-tune the model exceeds what we have for our dataset. TCR-BERT is a 57 million-parameter model, and SCEPTR has 153 thousand parameters. This means that if we have only 190 patients' repertoire data, it will most probably cause the model to overfit quickly, as we have seen that TCR-BERT's 1538 parameter downstream model overfitted quickly.

To mitigate overfitting due to a lack of data, we propose fine-tuning one LLM by incorporating more TCRs from patients infected with diseases other than cancer. Using these data, we can train a series of downstream classifiers, each with the same architecture as ours.

Since we know what disease the patient is infected with, we can train each model so each model identifies one disease in a One-vs-All manner. During each update step, we update the classifying and embedding models. We must update each classifier sequentially rather than completing the training for one classifying model and going on to the next. This prevents the encoding model from forgetting knowledge from the first processed disease. We believe this training regime would work because it leverages data from multiple diseases and encourages the model to rearrange its hyperspace such that the TCRs are placed in the hyperspace according to its disease specificity.

A diagram of this training paradigm is attached as Figure 5.2. However, this approach does not mitigate the problem with GPU memory consumption.

5.4.2 Patient Data

It has been observed in [31] that when the patient's background, such as ethnicity, age, gender and other factors, are included in the consideration of classifying whether the patient is infected with COVID-19, Lupus or HIV, this will increase the AUC of the classification rather than simply analysing TCRs and B cell Receptors (BCRs).

Given this promise, we believe that allowing the model to know more about the patient enables the model to make well-informed decisions regarding whether the patient has cancer. This also follows what is known in biology, since the likelihood of having cancer is positively correlated with age [90], and is dependent on family background, such as a strong family history of cancer increases the risk of cancer [91].

To incorporate this information into our model, we could concatenate the vector representing the patient's background onto the patient's bag-representing vector. This assumes that the patient background vector is scaled within an appropriate range.

Our data also comes from patients exhibiting lung cancer symptoms. Whilst it is aspired that we can use our model to find an asymptomatic cancer patient, classifying between a healthy patient and an asymptomatic cancer patient's TCR repertoire is a more difficult task than what we attempted.

5.4.3 Model Verification

We propose two tests of varying difficulty to verify whether the model has learnt a generalisable relationship, whereby the positively predictive TCRs are assigned a non-zero weighting. Both tests rely on having acquired TCRs known to be cancer-targeting, which we will denote as 'the cancer TCR'. This data should not be difficult to obtain, where we found 182 TCRs that are known, with high confidence, to be cancer-targeting in VDJdb [59].

We name the more straightforward test the classifying test, demonstrated graphically in Figure 5.3. We perform this test by artificially creating a patient's TCR repertoire by gathering control TCRs and grouping them into one file. We assert that this set of control TCRs should not have been involved in the model's training. We will then pass this augmented repertoire into the model,

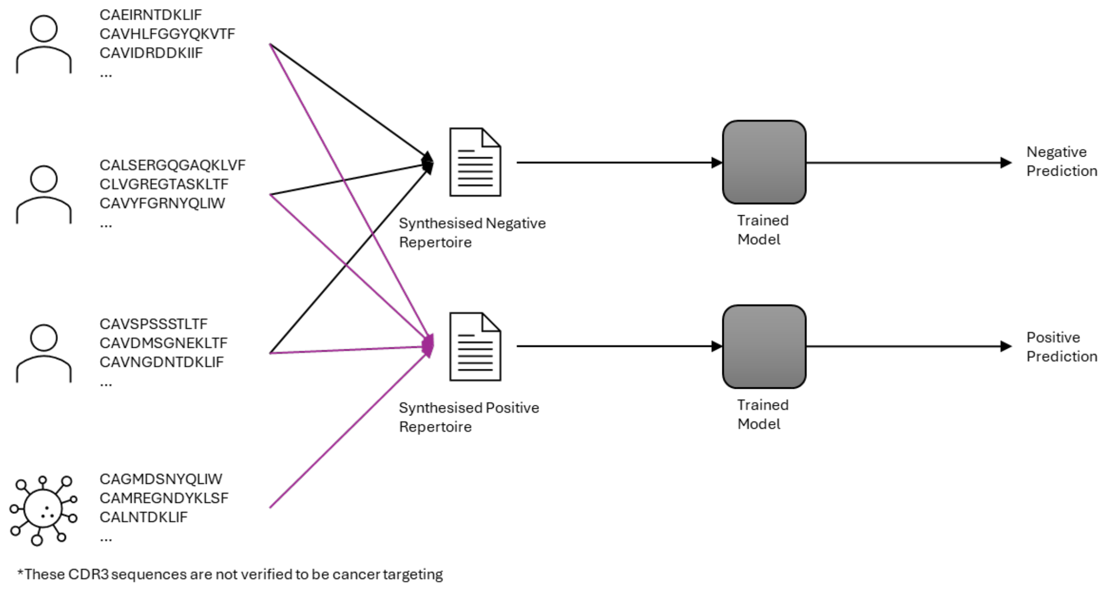


Figure 5.3: Classifying Test

where we should expect the output of the model, p, i.e. the probability of having cancer, to be lesser than 0.5, corresponding to a negative prediction.

Subsequently, we add the cancer TCRs into the augmented repertoire, simulating a cancer patient's blood draw. We expect the model to output a probability higher than p and higher than 0.5, corresponding to a positive prediction. This is because we know cancer-targeting TCRs exist within the repertoire, which can only occur when the patient has cancer.

The second test is the score test, which is more complicated than the classifying test. For a model to pass this test, we expect its scoring layer to output a high score for the cancer TCR. We can objectively review whether this score is high by comparing it with the highest score the model outputted when given a TCR repertoire. This comparison can be a percentage difference between this highest score and the score for the cancer TCRs.

The intuition for this test is that if the model outputs a similarly high score for the cancer-targeting TCR, then this demonstrates evidence that the model has captured the relationship between TCRs and cancer. We argue that this is more difficult than the classifying test as this is also a test for the encoding model. If an encoding model does not know TCR specificity, it will probably fail the score test as it does not assign spatial locality by TCR specificity.

Provided that the model has passed these two tests, it poses a strong argument that the model has acquired a generalisable relationship between TCRs and cancer. This is because different cancer types and patients lead to different TCRs being proliferated, so passing these two tests is not simple.

5.4.4 Usage of TCR-BERT

As we cannot create a convincingly generalisable performance when we use TCR-BERT as the upstream encoding model, we propose using two algorithms for reduction on TCR-BERT's feature

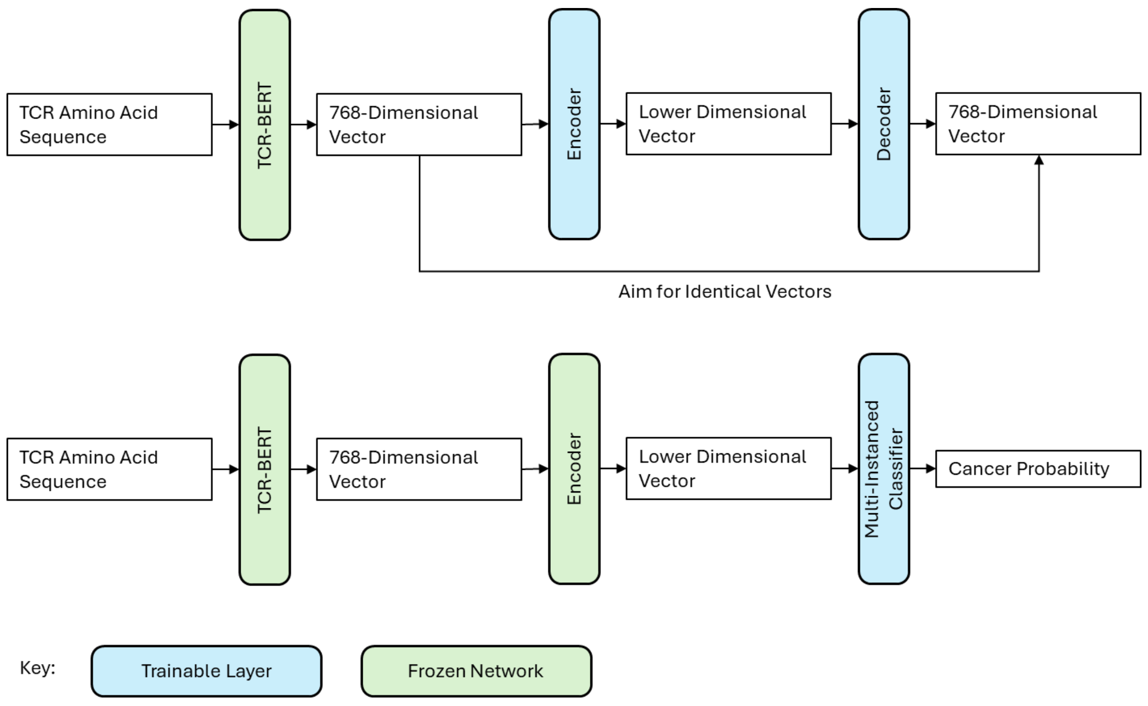


Figure 5.4: TCR-BERT's Encoder Training Paradigm

space to reduce the trainable parameters in the downstream classifier.

AutoEncoders

Autoencoders are a pair of neural networks that are trained simultaneously. The encoder takes in the raw embedding of 768 dimensions and downscales it to a lower dimension. In contrast, the decoder takes this lower-dimension vector and aims to restore it to its original input.

After training, we will obtain a lower-dimension vector from the encoder. By design, this vector will represent the TCR similarly to the original output from TCR-BERT. Note that this vector will not be as expressive as the output from TCR-BERT as there are information lost.

After training the encoder, we freeze its parameters and use the encoded vector to train the downstream classifier. This can significantly reduce the amount of parameters in the classifier. A diagram of the training paradigm for the autoencoder and the downstream classifier is as Figure 5.4.

To train the encoder, we argue that using the same TCRs we will use for training the downstream classifier will not cause data leakage as long as we do not include the test set in training the autoencoder. This is because the encoder does not know the specificity of the TCR, as its end goal is to encode TCRs onto a lower dimension. Hence, the encoder never knows the specificity of the TCR and thus cannot assign spatial locality for TCRs with the same specificity.

Johnson-Lindenstrauss Lemma

The Johnson-Lindenstrauss Lemma provides a simplified approach as the AutoEncoder. The Lemma is defined as below [92, 93]:

Lemma 3 (Johnson-Lindenstrauss) *Let $P \in \mathbb{R}^d$ be a set of n points, $\epsilon > 0$ be a parameter and $k = (1/\epsilon^2)\log n$. If Q, a $k < n$ dimensional space, is a subspace of P obtained from a linear projection, then all pairwise distanced in P are within $1-\epsilon$ close to each other in Q with probability at least 0.5.*

Whilst we do not provide the proof of the Johnson-Lindenstrauss lemma here, both the classical proof [92] and the simplified proof [93] used a random projection. Given this, we can construct a linear map which contains randomly initialised values taken from a standard uniform distribution and map each output embedding generated by TCR-BERT with this linear map so the output embedding is now in a lower dimension Q.

Using the lemma, we are guaranteed that the pairwise distance of datapoints in this lower-dimensional subspace is somewhat close to the pairwise distance of the same datapoints in the high-dimension subspace, even with a random projection. This means the geometric properties of how TCR-BERT represents TCRs are ϵ preserved in the lower dimension subspace. Hence, a linear map can reduce the minimum number of parameters needed to model TCR-BERT's hyperspace.

5.4.5 Autoencoders for symbolic encodings

Although we showed that symbolic encodings are not as expressive as subsymbolic encodings in this study, taking a feature-wise average to obtain a numerical representation of the TCR CDR3 sequence may not be the most effective way of representing it in a vector. We have seen in MINN-SA [23] that they used TESSA [44] to encode the Atchley matrices onto a 30-dimensional vector, which could be a better way to represent the TCRs numerically.

However, we observed that TESSA is poor in encoding TCRs sampled from a tumour environment, and TESSA oversees the language-like property of TCRs, which is why we did not use TESSA in our study. We believe that if there is another autoencoder for each of the symbolic encodings we used, utilising them to encode the matrix representing the CDR3 sequences into a vector could be better than simply taking an average.

That said, it could still be a worthy study to use TESSA on this problem should time allow.

Chapter 6

Conclusions

We conclude this report by objectively reviewing how much we met the goals set in our Research Aims in Section 2.3. We will also summarise our achievements in this research project and the potential future developments that could be done to this project.

6.1 Summary of Achievements

In this report, we have demonstrated with novelty that representing TCRs numerically using large language models' embedding hyperspace provides more robust and better performances than physico-chemical properties.

We showed this through a simple 2-layered neural network, where we spotted three significant differences between the performance of downstream classifiers that use subsymbolic and symbolic encoding spaces.

1. The classifier which uses TCRs encoded from physico-chemical properties cannot descend in training loss as effectively as the model which takes in large language model embeddings. This demonstrates that there are no minima to reach in this optimisation problem, or the minima are hard to reach when we use physico-chemical properties to encode TCRs. This observation is not made when we use large language model embeddings to encode TCRs.
2. Ignoring TCR-BERT's downstream model's performance due to overfitting caused by over-determination, most symbolic encodings' downstream models have a significantly lower test set AUC than SCEPTR's downstream model's test set AUC. This demonstrates that SCEPTR's downstream model is more powerful than symbolic encodings. Although Kidera Factor's downstream model's mean test set AUC was similar to SCEPTR's, it cannot match the maximum AUC that SCEPTR's downstream model reached.
3. We used a random encoding generated from a random uniform distribution from 0 and 1 to form a 5-dimensional vector as a 'control' symbolic encoding. Since random encoding is not designed to provide meaning for each amino acid as opposed to physico-chemical properties, which provide high-level meanings for each amino acid, we expect random encoding to perform worst among all symbolic encodings.

However, this has not been observed, whereby the random encoding occasionally performs better than physico-chemical encodings. This suggests evidence that physico-chemical encodings are as expressive as random encodings.

We showed robustness in the subsymbolic encoding methods through their downstream model's performance, such as high accuracies and AUC. The two subsymbolic encoding methods are robust since the encoding models were never trained with disease specificity. Yet, they showed promising results with high AUC and accuracy in the downstream model.

However, SCEPTR does not assign spatial locality based on TCR specificity. This can be concluded as different training instances' best-performing checkpoints have significantly different model parameters, which the cosine similarity and Euclidean distance can verify. We showed that on certain occasions, the scoring layer's weights are orthogonal to the classifying layer's weights.

We used the Area Under the Receiver Operating Characteristic Curve (AUC) to demonstrate that the subsymbolic encodings perform better than the symbolic encodings if we keep the model architecture as an invariant factor within this comparison.

TCR-BERT's downstream classifier did not have a better mean AUC than the symbolic encodings. It had too many trainable parameters compared to the training data. On the other hand, SCEPTR, a model that casts TCRs onto a lower dimension space, performed extraordinarily. One of its downstream models had a 100% AUC in the test and evaluation sets, demonstrating its robustness.

We also interpreted the signals that this SCEPTR's downstream model picked up. We noticed that the downstream model preferred beta chains over alpha chains in finding whether a patient has cancer, which aligns with what we know in biology, as beta chains are thought to contain more information about TCR specificity than alpha chains. The model also picked up chains that are expanded within multiple patients, even considering that the model never knew how expanded a TCR is within a repertoire.

In particular, this simple 2-layer neural network model with only 130 trainable parameters trained on SCEPTR's encodings has achieved state-of-the-art AUC when we average its AUC across the ten training instances. The testing AUC and evaluation set AUC are 95% and 98%, respectively, in the early stages of non-small cell lung cancer.

Although, arguably, we did not use the same dataset as previous works, the simplicity of our model and extreme results, such as a 100% AUC on both the test set and evaluation set, have shown robustness in our work.

In conclusion, our research has shed light on using subsymbolic encodings in cancer prediction. We have shown that it is more robust and expressive than symbolic encodings with novelty. Whilst we do not know how the embedding models represent TCRs numerically, we have observed that the downstream classifier of one optimally fitted instance follows what is known to be biologically accurate.

6.2 Summary of Future Works

To further our research in this paper, we have proposed five different directions as the future works. These five directions are in 3 directions: improving the accuracy and predictability of our algorithm in TCR cancer prediction (Point 1), evaluating whether subsymbolic encodings are more expressive than symbolic encodings (Points 4, 5) and increasing the robustness of our study (Points 2, 3).

1. The encoding models are not informed about disease specificity. Therefore, we suggested that fine-tuning the encoding models on disease specificity could help the model understand how to encode TCRs better. This could help the encoding model create better embeddings to solve other problems, such as alpha and beta chain pairing.

2. Since our data is collected from 6 hospitals in the United Kingdom. Therefore, the data might potentially be biased. This is because it is expected that the patients involved in this study will share a similar genetic background. This leads to a weakness in our model, where our model might not perform equally well on patients with a significantly different genetic background. We propose that collecting more data from other patients with a broader background could improve the robustness of the classifier.

   Our data also comes from lung cancer patients who exhibited symptoms. A future extension of our work could be using asymptomatic patients to train our model.

3. We propose that we verify our models with the use of TCRs that are known to be cancer-targeting. We proposed two tests of varying difficulty to verify the model's knowledge. One focuses on making sure the model picks up on the TCR known to be cancer-targeting, and another test focuses on making sure the score for the cancer-targeting TCR is high.

