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: 13th 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 to become who I am now without all your nurture. In particular, I thank my grandfather for being the source of motivation to complete this project.

My most sincere gratitude also goes to my supervisors: Yuta Nagano, Benny Chain and John Shawe-Taylor for their endless support and teachings during the course of 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 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 whose aim is to eradicate cancer. Thus, it is possible to track peripheral T cells for early cancer prediction in asymptomatic patients. Previous attempts of constructing such model represents T cell receptors (TCRs) numerically using Atchley factors, Kidera factors and Amino Acid 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. This is because language models create numerical representations for TCRs based on the whole amino acid sequence, whereas physico-chemical properties encode TCRs based on individual amino acids only.

In this article, we demonstrate that the output embeddings from pre-trained large protein language models are more representative than physico-chemical properties. Using these 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 downstream multi-instanced classifier. Our results also show that using embedding models gives a more generalisable performance than physico-chemical properties in cancer prediction.

Contents

Chapter 1

Executive Summary

T cells play an important role in the adaptive immune system. They host receptors which bind to foreign antigens to cause an immune response during an invasion by a foreign agent. This response includes proliferation, which increases the concentration of that particular T cell receptor in blood to seek for the same antigen in the body to eradicate this invader as quickly as possible.

This proliferation process increases the chances of sampling the T cell receptors that target the invader from a blood draw. This subsequently leads to a conjecture where analysing T cells circulating in the blood can provide us with an insight into the set of invaders to the human at the time the blood is sampled.

Given this conjecture, many researchers model T cells to understand their specificity and subsequently aim to identify diseases in an asymptomatic patient. Many of this research quantify T cell receptors, which is a string, with physico-chemical properties. 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. There are other physico-chemical properties such as Amino Acid Properties and Kidera Factors which model different features of amino acids, which are used occasionally as well.

We define such means of representing TCRs numerically as a symbolic encoding. This is because they are encoding each amino acid in the TCR with a high-level meaning. However, we hypothesise that this symbolic means of representing T cells numerically can be improved through the use of subsymbolic methods, where subsymbolic encoding refers to the technique of encoding TCRs using transfer learning on pre-trained language models.

This hypothesis originates from the fact that randomly initialised models lack an understanding of these physico-chemical properties, and cannot, therefore, inference on subsequent problems about T cell receptors accurately. Furthermore, pre-trained models assign similar inputs with a close spatial locality, further assisting classifiers in learning about the research problem with T cell receptors.

Through the use of priori information about TCRs from pre-trained models, we can use the optimum embedding for TCRs, rather than physico-chemical encoding which is the same for all types of amino acids. We believe that this helps the downstream classifier and algorithms to more effectively learn the relationship with the input.

To test our hypothesis, we apply 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 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 which is significantly smaller than TCR-BERT. SCEPTR is trained on masked amino acid modelling followed by auto-contrastive learning.

Our data is T cell receptors from PBMC, sampled from a blood draw for both control and positive class. The control class comes from two different studies, and we only use data prior to experimental intervention; the positive class is the TRACERx Dataset, which contains TCR sequences sampled from a patient clinically diagnosed with NSCLC Stage I to IIIa.

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. Using these weights, we compute a weighted sum to obtain a repertoire-representing vector. A binary classification on this repertoire representing vector is then done to find the probability of cancer. The model that we propose has 2 linear layers, to demonstrate that even such a simple model can already identify cancer patients accurately.

The model is trained on both TRA and TRB CDR3 sequences, and we obtain good classification performance on the model which takes in SCEPTR's embeddings. The performances are not matched by models which encode TCRs using simply 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 layer to mitigate the risk of overfitting, we put a focus 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 SCEPTR repeats which identified TCRB sequences that are public and expanded in the evaluation set.

Chapter 2

Introduction

Cancer is caused by mutations that cause cells to divide uncontrollably, creating a tumour of malfunctioned cells that disrupts 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.

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

With technological advancements and researches 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 as opposed to laboratory experiments, there are now plenty of ongoing research to apply ML to the early diagnosis of cancer [6].

The promise in the future of deploying ML in clinical settings have been demonstrated through its success in detecting cancer within asymptotic patients [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 on the detection of a foreign agent such as viruses, bacteria or cancer cells. Ordinarily, T cells circulate around the body to seek for a foreign antigen that their receptor can bind to [16]. These foreign antigens are often found on either 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 of these chains contain 3 Complementarily-Determining Regions (CDR1, 2

and 3) which determines the T cell's specificity. TCRs are formed from a V(D)J Recombination Process, a stochastic process. This process generates a large variety of T cell receptors, consequentially enabling them to gain a wide range of specificities. This allows the T cell system to be able to target all possible foreign antigens [17].