4. We have seen that TCR-BERT's downstream model overfitted quickly. This is because TCR-BERT has a high dimensional output, which causes the downstream classifier to have many trainable parameters compared to the labels we have. We propose using two reduction downstream algorithms: the Johnson-Lindenstrauss Lemma, which involves a random linear map, and an autoencoder. We believe that we can use the training data for the encoder to be the same as the data that trains the classifier since there is no data leakage of disease specificity to the encoder.

5. We created vector representations of CDR3 sequences from TCR-BERT's embedding and the symbolic encodings by taking a feature-wise average. This may not be the best way to obtain a vector representation of a CDR3 sequence. Therefore, we believe we could use an autoencoder explicitly trained to transform the matrix representing the CDR3 sequence into a vector. However, this autoencoder must capture amino acids' language-like structure.

We have proposed methods to achieve these future works in Section 5.4.

Appendix A

Figures & Tables

A.1 Train-Test Split

We enclose the statistics obtained from each model's training and testing phase. Table A.1 illustrates the amount of repeats we took for each encoding method.

Repeats	Encoding Method
5	TCR-BERT, Physico-Chemical Encodings, Random Encoding
10	SCEPTR

Table A.1: Repeats taken for each encoding method

Note that vertical lines on the graphs denote the best checkpoint algorithm that has been selected. This checkpoint is selected based on the best test set AUC. Each entry in the confusion matrix is in the form of $\mu \pm \sigma$, where μ is the mean and σ is the standard deviation of that entry from each repeat's best checkpoints' performance.

A.1.1 TCR-BERT

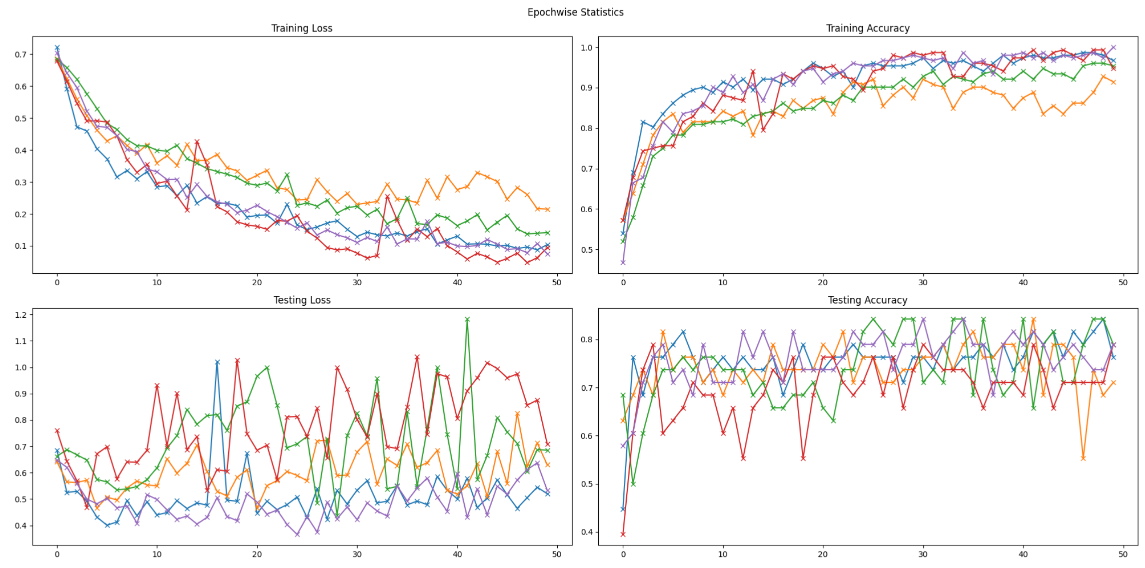


Figure A.1: Loss & Accuracy on Training and Test Sets for each Epoch

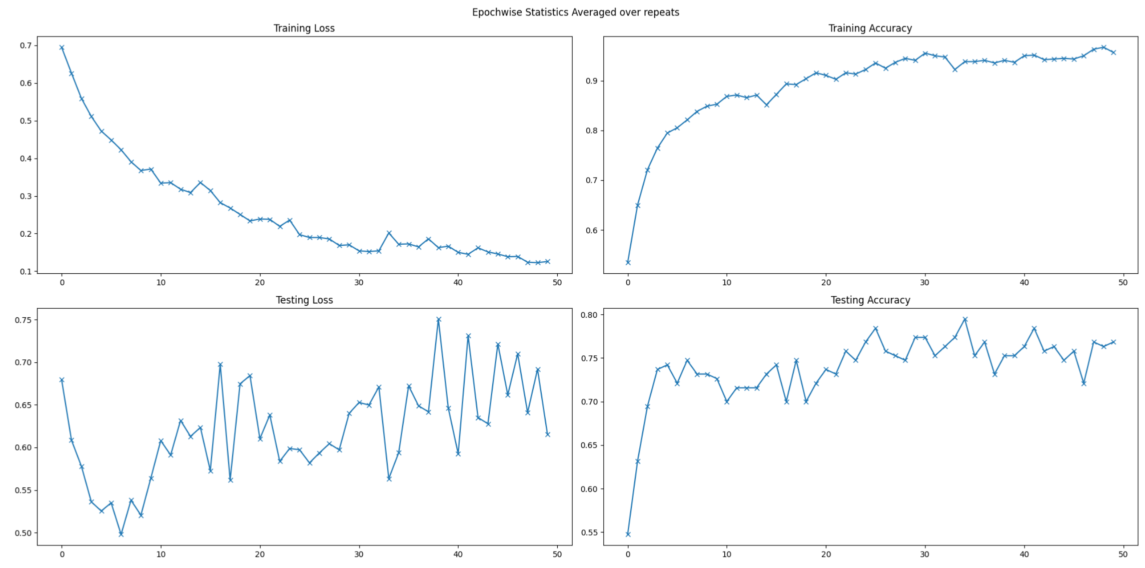


Figure A.2: Loss & Accuracy on Training and Test Sets Averaged on Repeats

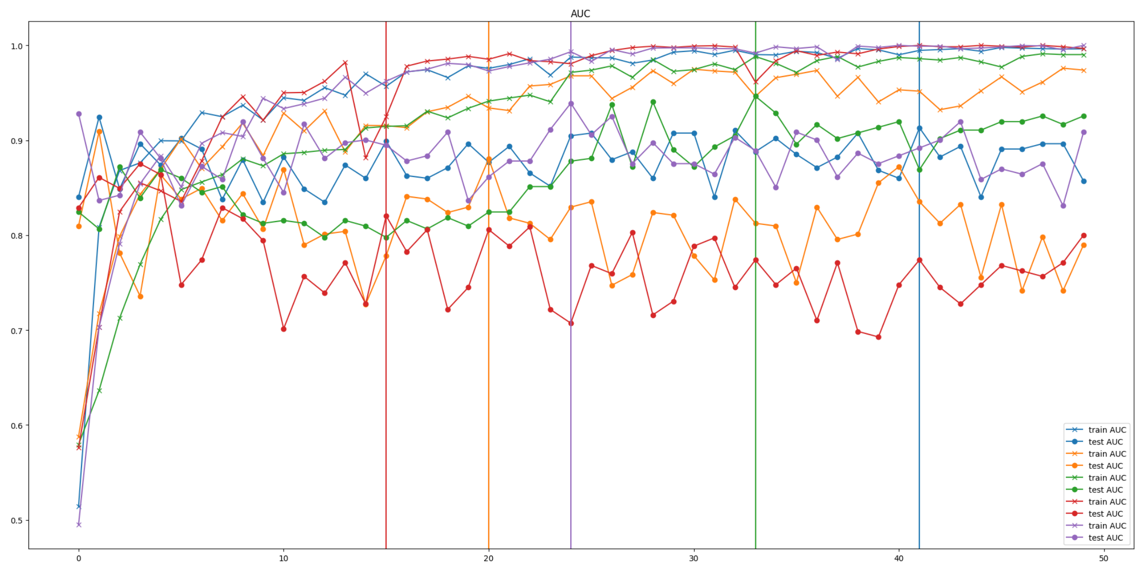


Figure A.3: AUC on Training and Test Sets

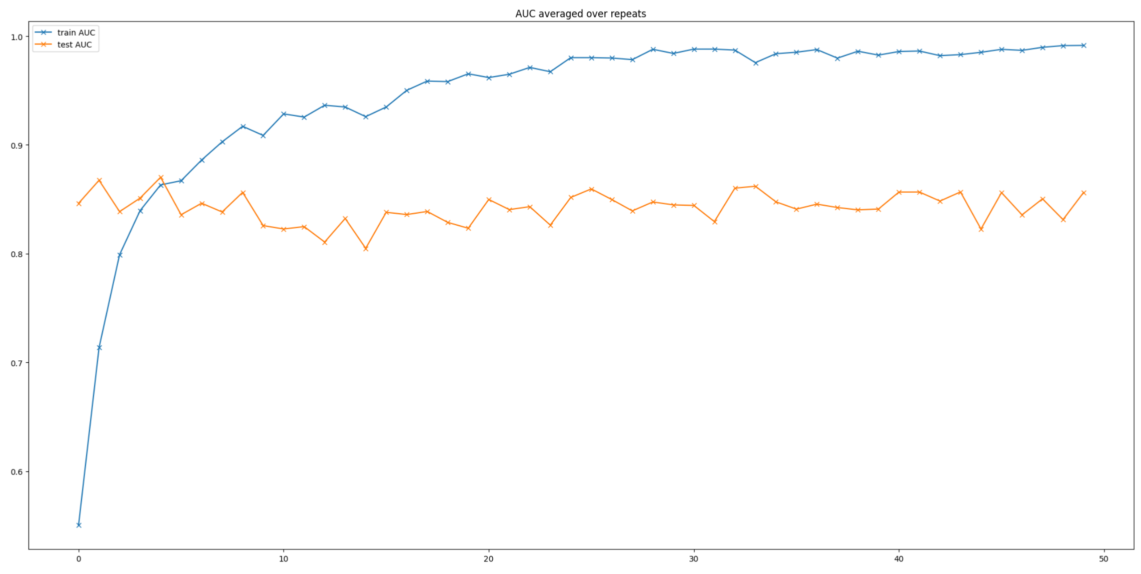


Figure A.4: AUC Averaged on Training and Test Sets

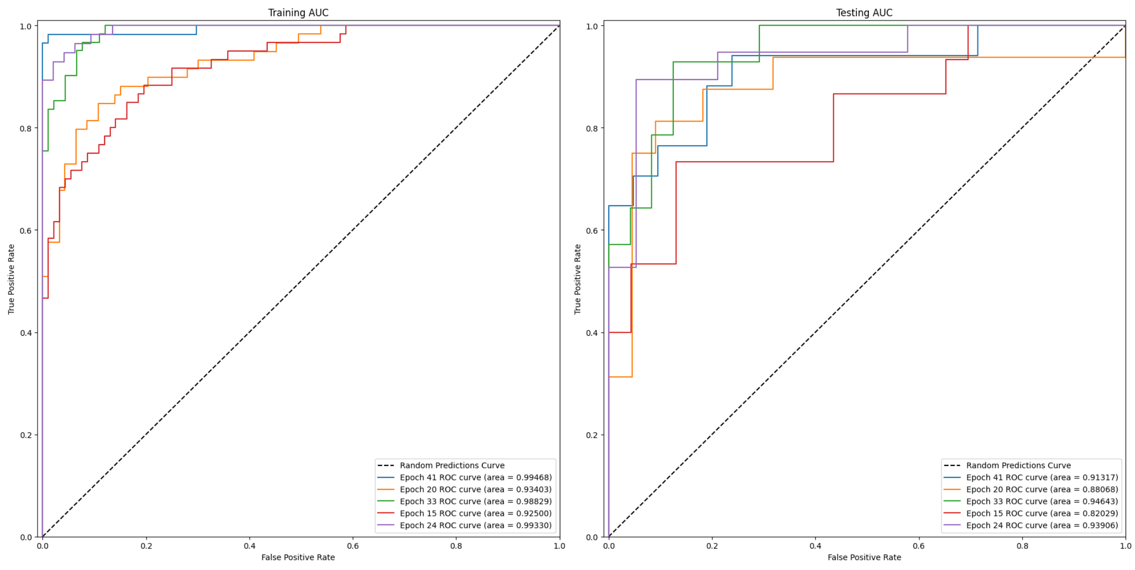


Figure A.5: AUC Curve for each repeated run

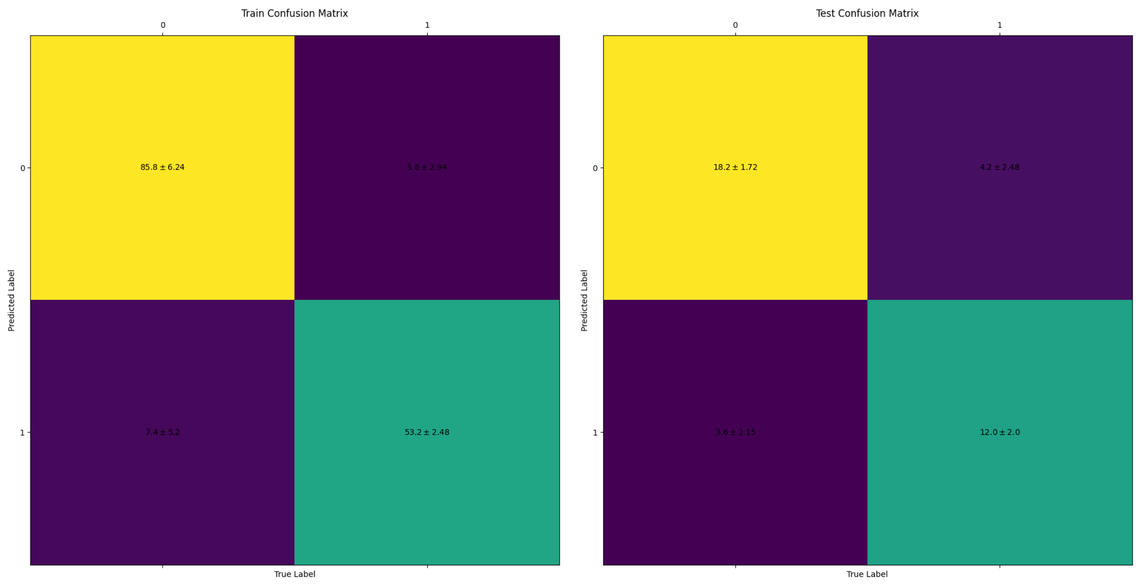


Figure A.6: Confusion Matrix with Mean and Standard Deviation Across Repeats

A.1.2 SCEPTR

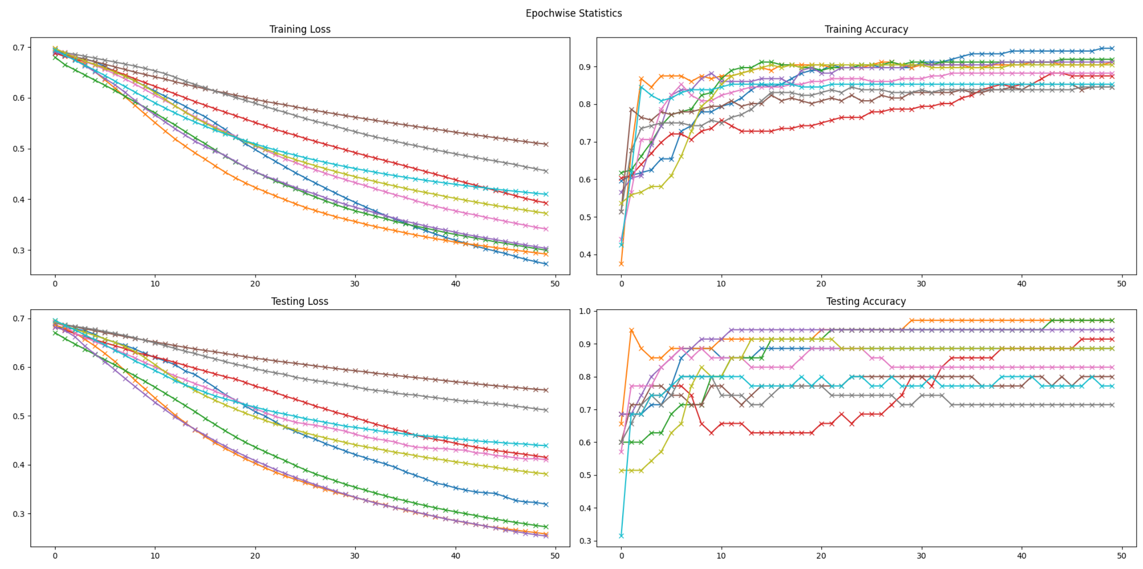


Figure A.7: Loss & Accuracy on Training and Test Sets for each Epoch

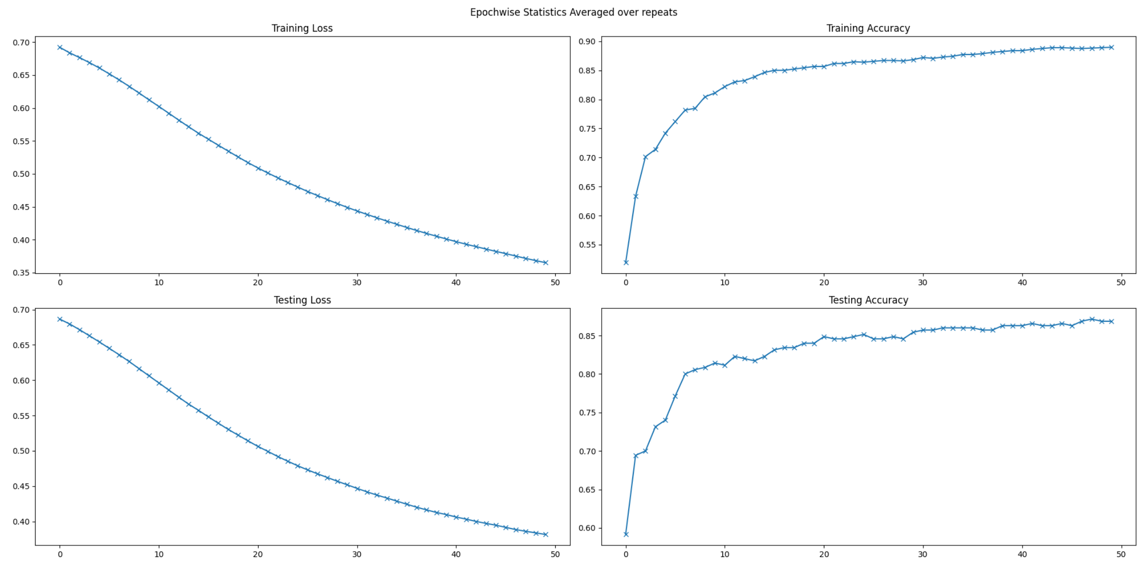


Figure A.8: Loss & Accuracy on Training and Test Sets Averaged on Repeats

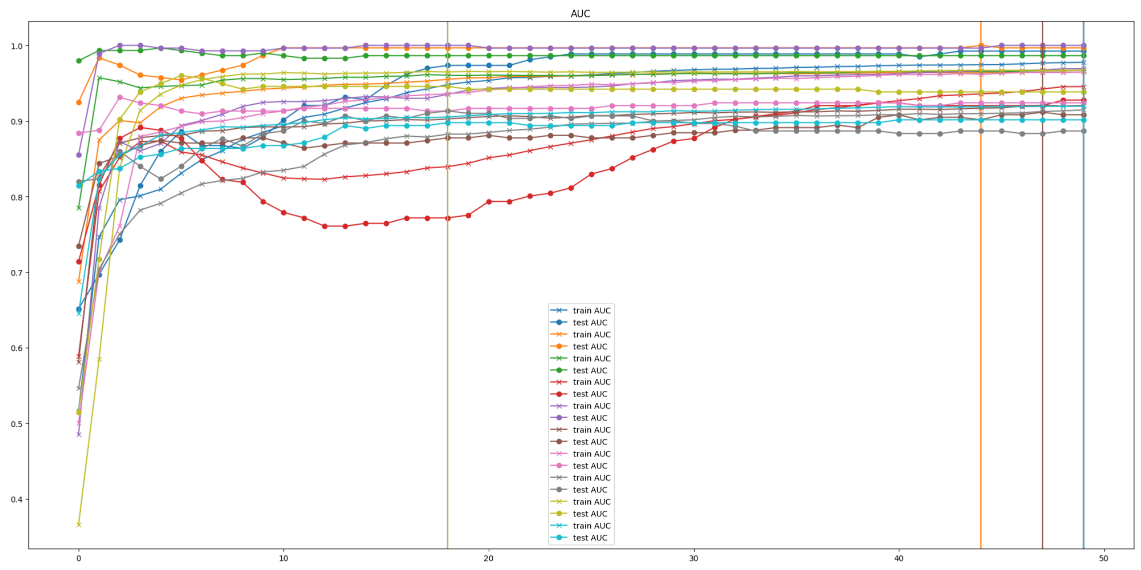


Figure A.9: AUC on Training and Test Sets

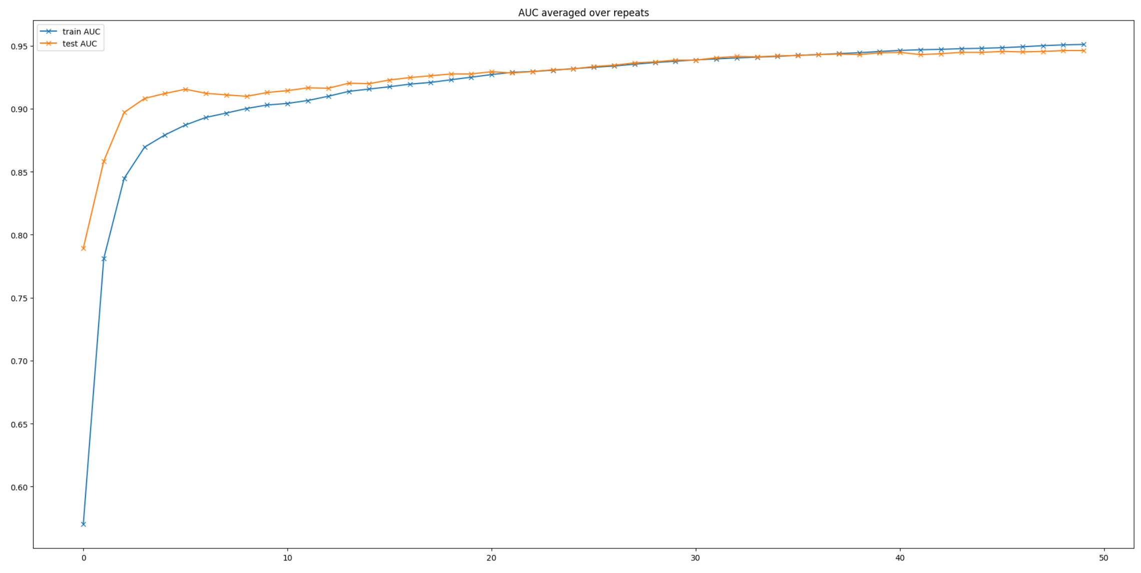


Figure A.10: AUC Averaged on Training and Test Sets

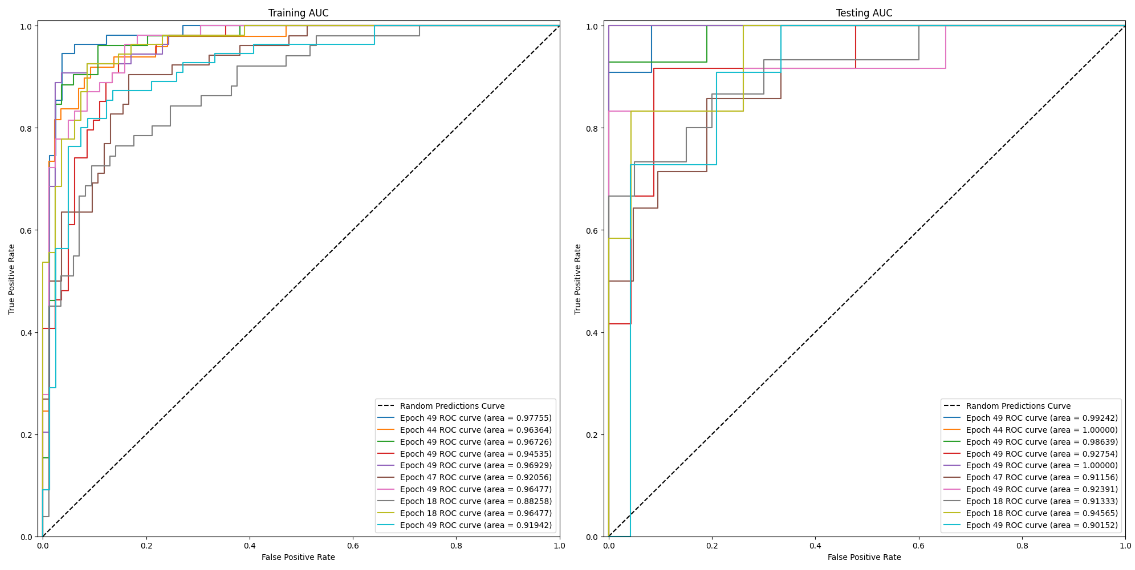


Figure A.11: AUC Curve for each repeated run

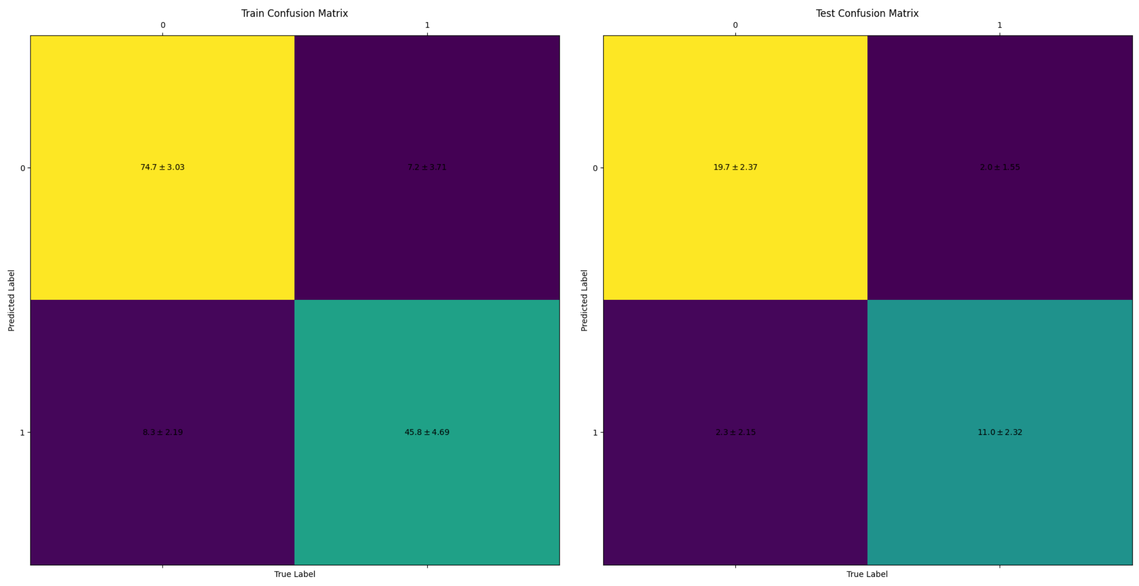


Figure A.12: Confusion Matrix with Mean and Standard Deviation Across Repeats

A.1.3 Atchley Factors

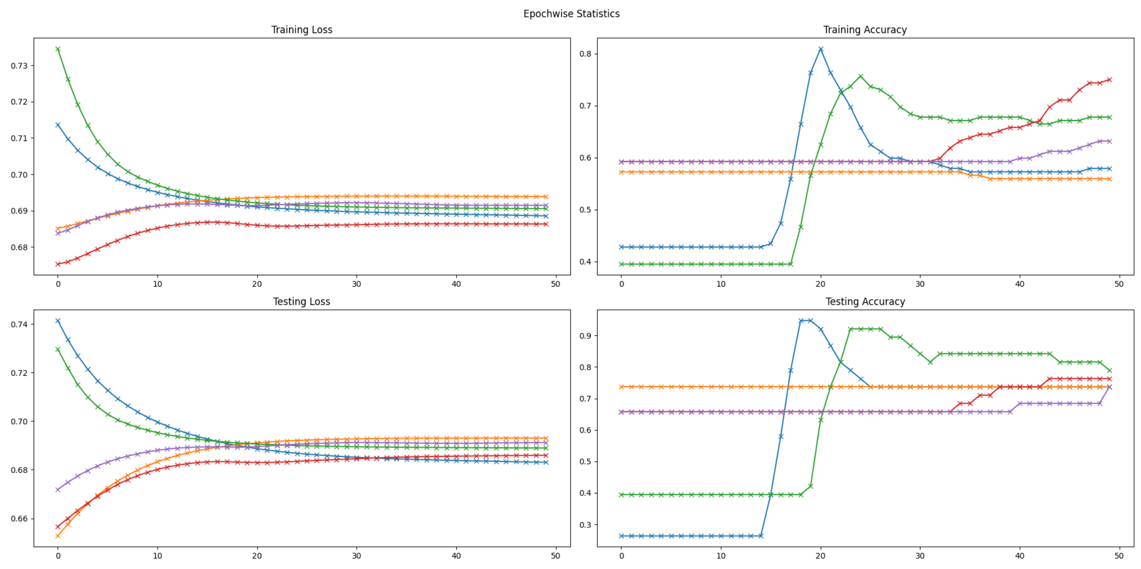


Figure A.13: Loss & Accuracy on Training and Test Sets for each Epoch

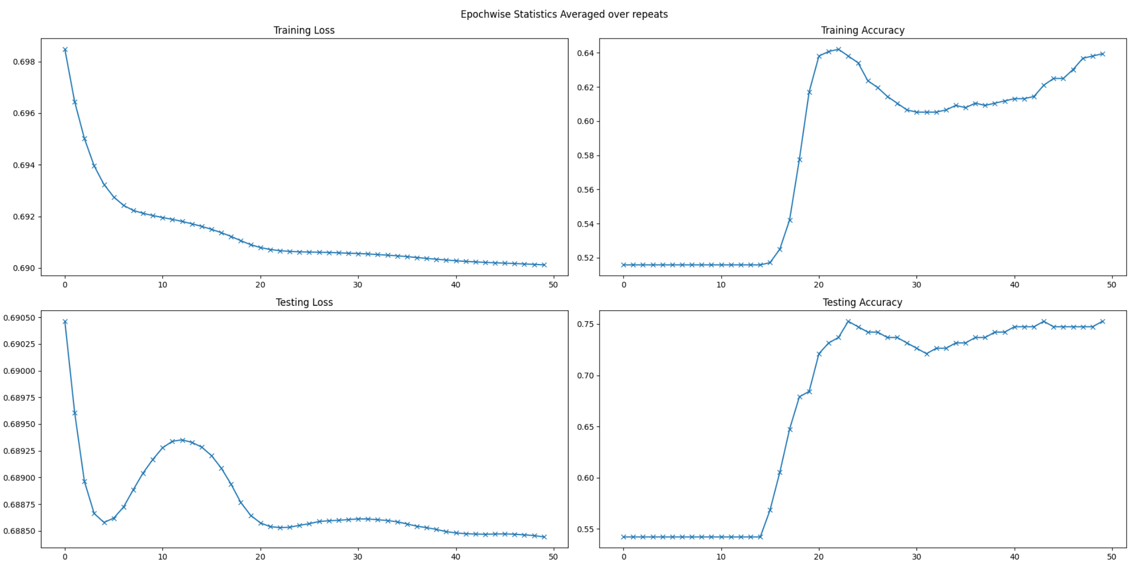


Figure A.14: Loss & Accuracy on Training and Test Sets Averaged on Repeats

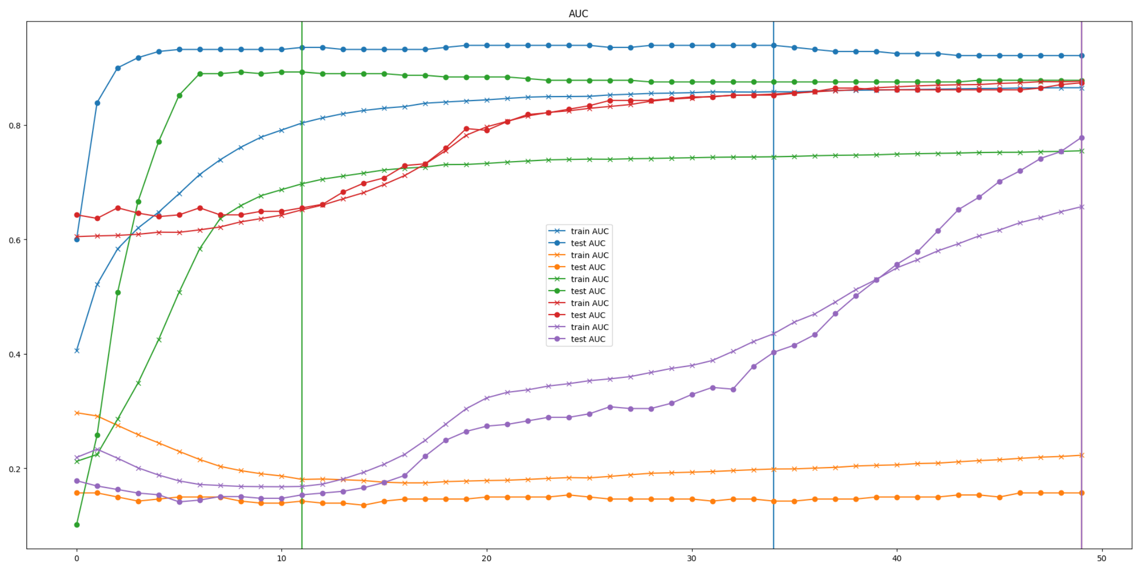


Figure A.15: AUC on Training and Testing Sets

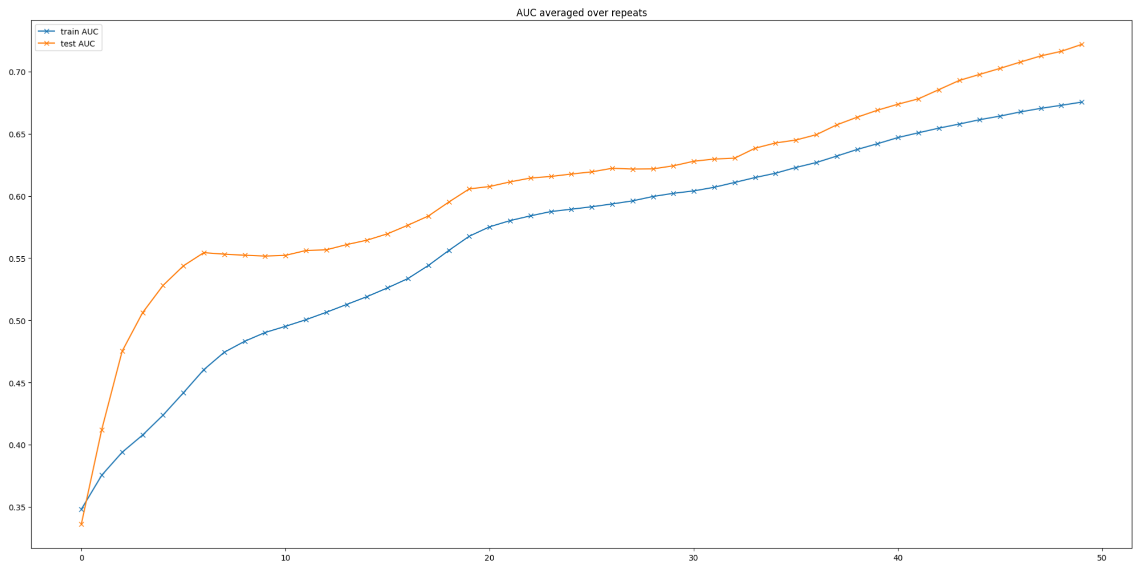


Figure A.16: AUC Averaged on Training and Test Sets

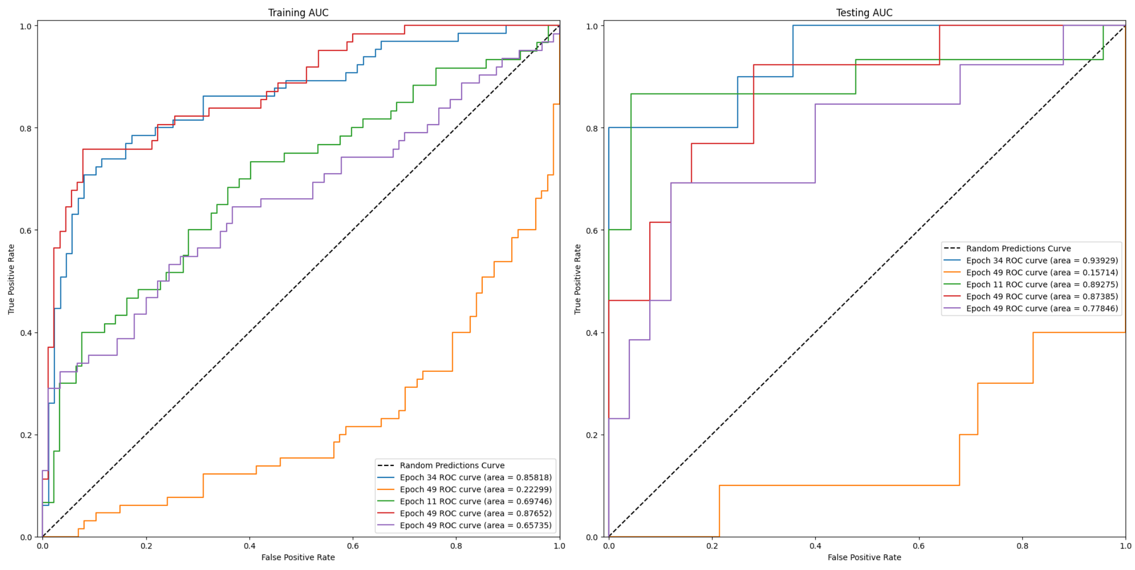


Figure A.17: AUC Curve for each repeated run

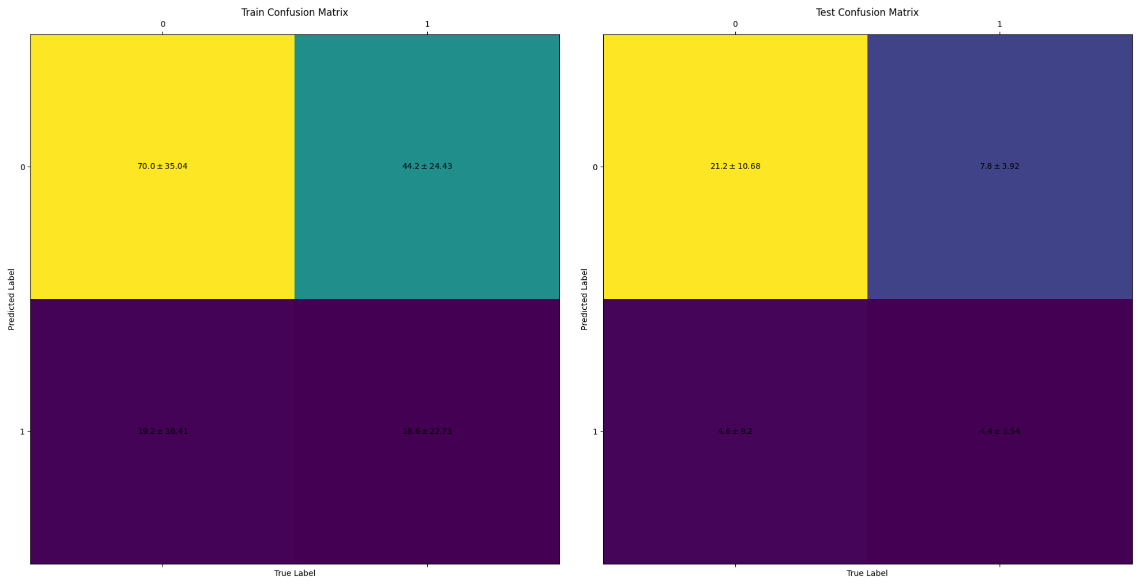


Figure A.18: Confusion Matrix with Mean and Standard Deviation Across Repeats

A.1.4 Kidera Factors

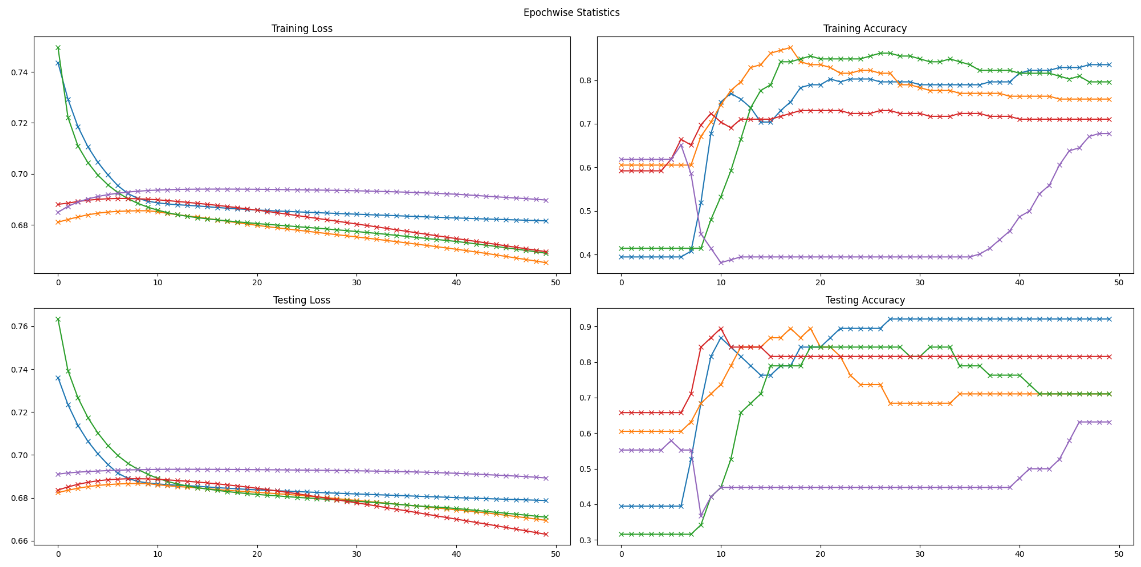


Figure A.19: Loss & Accuracy on Training and Test Sets for each Epoch

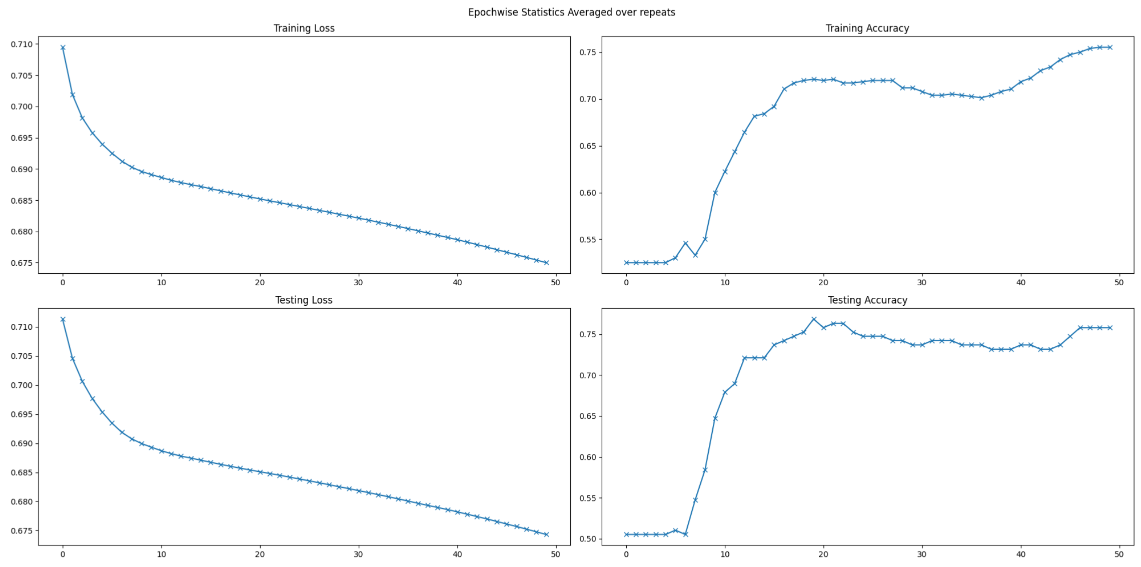


Figure A.20: Loss & Accuracy on Training and Test Sets Averaged on Repeats

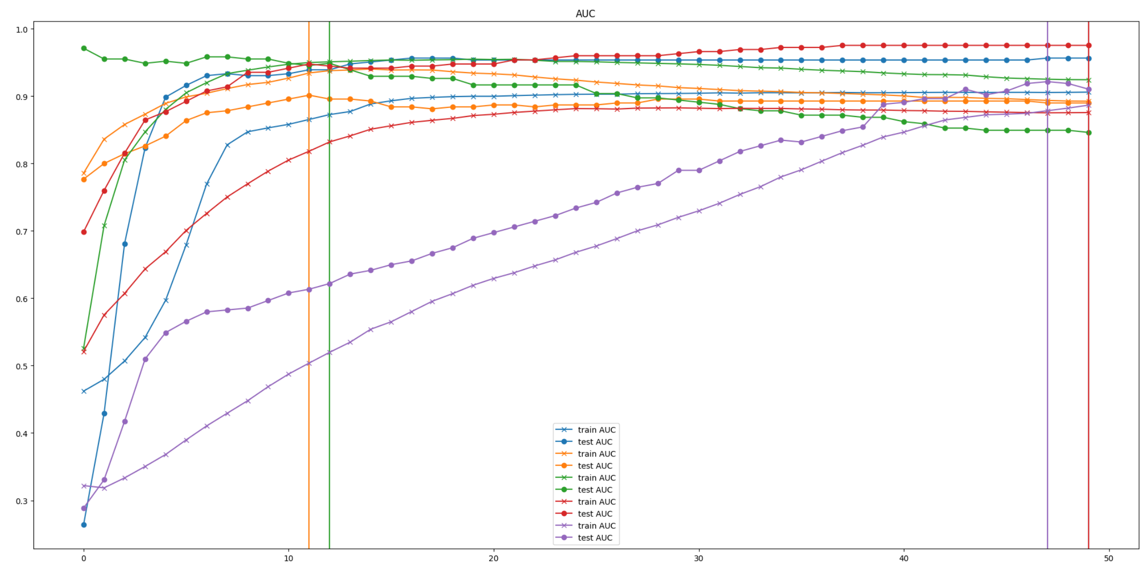


Figure A.21: AUC on Training and Test Sets

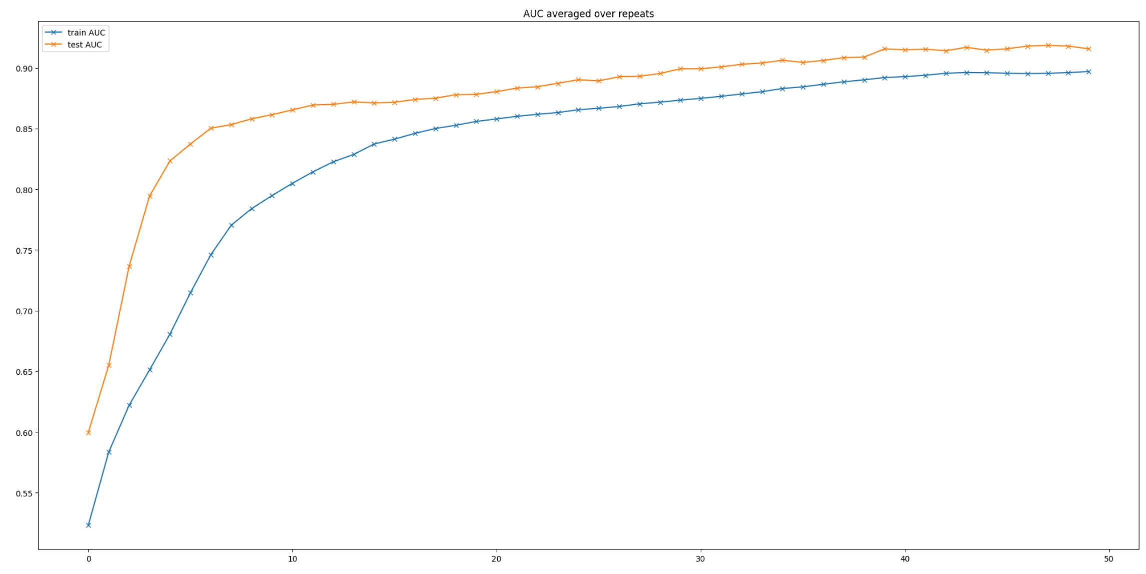


Figure A.22: AUC Averaged on Training and Test Sets

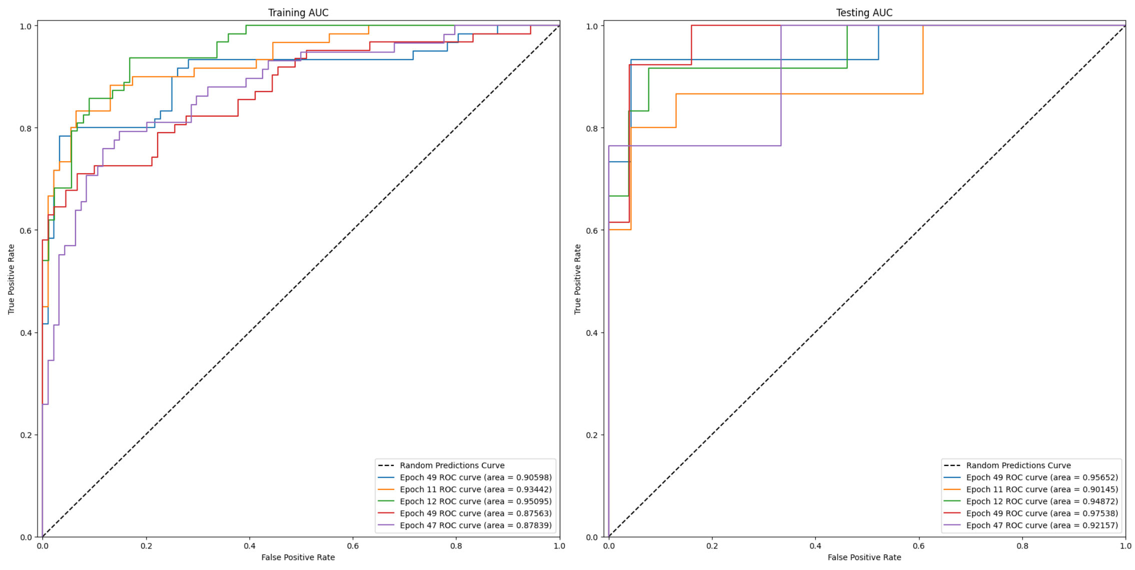


Figure A.23: AUC Curve for each repeated run

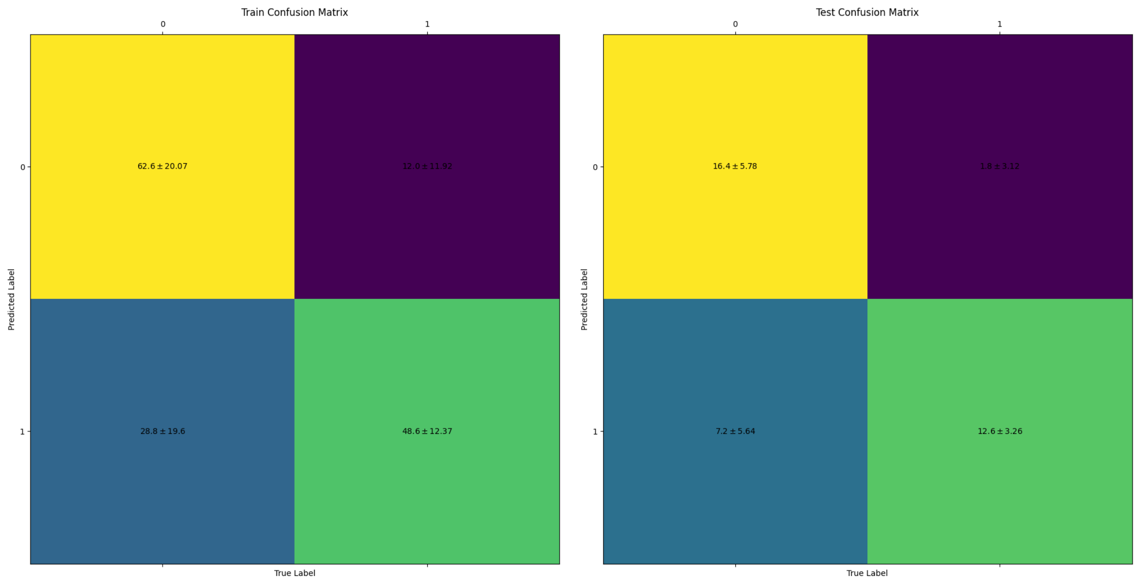


Figure A.24: Confusion Matrix with Mean and Standard Deviation Across Repeats

A.1.5 Amino Acid Properties

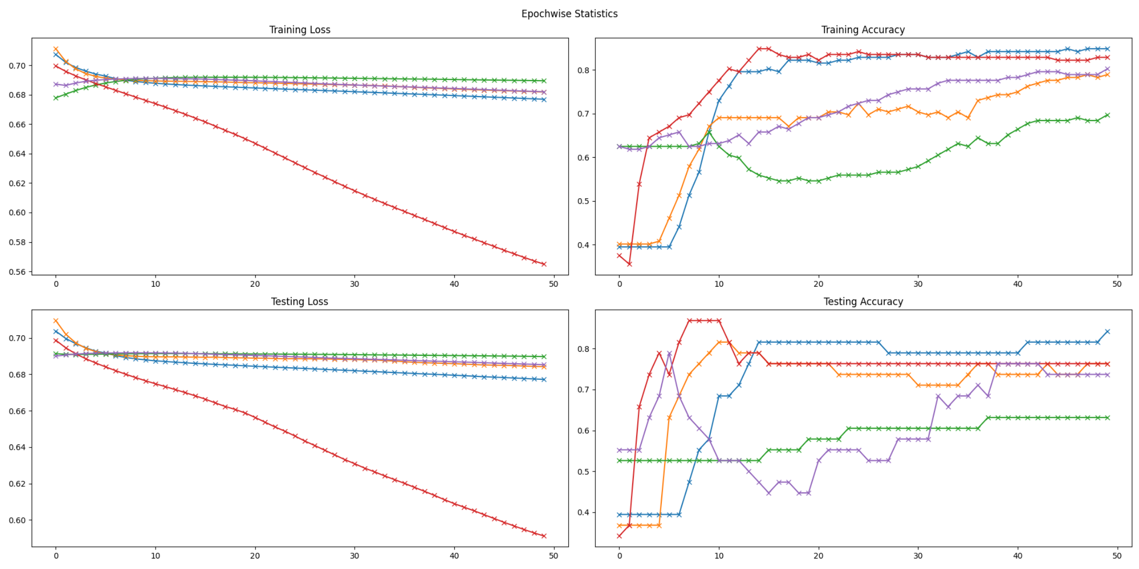


Figure A.25: Loss & Accuracy on Training and Test Sets for each Epoch

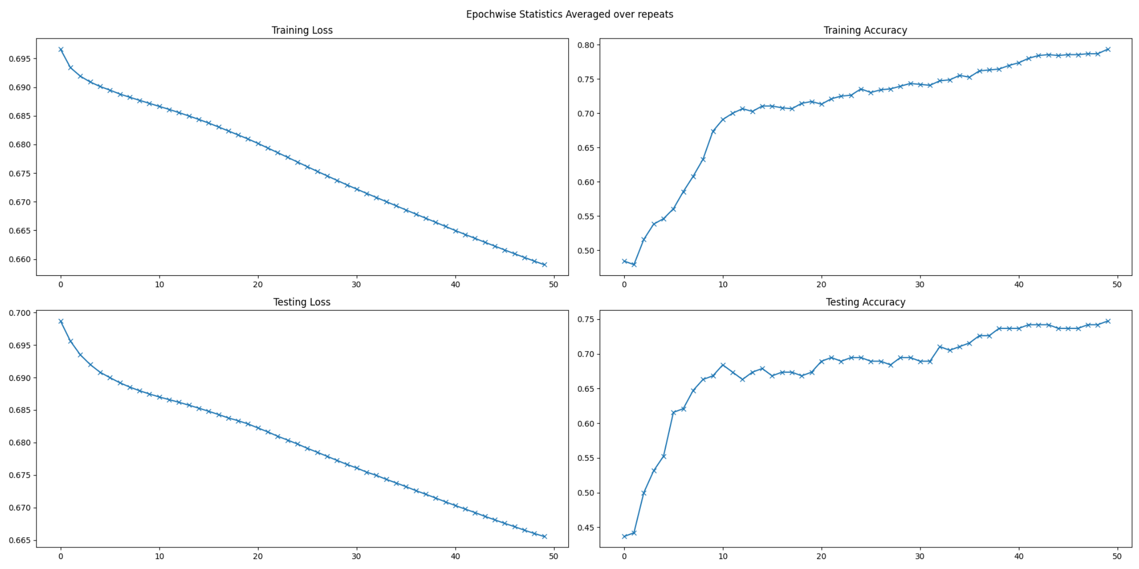


Figure A.26: Loss & Accuracy on Training and Test Sets Averaged on Repeats

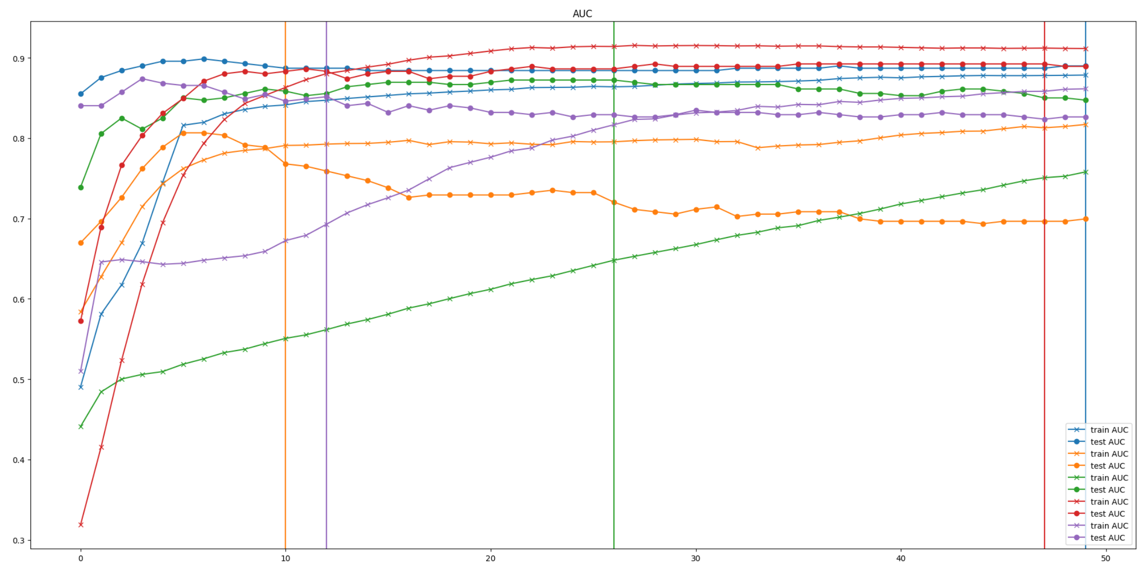


Figure A.27: AUC on Training and Testing Sets

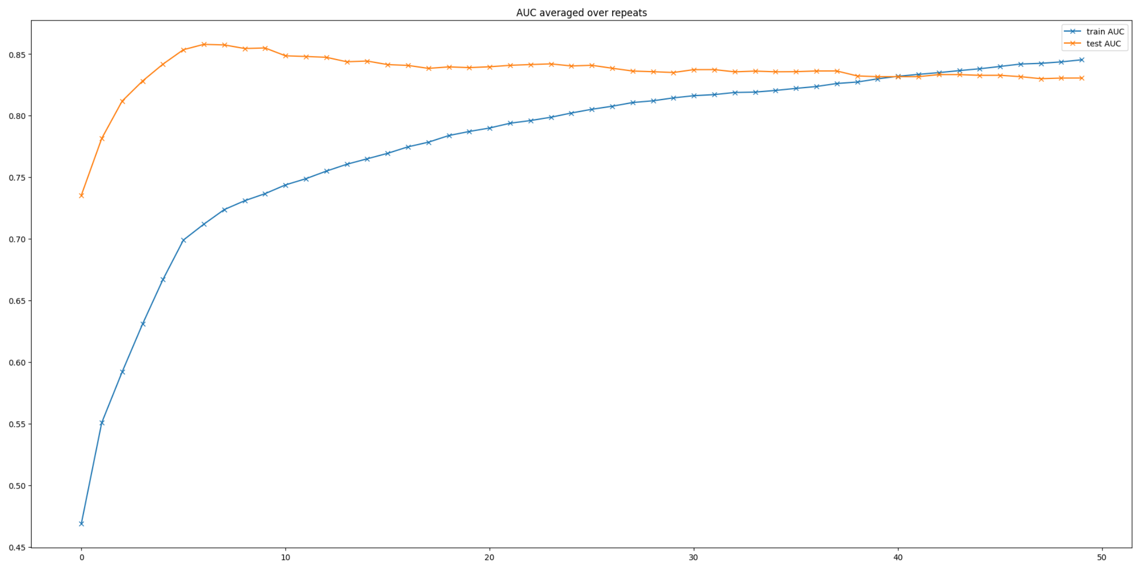


Figure A.28: AUC Averaged on Training and Test Sets

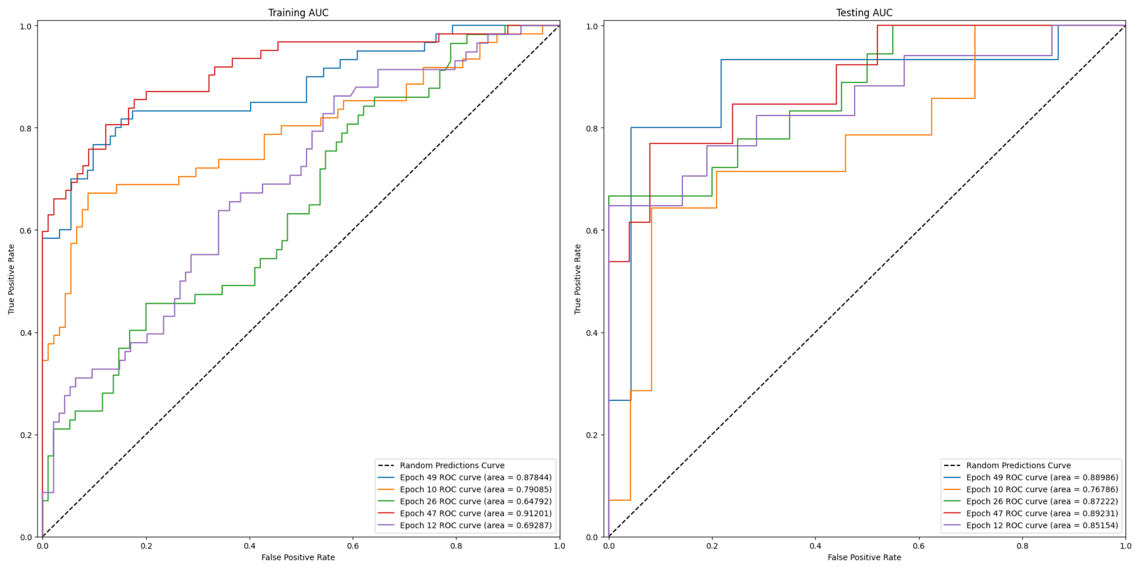


Figure A.29: AUC Curve for each repeated run

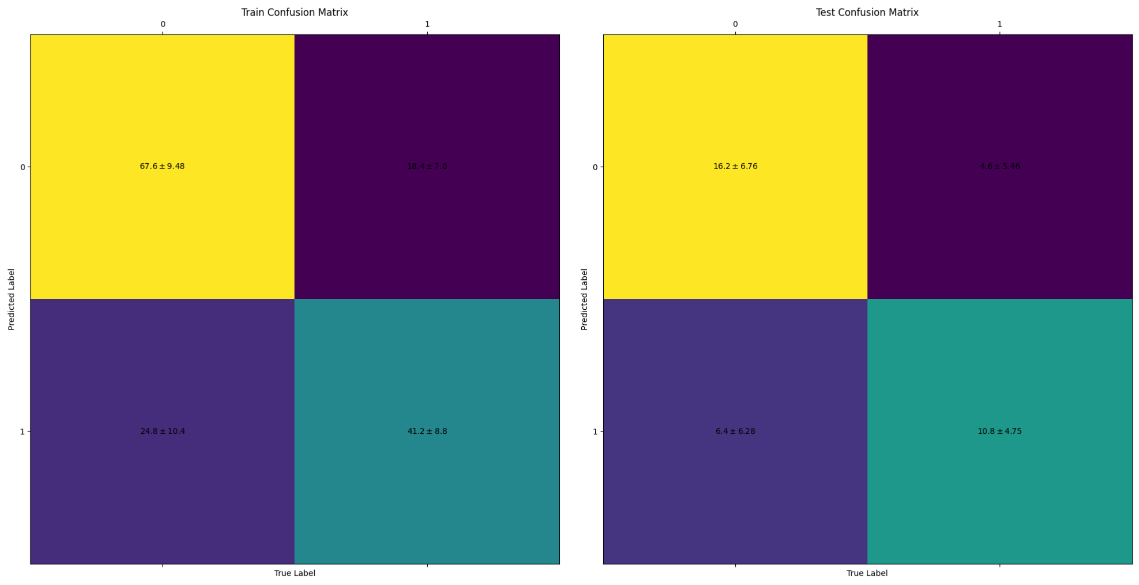


Figure A.30: Confusion Matrix with Mean and Standard Deviation Across Repeats

A.1.6 Random Embedding

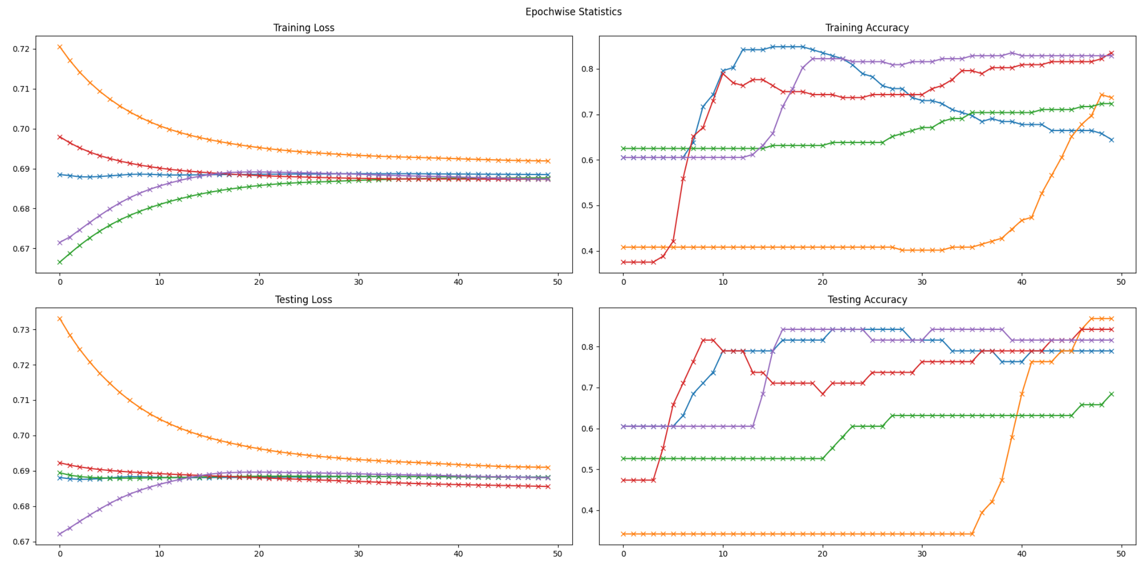


Figure A.31: Loss & Accuracy on Training and Test Sets for each Epoch

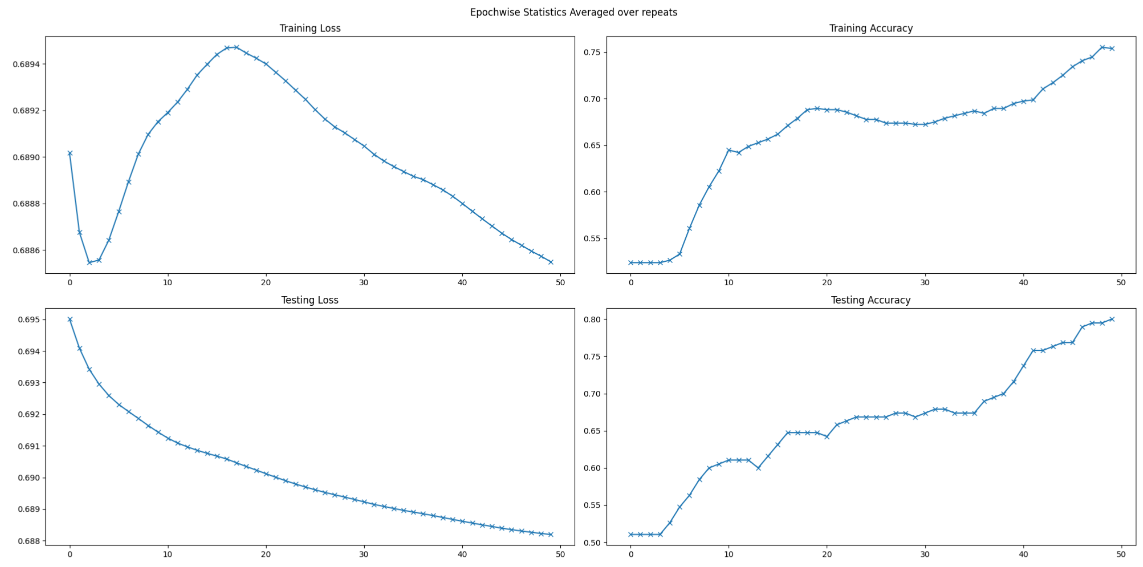


Figure A.32: Loss & Accuracy on Training and Test Sets Averaged on Repeats

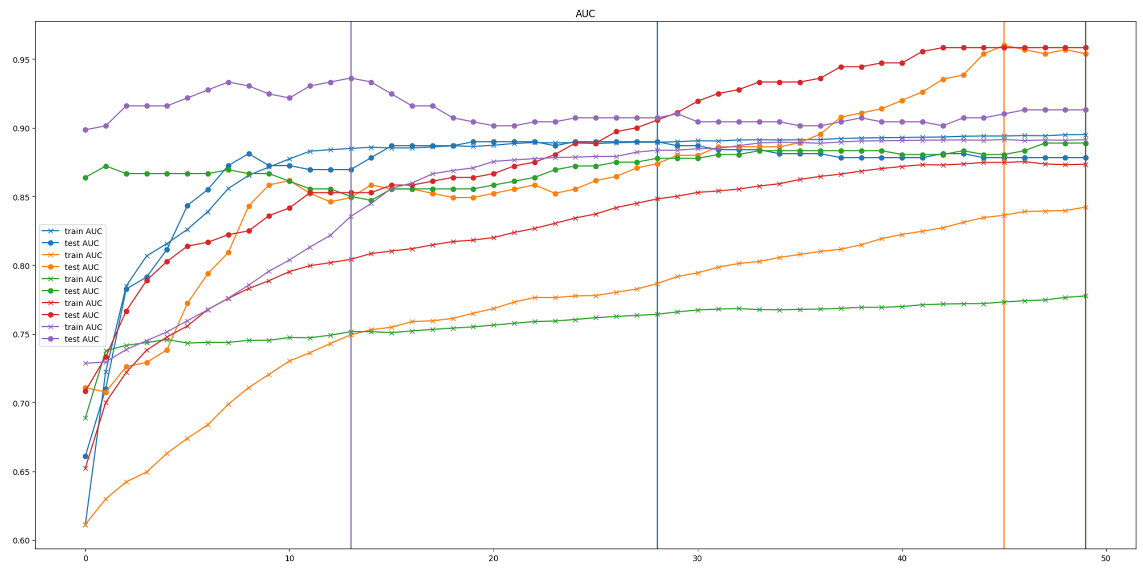


Figure A.33: AUC on Training and Testing Sets

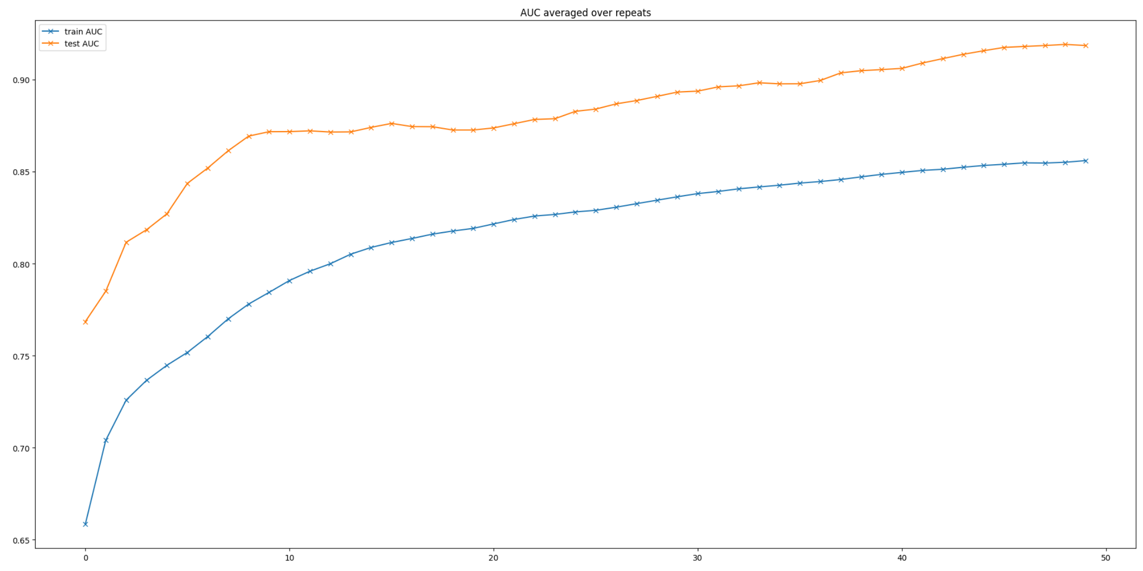


Figure A.34: AUC Averaged on Training and Test Sets

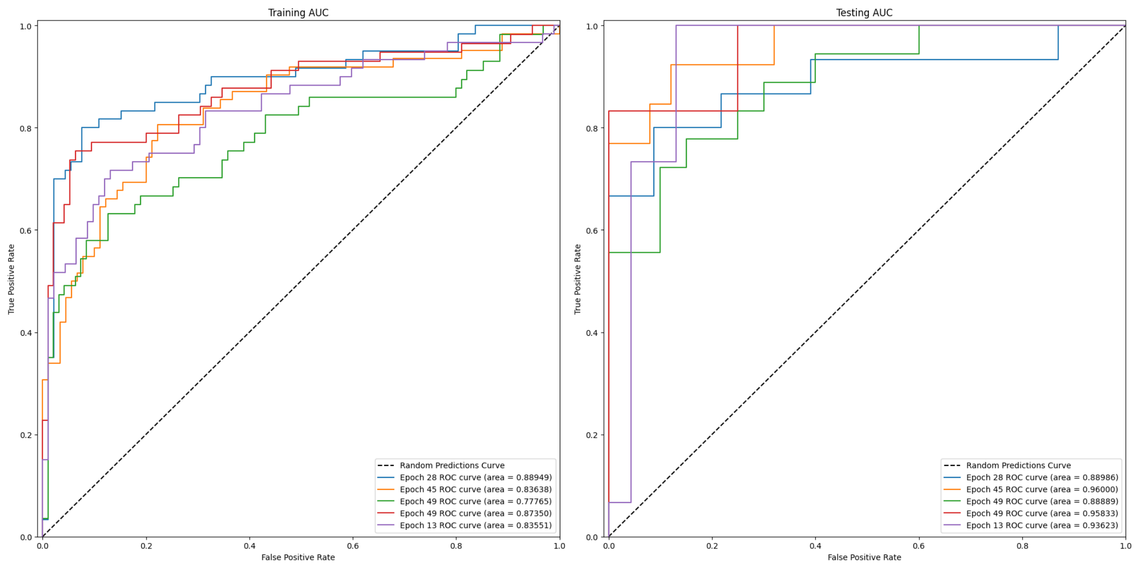


Figure A.35: AUC Curve for each repeated run

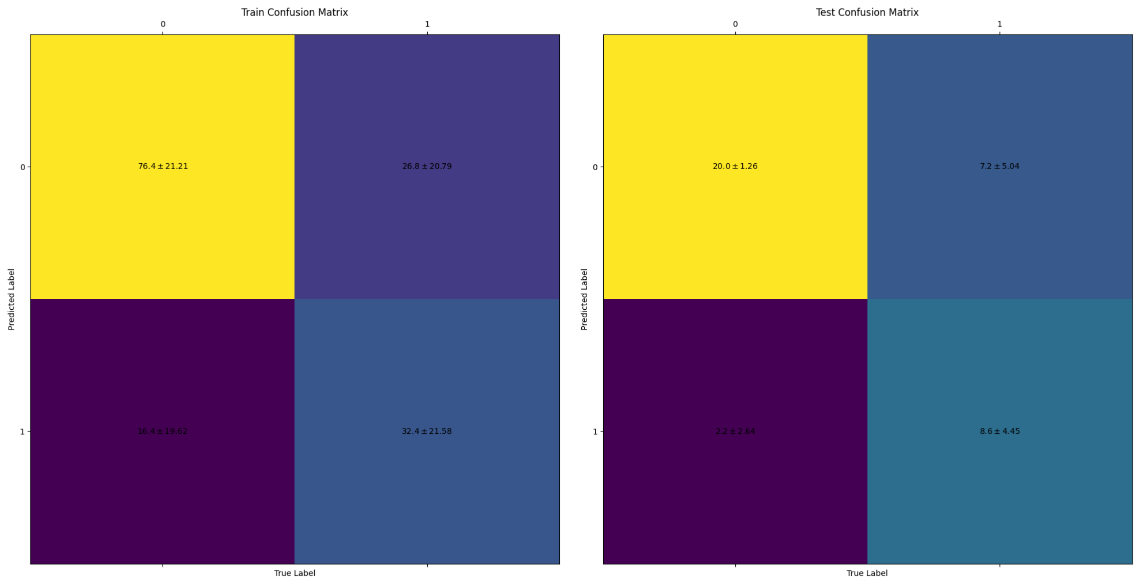


Figure A.36: Confusion Matrix with Mean and Standard Deviation Across Repeats

A.2 Evaluation Set (SCEPTR)

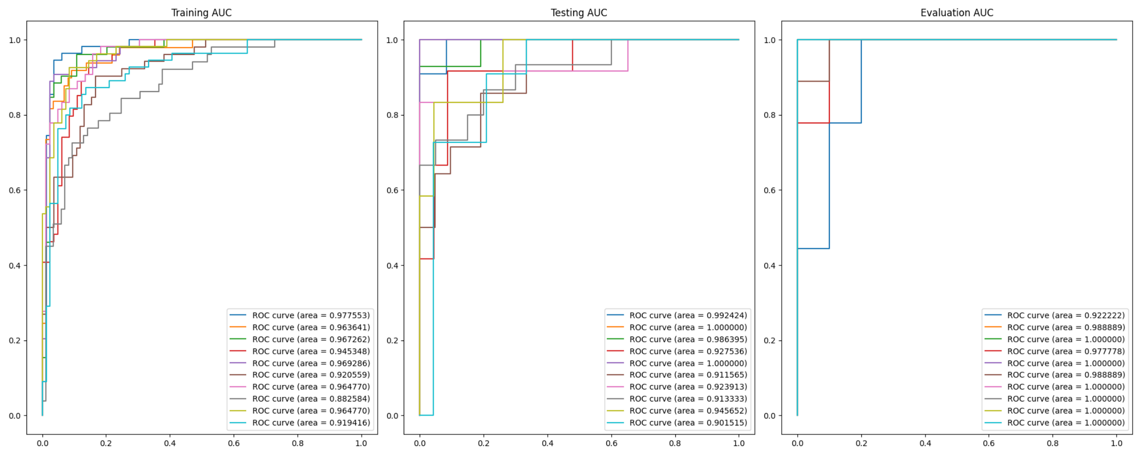


Figure A.37: AUC Curve with Evaluation Set

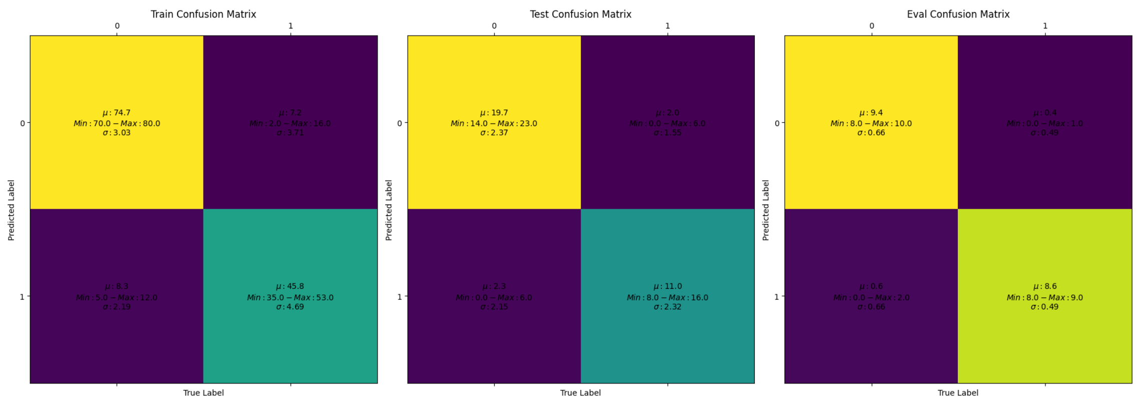


Figure A.38: Confusion Matrix with Evaluation Set

A.3 Interpretability

A.3.1 Positively Predictive TCRs

V/J Calls	Occurance
TRAV35	24
TRAV27	1
TRAV2	1
TRAV17	1
TRAV38-1	1
TRAV38-2/DV8	2
TRAV10	1
TRAV30	1
TRAJ54	19
TRAJ30	1
TRAJ32	4
TRAJ8	1
TRAJ44	1
TRAJ47	3
TRAJ42	1
TRAJ13	1
TRAJ29	1
TRBV2	292
TRBJ1-3	9
TRBJ2-2	73
TRBJ1-1	50
TRBJ2-4	15
TRBJ1-2	52
TRBJ2-3	33
TRBJ1-4	4
TRBJ2-7	19
TRBJ2-1	18
TRBJ2-5	19

Table A.2: V calls and J calls in TCRAB chains

V Call	J Call	CDR3	Duplicate Counts	Weighting Assigned
TRAV35	TRAJ32	CAGRGGATNKLIF	1	0.019446
TRAV35	TRAJ32	CAGRGGATNKLIF	1	0.006871
TRBV2	TRBJ2-2	CASRQGENTGELFF	1	0.082592
TRBV2	TRBJ2-2	CASRQGENTGELFF	1	0.085613
TRBV2	TRBJ1-2	CASSFRGGADEGYTF	16	0.043844
TRBV2	TRBJ1-2	CASSFRGGADEGYTF	32	0.034325
TRBV2	TRBJ2-2	CASSGPLLTGELFF	1	0.019762
TRBV2	TRBJ2-2	CASSGPLLTGELFF	1	0.022783
TRBV2	TRBJ2-2	CASNLGQGDTGELFF	1	0.049302
TRBV2	TRBJ2-2	CASNLGQGDTGELFF	1	0.036761
TRBV2	TRBJ2-2	CASGGTGETGELFF	7	0.040601
TRBV2	TRBJ2-2	CASGGTGETGELFF	1	0.028059
TRBV2	TRBJ1-2	CASRGLTGNYGYTF	12	0.040321
TRBV2	TRBJ1-2	CASRGLTGNYGYTF	2	0.043343
TRBV2	TRBJ1-2	CASRGLTGNYGYTF	1	0.042006
TRBV2	TRBJ1-2	CASRGLTGNYGYTF	11	0.030801

Table A.3: Expanded TCRs. Each row with the same V Call, J Call and CDR3 sequences is extracted from different patients

A.3.2 Negatively Predictive TCRs

V/J Call	Occurance
TRAV35	90
TRAJ54	66
TRAJ23	3
TRAJ47	12
TRAJ43	1
TRAJ57	1
TRAJ32	1
TRAJ44	1
TRAJ4	2
TRAJ37	2
TRAJ13	1
TRBV2	51
TRBJ2-2	23
TRBJ2-3	1
TRBJ1-1	12
TRBJ1-2	13
TRBJ2-1	1
TRBJ2-4	1

Table A.4: V calls and J calls in TCRAB chains

CDR3	Duplicate Counts	Weighting Assigned
CAFQGAQKLVF	1	0.04183
CAFQGAQKLVF	2	0.035601
CAVNGGAQKLVF	1	0.030562
CAVNGGAQKLVF	1	0.03274
CAVSGGNKLVF	1	0.030304
CAVSGGNKLVF	1	0.032482
CATGGAQKLVF	1	0.026204
CATGGAQKLVF	3	0.032521
CAVSGGAQKLVF	1	0.026099
CAVSGGAQKLVF	1	0.01987
CAFRGAQKLVF	1	0.025121
CAFRGAQKLVF	1	0.027299
CAGGAQKLVF	2	0.019724
CAGGAQKLVF	1	0.025414
CAMSGGAQKLVF	1	0.011908
CAMSGGAQKLVF	4	0.018225
CAYSGGAQKLVF	1	0.008719
CAYSGGAQKLVF	1	0.015036
CAVGAQKLVF	1	0.00428
CAVGAQKLVF	1	0.010596
CAGGGAQKLVF	1	0.045068
CAGGGAQKLVF	1	0.056986
CAVLGAQKLVF	1	0.035886
CAVLGAQKLVF	1	0.048431

CAVMGAQKLVF	1	0.02752
CAVMGAQKLVF	1	0.039438
CALGGAQKLVF	2	0.02337
CALGGAQKLVF	1	0.031777
CVIQGAQKLVF	1	0.021921
CVIQGAQKLVF	2	0.034466
CAMRGAQKLVF	1	0.003204
CAMRGAQKLVF	1	0.011611
CAVEGAQKLVF	1	0.045849
CAVEGAQKLVF	1	0.04936
CAVGGGGKLIF	1	0.01979
CAVGGGGKLIF	1	0.023928
CAPQGAQKLVF	1	0.008132
CAPQGAQKLVF	1	0.011643
CAAEGAQKLVF	2	0.005793
CAAEGAQKLVF	2	0.009931
CAARGAQKLVF	1	0.018378
CAARGAQKLVF	4	0.019005
CAVGRGAQKLVF	2	0.001662
CAVGRGAQKLVF	1	0.002289
CAGVQGAQKLVF	1	0.001227
CAGVQGAQKLVF	2	0.001853
CAGGGGAQKLVF	1	0.025808
CAGGGGAQKLVF	1	0.01958
CAGGGGAQKLVF	1	0.031498
CAVGYGNKLVF	2	0.02016
CAVGYGNKLVF	1	0.013932
CAVGYGNKLVF	1	0.02585
CAVGQGAQKLVF	1	0.015286
CAVGQGAQKLVF	1	0.009057
CAVGQGAQKLVF	4	0.017464
CAARGGAQKLVF	1	0.01278
CAARGGAQKLVF	1	0.014958
CAARGGAQKLVF	1	0.019097
CAVVGGNKLVF	2	0.010398
CAVVGGNKLVF	2	0.012576
CAVVGGNKLVF	1	0.016088
CAVGGYNKLIF	1	0.007118
CAVGGYNKLIF	2	0.000889
CAVGGYNKLIF	8	0.013435
CAMKGAQKLVF	2	0.00426
CAMKGAQKLVF	2	0.006439
CAMKGAQKLVF	1	0.010577
CAVGGFQKLVF	2	0.002212
CAVGGFQKLVF	1	0.007902

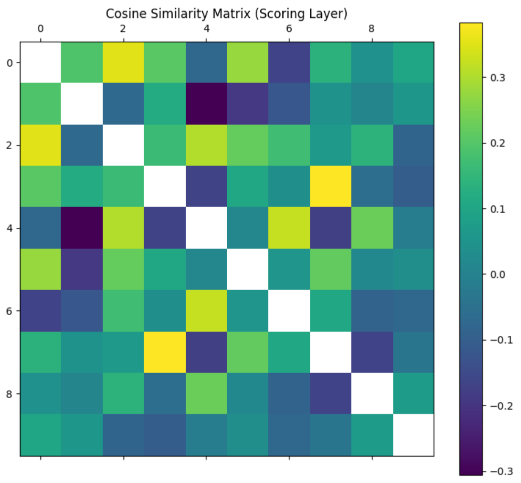
CAVGGFQKLVF	3	0.008528
CAVRGGAQKLVF	1	0.00992
CAVRGGAQKLVF	1	0.021838
CAVRGGAQKLVF	1	0.022465
CAEGAQKLVF	1	0.016712
CAEGAQKLVF	2	0.020224
CAEGAQKLVF	2	0.02085
CASQGAQKLVF	1	0.034101
CASQGAQKLVF	1	0.027873
CASQGAQKLVF	1	0.03628
CASQGAQKLVF	1	0.039791
CASQGAQKLVF	1	0.040418
CAASGGAQKLVF	2	0.021184
CAASGGAQKLVF	1	0.014956
CAASGGAQKLVF	2	0.023363
CAASGGAQKLVF	3	0.026874
CAASGGAQKLVF	2	0.027501
CAVKGAQKLVF	1	0.028551
CAVKGAQKLVF	2	0.022322
CAVKGAQKLVF	1	0.030729
CAVKGAQKLVF	3	0.034868
CAAQGAQKLVF	2	0.011213
CAAQGAQKLVF	2	0.01962
CAAQGAQKLVF	2	0.023132
CAAQGAQKLVF	2	0.023758
CAVGGNKLVF	4	0.005764
CAVGGNKLVF	1	0.014171
CAVGGNKLVF	4	0.017682
CAVGGNKLVF	1	0.018309
CAVQGAQKLVF	1	0.048805
CAVQGAQKLVF	4	0.042576
CAVQGAQKLVF	1	0.050983
CAVQGAQKLVF	1	0.054495
CAVQGAQKLVF	1	0.055121

Table A.5: Expanded TCRs (with only alpha chain CDR3 sequences). Each row with the same VDR3 sequences is sampled from different patients.

V Call	J Call	CDR3	Duplicate Counts	Weightings Assigned
TRAV35	TRAJ54	CAVQGAQKLVF	1	0.121364
TRAV35	TRAJ54	CAVQGAQKLVF	1	0.097269
TRAV35	TRAJ54	CAGQRGAQKLVF	2	0.06756
TRAV35	TRAJ54	CAGQRGAQKLVF	3	0.065152
TRAV35	TRAJ54	CAGQRGAQKLVF	3	0.061006
TRAV35	TRAJ47	CAGQYGNKLVF	1	0.02936
TRAV35	TRAJ47	CAGQYGNKLVF	4	0.084459
TRAV35	TRAJ54	CAGLQGAQKLVF	1	0.02247
TRAV35	TRAJ54	CAGLQGAQKLVF	2	0.042526
TRAV35	TRAJ54	CAGLQGAQKLVF	1	0.019193
TRAV35	TRAJ54	CAGQGGAQKLVF	1	0.098451
TRAV35	TRAJ54	CAGQGGAQKLVF	2	0.098451
TRAV35	TRAJ54	CAGQGGAQKLVF	1	0.074357
TRAV35	TRAJ47	CAGEYGNKLVF	6	0.038398
TRAV35	TRAJ47	CAGEYGNKLVF	1	0.016374
TRAV35	TRAJ47	CAGEYGNKLVF	1	0.014304
TRAV35	TRAJ54	CAGRGGAQKLVF	1	0.067308
TRAV35	TRAJ54	CAGRGGAQKLVF	1	0.066102
TRAV35	TRAJ54	CAGRGGAQKLVF	1	0.064032
TRAV35	TRAJ54	CAGLYGAQKLVF	1	0.04465
TRAV35	TRAJ54	CAGLYGAQKLVF	1	0.022625
TRAV35	TRAJ54	CAGQKGAQKLVF	1	0.041444
TRAV35	TRAJ54	CAGQKGAQKLVF	1	0.020626
TRAV35	TRAJ54	CAGIQGAQKLVF	1	0.040243
TRAV35	TRAJ54	CAGIQGAQKLVF	1	0.039036
TRAV35	TRAJ54	CAGQLGAQKLVF	1	0.036714
TRAV35	TRAJ54	CAGQLGAQKLVF	1	0.035507
TRAV35	TRAJ54	CAGRRGAQKLVF	1	0.0356
TRAV35	TRAJ54	CAGRRGAQKLVF	1	0.032323
TRAV35	TRAJ54	CAGQEGAQKLVF	2	0.009764
TRAV35	TRAJ54	CAGQEGAQKLVF	1	0.008558
TRAV35	TRAJ54	CAGLRGAQKLVF	1	0.002284
TRAV35	TRAJ54	CAGLRGAQKLVF	1	0.001078
TRAV35	TRAJ54	CAGPRGAQKLVF	1	0.012101
TRAV35	TRAJ54	CAGPRGAQKLVF	2	0.010031
TRAV35	TRAJ54	CAGQGAQKLVF	3	0.09762
TRAV35	TRAJ54	CAGQGAQKLVF	3	0.096413
TRAV35	TRAJ54	CAGQGAQKLVF	3	0.188686
x	x	CAGQGAQKLVF	2	0.037973
TRAV35	TRAJ54	CANQGAQKLVF	1	0.071774
x	x	CANQGAQKLVF	1	0.012621
TRAV35	TRAJ54	CAGRGAQKLVF	1	0.088424
x	x	CAGRGAQKLVF	1	0.035873
TRAV35	TRAJ54	CAGEGGAQKLVF	1	0.055349
x	x	CAGEGGAQKLVF	1	0.014777
TRAV35	TRAJ54	CAGEGAQKLVF	1	0.102333
x	x	CAGEGAQKLVF	1	0.034023
x	x	CAGEGAQKLVF	1	0.04243
TRAV35	TRAJ54	CAIQGAQKLVF	1	0.07486
x	x	CAIQGAQKLVF	3	0.019715
x	x	CAIQGAQKLVF	1	0.013487

Table A.6: Expanded TCRs. Each row with the same V Call, J Call and CDR3 sequences is extracted from different patients

A.3.3 Similarity



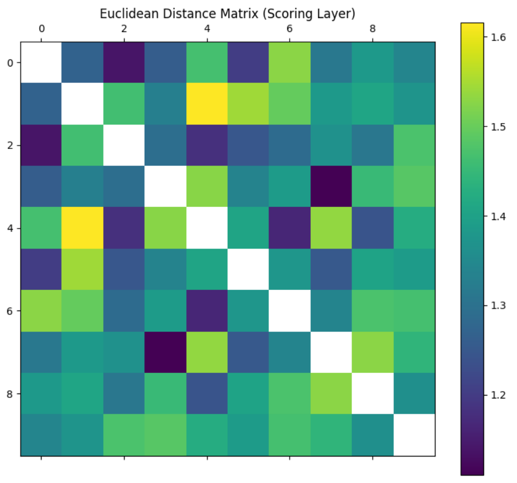
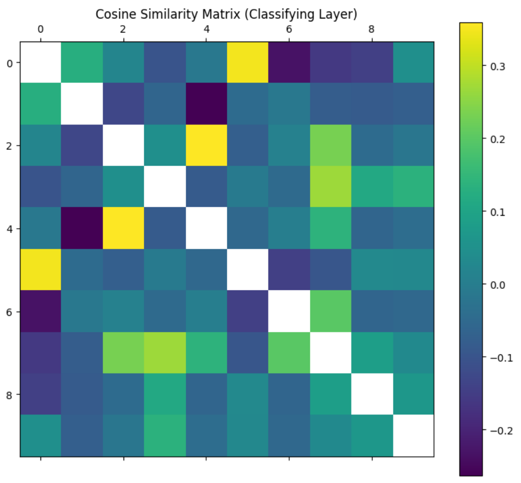


Figure A.39: Similarity between different instances' scoring layer's weights



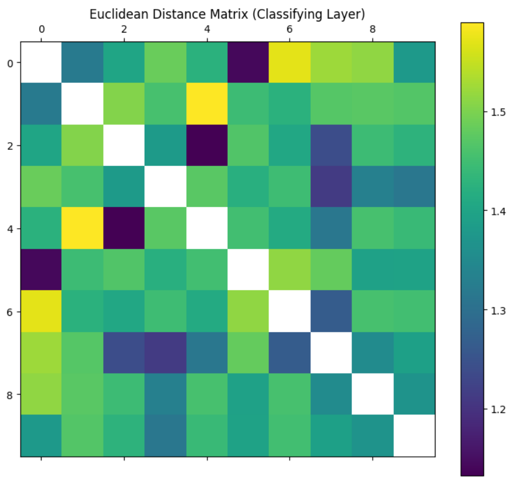


Figure A.40: Similarity between different instances' classifying layer's weights

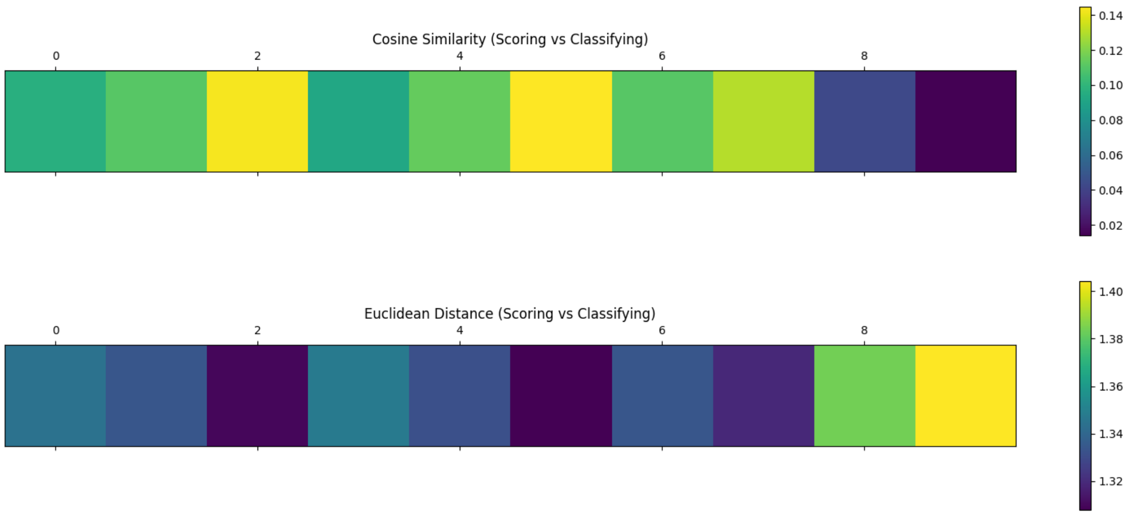


Figure A.41: Similarity between same instances' scoring and classifying layer's weights

Appendix B

GitHub Repository

The code for this project and all experimental results are available on a public GitHub repository. This repository is under the MIT License, where instructions on how to use the repository, including information on how to install the virtual environment, are in the `README.md`

`https://github.com/RcwYuen/TCR-Cancer-Prediction`

Bibliography

[1] K. E. Hellstrom and I. Hellstrom, “From the hellstrom paradox toward cancer cure,” *Progress in Molecular Biology and Translational Science*, p. 1–24, 2019.

[2] F. H. Igney and P. H. Krammer, “Death and anti-death: Tumour resistance to apoptosis,” *Nature Reviews Cancer*, vol. 2, p. 277–288, Apr 2002.

[3] M. Roser and H. Ritchie, “Cancer, https://ourworldindata.org/cancer#cancer-is-one-of-the-leading-causes-of-death,” Jul 2015.

[4] S. McPhail, S. Johnson, D. Greenberg, M. Peake, and B. Rous, “Stage at diagnosis and early mortality from cancer in england,” *British Journal of Cancer*, vol. 112, Mar 2015.

[5] R. S. Sealfon, A. K. Wong, and O. G. Troyanskaya, “Machine learning methods to model multicellular complexity and tissue specificity,” *Nature Reviews Materials*, vol. 6, no. 8, p. 717–729, 2021.

[6] B. Hunter, S. Hindocha, and R. W. Lee, “The role of artificial intelligence in early cancer diagnosis,” *Cancers*, vol. 14, no. 6, p. 1524, 2022.

[7] L. Nguyen, A. V. Hoeck, and E. Cuppen, “Machine learning-based tissue of origin classification for cancer of unknown primary diagnostics using genome-wide mutation features,” *Nature Communications*, vol. 13, no. 4013, 2022.

[8] I. Moon, J. LoPiccolo, and A. Gusev, “Machine learning for improved clinical management of cancers of unknown primary,” *Nature Medicine*, vol. 29, pp. 1920–1921, 2023.

[9] P. Freitas, F. Silva, J. V. Sousa, R. M. Ferreira, C. Figueiredo, T. Pereira, and H. P. Oliveira, “Machine learning-based approaches for cancer prediction using microbiome data,” *Scientific Reports*, vol. 13, no. 1, 2023.

[10] G. Menna, G. Piaser Guerrato, L. Bilgin, G. M. Ceccarelli, A. Olivi, and G. M. Della Pepa, “Is there a role for machine learning in liquid biopsy for brain tumors? a systematic review,” *International Journal of Molecular Sciences*, vol. 24, no. 11, p. 9723, 2023.

[11] S. Connal, J. M. Cameron, A. Sala, P. M. Brennan, D. S. Palmer, J. D. Palmer, H. Perlow, and M. J. Baker, “Liquid biopsies: The future of cancer early detection,” *Journal of Translational Medicine*, vol. 21, no. 1, 2023.

[12] L. Liu, X. Chen, O. O. Petinrin, W. Zhang, S. Rahaman, Z.-R. Tang, and K.-C. Wong, “Machine learning protocols in early cancer detection based on liquid biopsy: A survey,” *Life*, vol. 11, no. 7, p. 638, 2021.

[13] Y. Said, A. A. Alsheikhy, T. Shawly, and H. Lahza, “Medical images segmentation for lung cancer diagnosis based on deep learning architectures,” *Diagnostics*, vol. 13, no. 3, p. 546, 2023.

[14] X. Jiang, Z. Hu, S. Wang, and Y. Zhang, “Deep learning for medical image-based cancer diagnosis,” *Cancers*, vol. 15, no. 14, p. 3608, 2023.

[15] D. Beshnova, J. Ye, O. Onabolu, B. Moon, W. Zheng, Y.-X. Fu, J. Brugarolas, J. Lea, and B. Li, “De novo prediction of cancer-associated t cell receptors for noninvasive cancer detection,” *Science Translational Medicine*, vol. 12, no. 557, 2020.

[16] M. Cavanagh and E. Gwyer Findlay, “T-cell activation.” British Society for Immunology, 2023.

[17] M. L. Russell, A. Souquette, D. M. Levine, S. A. Schattgen, E. K. Allen, G. Kuan, N. Simon, A. Balmaseda, A. Gordon, P. G. Thomas, and et al., “Combining genotypes and t cell receptor distributions to infer genetic loci determining v(d)j recombination probabilities,” *eLife*, vol. 11, 2022.

[18] LibreTexts, “20.7a: Clonal selection and t-cell differentiation.” Medicine LibreTexts, 2023.

[19] S. Valpione, P. A. Mundra, E. Galvani, L. G. Campana, P. Lorigan, F. De Rosa, A. Gupta, J. Weightman, S. Mills, N. Dhomen, and et al., “The t cell receptor repertoire of tumor infiltrating t cells is predictive and prognostic for cancer survival,” *Nature Communications*, vol. 12, no. 1, 2021.

[20] Z. Sethna, G. Isacchini, T. Dupic, T. Mora, A. M. Walczak, and Y. Elhanati, “Population variability in the generation and selection of t-cell repertoires,” *PLOS Computational Biology*, vol. 16, no. 12, 2020.

[21] D. S. Bortone, M. G. Woodcock, J. S. Parker, and B. G. Vincent, “Improved t-cell receptor diversity estimates associate with survival and response to anti–pd-1 therapy,” *Cancer Immunology Research*, vol. 9, no. 1, p. 103–112, 2021.

[22] M. Li, C. Zhang, S. Deng, L. Li, S. Liu, J. Bai, Y. Xu, Y. Guan, X. Xia, L. Sun, and et al., “Lung cancer-associated t cell repertoire as potential biomarker for early detection of stage i lung cancer,” *Lung Cancer*, vol. 162, p. 16–22, Sep 2021.

[23] Y. Kim, T. Wang, D. Xiong, X. Wang, and S. Park, “Multiple instance neural networks based on sparse attention for cancer detection using t-cell receptor sequences,” *BMC Bioinformatics*, vol. 23, no. 1, 2022.

[24] J. Ostmeyer, S. Christley, I. T. Toby, and L. G. Cowell, “Biophysicochemical motifs in t-cell receptor sequences distinguish repertoires from tumor-infiltrating lymphocyte and adjacent healthy tissue,” *Cancer Research*, vol. 79, no. 7, p. 1671–1680, 2019.

[25] W. R. Atchley, J. Zhao, A. D. Fernandes, and T. Drüke, “Solving the protein sequence metric problem,” *Proceedings of the National Academy of Sciences*, vol. 102, p. 6395–6400, Apr 2005.

[26] A. Kidera, Y. Konishi, M. Oka, T. Ooi, and H. A. Scheraga, “Statistical analysis of the physical properties of the 20 naturally occurring amino acids,” *Journal of Protein Chemistry*, vol. 4, p. 23–55, Feb 1985.

[27] Y. Elhanati, Z. Sethna, Q. Marcou, C. G. Callan, T. Mora, and A. M. Walczak, “Inferring processes underlying b-cell repertoire diversity,” *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 370, p. 20140243, Sept. 2015.

[28] L. Li, Y. Nagano, and B. Chain, “Identifying conserved tcr beta chain motifs through a data-driven approach: A novel approach using sentencepiece.” UCL Dissertations (Internal), Apr 2023.

[29] A. Madani, B. Krause, E. R. Greene, S. Subramanian, B. P. Mohr, J. M. Holton, J. L. Olmos, C. Xiong, Z. Z. Sun, R. Socher, and et al., “Large language models generate functional protein sequences across diverse families,” *Nature Biotechnology*, vol. 41, no. 8, p. 1099–1106, 2023.

[30] N. Ferruz, S. Schmidt, and B. Höcker, “Protgpt2 is a deep unsupervised language model for protein design,” *Nature Communications*, vol. 13, no. 1, 2022.

[31] M. E. Zaslavsky, E. Craig, J. K. Michuda, N. Ram-Mohan, J.-Y. Lee, K. D. Nguyen, R. A. Hoh, T. D. Pham, E. S. Parsons, S. R. Macwana, and et al., “Disease diagnostics using machine learning of immune receptors,” *Disease diagnostics using machine learning of immune receptors*, Apr 2022.

[32] K. Wu, K. E. Yost, B. Daniel, J. A. Belk, Y. Xia, T. Egawa, A. Satpathy, H. Y. Chang, and J. Zou, “Tcr-bert: learning the grammar of t-cell receptors for flexible antigen-xbinding analyses,” *TCR-Bert: Learning the grammar of T-cell receptors for flexible antigen-xbinding analyses*, Nov 2021.

[33] J. Glanville, H. Huang, A. Nau, O. Hatton, L. E. Wagar, F. Rubelt, X. Ji, A. Han, S. M. Krams, C. Pettus, and et al., “Identifying specificity groups in the t cell receptor repertoire,” *Nature*, vol. 547, no. 7661, p. 94–98, 2017.

[34] T. Kudo and J. Richardson, “Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing,” 2018.

[35] D. Ofer, N. Brandes, and M. Linial, “The language of proteins: Nlp, machine learning & protein sequences,” *Computational and Structural Biotechnology Journal*, vol. 19, pp. 1750–1758, 2021.

[36] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, and et al., “Highly accurate protein structure prediction with alphafold,” *Nature*, vol. 596, no. 7873, p. 583–589, 2021.

[37] J. H. Wilson and T. Hunt, *Molecular biology of the cell, 4th edition: A problems approach.* Garland Science, 2002.

[38] N. Thomas, K. Best, M. Cinelli, S. Reich-Zeliger, H. Gal, E. Shifrut, A. Madi, N. Friedman, J. Shawe-Taylor, and B. Chain, “Tracking global changes induced in the cd4 t cell receptor repertoire by immunization with a complex antigen using short stretches of cdr3 protein sequence.,” *Bioinformatics*, vol. 30, p. 3181–3188, Nov 2014.

[39] K. Joshi, M. R. de Massy, M. Ismail, J. L. Reading, I. Uddin, A. Woolston, E. Hatipoglu, T. Oakes, R. Rosenthal, T. Peacock, and et al., “Spatial heterogeneity of the t cell receptor repertoire reflects the mutational landscape in lung cancer,” *Nature Medicine*, vol. 25, p. 1549–1559, Oct 2019.

[40] C. Krishna, D. Chowell, M. Gönen, Y. Elhanati, and T. A. Chan, "Genetic and environmental determinants of human tcr repertoire diversity," *Immunity &amp; Ageing*, vol. 17, Sep 2020.

[41] Z. Zhang, Y. Feng, S. Ying, and Y. Gao, "Deep hypergraph structure learning," 2022.

[42] C. Gote, V. Perri, and I. Scholtes, "Predicting influential higher-order patterns in temporal network data," 2022.

[43] S. Kawashima, H. Ogata, and M. Kanehisa, "Aaindex: Amino acid index database," *Nucleic Acids Research*, vol. 27, p. 368–369, Jan 1999.

[44] Z. Zhang, D. Xiong, X. Wang, H. Liu, and T. Wang, "Mapping the functional landscape of t cell receptor repertoires by single-t cell transcriptomics," *Nature Methods*, vol. 18, p. 92–99, Jan 2021.

[45] F. Bieberich and S. T. Reddy, "The unexpected benefit of tcr cross-reactivity in cancer immunotherapy," *Cancer Research*, vol. 83, no. 19, p. 3168–3169, 2023.

[46] A. F. T. Martins and R. F. Astudillo, "From softmax to sparsemax: A sparse model of attention and multi-label classification," 2016.

[47] Z. Zhang, D. Xiong, X. Wang, H. Liu, and T. Wang, "Mapping the functional landscape of t cell receptor repertoires by single-t cell transcriptomics," *Nature Methods*, vol. 18, p. 92–99, Jan 2021.

[48] OpenAI, "Chat generative pre-trained transformer (chatgpt)." `https://openai.com/blog/chatgpt`, 2022. Accessed: 2024-01-10.

[49] V. Ramanujan, T. Nguyen, S. Oh, L. Schmidt, and A. Farhadi, "On the connection between pre-training data diversity and fine-tuning robustness," 2023.

[50] Z. Liu, Y. Xu, Y. Xu, Q. Qian, H. Li, X. Ji, A. Chan, and R. Jin, "Improved fine-tuning by better leveraging pre-training data," 2022.

[51] N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, S. Hu, Y. Chen, C.-M. Chan, W. Chen, and et al., "Parameter-efficient fine-tuning of large-scale pre-trained language models," *Nature Machine Intelligence*, vol. 5, no. 3, p. 220–235, 2023.

[52] K. You, Y. Liu, Z. Zhang, J. Wang, M. I. Jordan, and M. Long, "Ranking and tuning pre-trained models: A new paradigm for exploiting model hubs," 2022.

[53] Z. Chen, W. K. Wong, Z. Zhong, J. Liao, and Y. Qu, "Effective transfer of pretrained large visual model for fabric defect segmentation via specifc knowledge injection," 2023.

[54] V. Srinivasan, N. Strodthoff, J. Ma, A. Binder, K.-R. Müller, and W. Samek, "To pretrain or not? a systematic analysis of the benefits of pretraining in diabetic retinopathy," *PLOS ONE*, vol. 17, no. 10, 2022.

[55] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet,

T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, "Llama 2: Open foundation and fine-tuned chat models," 2023.

[56] Y. Li, Z. Li, K. Zhang, R. Dan, S. Jiang, and Y. Zhang, "Chatdoctor: A medical chat model fine-tuned on a large language model meta-ai (llama) using medical domain knowledge," 2023.

[57] Google DeepMind, "Gemini: A family of highly capable multimodal models," Technical Report, Google DeepMind, December 2023. [Online; accessed 7-December-2023].

[58] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[59] M. Shugay, D. V. Bagaev, I. V. Zvyagin, R. M. Vroomans, J. C. Crawford, G. Dolton, E. A. Komech, A. L. Sycheva, A. E. Koneva, E. S. Egorov, and et al., "Vdjdb: A curated database of t-cell receptor sequences with known antigen specificity," *Nucleic Acids Research*, vol. 46, no. D1, 2017.

[60] W. Zhang, L. Wang, K. Liu, X. Wei, K. Yang, W. Du, S. Wang, N. Guo, C. Ma, L. Luo, and et al., "Pird: Pan immune repertoire database," *Bioinformatics*, vol. 36, no. 3, p. 897–903, 2019.

[61] Z. Lin, H. Akin, R. Rao, B. Hie, Z. Zhu, W. Lu, N. Smetanin, R. Verkuil, O. Kabeli, Y. Shmueli, and et al., "Evolutionary-scale prediction of atomic-level protein structure with a language model," *Science*, vol. 379, no. 6637, p. 1123–1130, 2023.

[62] R. Rao, N. Bhattacharya, N. Thomas, Y. Duan, X. Chen, J. Canny, P. Abbeel, and Y. S. Song, "Evaluating protein transfer learning with tape," 2019.

[63] V. A. Traag, L. Waltman, and N. J. van Eck, "From louvain to leiden: guaranteeing well-connected communities," *Scientific Reports*, vol. 9, Mar. 2019.

[64] M. Jamal-Hanjani, A. Hackshaw, Y. Ngai, J. Shaw, C. Dive, S. Quezada, G. Middleton, E. de Bruin, J. Le Quesne, S. Shafi, and et al., "Tracking genomic cancer evolution for precision medicine: The lung tracerx study," *PLoS Biology*, vol. 12, Jul 2014.

[65] M. Milighetti, Y. Peng, C. Tan, M. Mark, G. Nageswaran, S. Byrne, T. Ronel, T. Peacock, A. Mayer, A. Chandran, and et al., "Large clones of pre-existing t cells drive early immunity against sars-cov-2 and lcmv infection," *iScience*, vol. 26, p. 106937, Jun 2023.

[66] E. Shaw, "Variation in t cell immunity in health," Dec 2022.

[67] Y. Nagano and B. Chain, "tidytcells: standardizer for tr/mh nomenclature," *Frontiers in Immunology*, vol. 14, 2023.

[68] E. Rosati, C. M. Dowds, E. Liaskou, E. K. Henriksen, T. H. Karlsen, and A. Franke, "Overview of methodologies for t-cell receptor repertoire analysis," *BMC Biotechnology*, vol. 17, Jul 2017.

[69] H. Tanno, T. M. Gould, J. R. McDaniel, W. Cao, Y. Tanno, R. E. Durrett, D. Park, S. J. Cate, W. H. Hildebrand, C. L. Dekker, and et al., "Determinants governing t cell recep-

tor alpha/beta-chain pairing in repertoire formation of identical twins," *Proceedings of the National Academy of Sciences*, vol. 117, p. 532–540, Dec 2019.

[70] Y. Huang, J. Xu, Z. Jiang, J. Lai, Z. Li, Y. Yao, T. Chen, L. Yang, Z. Xin, and X. Ma, "Advancing transformer architecture in long-context large language models: A comprehensive survey," 2023.

[71] J. Pfeiffer, I. Vulić, I. Gurevych, and S. Ruder, "MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (B. Webber, T. Cohn, Y. He, and Y. Liu, eds.), (Online), pp. 7654–7673, Association for Computational Linguistics, Nov. 2020.

[72] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, eds.), (Online), pp. 8440–8451, Association for Computational Linguistics, July 2020.

[73] A. Ansell, E. M. Ponti, A. Korhonen, and I. Vulić, "Composable sparse fine-tuning for cross-lingual transfer," 2023.

[74] Y. Chen, K. Marchisio, R. Raileanu, D. I. Adelani, P. Stenetorp, S. Riedel, and M. Artetxe, "Improving language plasticity via pretraining with active forgetting," 2024.

[75] B. U. Tayyab and N. Chua, "Pre-training transformers for domain adaptation," 2021.

[76] D. Hendrycks, K. Lee, and M. Mazeika, "Using pre-training can improve model robustness and uncertainty," 2019.

[77] D. Kim, K. Wang, S. Sclaroff, and K. Saenko, "A broad study of pre-training for domain generalization and adaptation," 2022.

[78] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith, "Don't stop pretraining: Adapt language models to domains and tasks," 2020.

[79] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," 2014.

[80] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[81] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," 2013.

[82] N. F. Liu, M. Gardner, Y. Belinkov, M. E. Peters, and N. A. Smith, "Linguistic knowledge and transferability of contextual representations," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (J. Burstein, C. Doran, and T. Solorio, eds.), (Minneapolis, Minnesota), pp. 1073–1094, Association for Computational Linguistics, June 2019.

[83] A. Tamkin, T. Singh, D. Giovanardi, and N. Goodman, "Investigating transferability in pretrained language models," in *Findings of the Association for Computational Linguistics:*

EMNLP 2020 (T. Cohn, Y. He, and Y. Liu, eds.), (Online), pp. 1393–1401, Association for Computational Linguistics, Nov. 2020.

[84] H. Zhao, H. Chen, F. Yang, N. Liu, H. Deng, H. Cai, S. Wang, D. Yin, and M. Du, "Explainability for large language models: A survey," 2023.

[85] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[86] T. Huang, Y. Shu, and Y.-D. Cai, "Genetic differences among ethnic groups," *BMC Genomics*, vol. 16, Dec 2015.

[87] A. Sharma-Oates, D. T. Zemedikun, K. Kumar, J. A. Reynolds, A. Jain, K. Raza, J. A. Williams, L. Bravo, V. R. Cardoso, G. Gkoutos, and et al., "Early onset of immune-mediated diseases in minority ethnic groups in the uk," *BMC Medicine*, vol. 20, Oct 2022.

[88] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021.

[89] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," 2023.

[90] National Cancer Institute, "Age associated risk with cancer; https://www.cancer.gov/about-cancer/causes-prevention/risk/age," Mar 2021.

[91] United Kingdom Cancer Research, "Family history and inherited cancer genes; https://www.cancerresearchuk.org/about-cancer/causes-of-cancer/inherited-cancer-genes-and-increased-cancer-risk/family-history-and-inherited-cancer-genes," Nov 2021.

[92] W. B. Johnson and J. Lindenstrauss, "Extensions of lipschitz mappings into hilbert space," *Contemporary mathematics*, vol. 26, pp. 189–206, 1984.

[93] M. Mendel, "A simple proof of the johnson–lindenstrauss extension theorem," *The American Mathematical Monthly*, vol. 126, p. 838–840, Oct. 2019.