Once a TCR successfully binds onto an antigen, or a peptide presented on a human cell's Major Histocompatability Complex, the T cell proliferate at an exponential rate. 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 that are circulating in 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].

A complexity to understanding the functionality of any specific TCR is with cross-reactivity, which states that one TCR can target several antigens, and one antigen can be recognised by multiple TCRs. Generally, the TCR that bind towards the same antigen between two different individuals will not be identical [20]. However there has been some evidence which suggest that these two TCR should be similar [20, 21].

In cancer, this process is further complicated as mutations that occur in cancer cells are not specific enough to guarantee the antigen 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 has 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 Quantifying T Cells

T cells can be sampled within any part of the body. Within each T cell, there are typically many TCRs, where each TCR chain is made up of 3 Complementary Determining Regions (CDR): CDR1, CDR2 and CDR3. Each of these can be represented with amino acids, which is a string.

In order to utilise TCRs to inference on the patient's condition, past literature have attempted in quantifying TCRs amino acid sequences using meaningful quantifiers [23, 24], which we refer 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 5 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 are able to encapsulate the physical and chemical properties of amino acids, we argue that they are not able to inform the computational model a deeper information about the TCR. These physico-chemical properties do not incorporate any information specific to TCRs but properties which applies to all amino acids. Furthermore, using physico-chemical properties to encode TCRs do not capture any interactions and dependencies between amino acids, whereby the symbolic numerical representation for a particular amino acid will be exactly the same irrespective of where it is placed within the sequence.

In contrast, large language models (LLMs) are able to overcome these problems. LLMs encodes

each amino acid depending on the structure of amino acids, therefore the embedding for one amino acid is dependent on where it is placed in the sequence. This is typically referred to as a context based embedding.

If we use a LLM that is trained with TCRs, it will capture intrinsic properties within the TCR, therefore generating a context based embedding that is specific for the TCR instead of an embedding that is universal across all amino acids. We denote representing TCRs numerically using LLM's output embeddings as a sub-symbolic encoding, since each value within the vector that represents the TCR does not carry high-level meaning about the TCR.

Instead of using an LLM that is trained from scratch, for the purpose of cancer identification using TCRs, we propose the use of a pre-trained model. This is further motivated from the observation that most of the previous works that uses TCRs to predict cancer do not pre-train their model with proteins or TCRs.

We assume that it is possible to create a meaningful sub-symbolic encoding space for TCRs, as it is a well known fact that proteins exhibit a grammatical structure [28, 29, 30]. A similar study with T cell receptors and B cell receptors have also used a subsymbolic encoding in one of their ensemble models to inference on whether a patient has Covid-19, Lupus or HIV. Their ensemble method has achieved an AUC of 98.1% [31], demonstrating that using sub-symbolic encoding can provide a better computational understanding on TCRs instead of symbolic encodings.

2.3 Research Aims

In this literature, we demonstrate with novelty that representing TCRs numerically sub-symbolically through the use of TCR-BERT [32] and an in-house model SCEPTR, two different pretrained 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 not in anyway related to the dataset used in this literature. Therefore, it is unlikely that the two encoding models' output embedding carries information about the disease specificity, but just information about the TCR. TCR-BERT was pretrained on masked amino acid prediction on T cell CDR3 sequences, whereas SCEPTR is a BERT-based model pretrained on V and J call sequences as well as CDR3 sequences from both the alpha and beta chain using masked language modelling followed by auto-contrastive learning.

Since we argue that subsymbolic encoding provides a better means of 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) and the confusion matrix.

2.4 Report Structure

This report is structured as follows:

In Chapter 3, we first demonstrate that there is 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 in predicting cancer using TCR Repertoire Analysis as well as their weaknesses. We will also prove that proteins exhibit a grammatical structure and subsequently review TCR Large Language Models.

Taking on Chapter 3's findings, we then formulate a formal methodology in Chapter 4 which uses TCR Repertoires to predict cancer. We will also describe our experiment which aims to show that subsymbolic encoding is more expressive than symbolic encodings. We then provide results from a simple models 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 to our experimental hypothesis. Subsequently, we discuss weaknesses in our approach. We also look for areas of improvements in our current approach and provide guides on future work. We will also argue our models' robustness through interpreting the model, and to know to what extent the model has learnt this biological problem.

Chapter 6 concludes the report, where we summarise all our findings, limitations and future work.

We provide a technical summary with a more thorough overview to the project in the Executive Summary.

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 demonstrates that a linguistic approach in understanding TCRs is a correct approach.

We then review literature that show there is a distinguishable difference between healthy patients' and cancer patient's TCR repertoires through a motif-based approach.

Subsequently, we study literature that uses machine learning to accomplish a similar task. All of these literature creates vectors that represent TCRs rigidly through the use of 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 the use of language model embeddings.

Then, we will study transfer learning as an approach in classifying TCRs. This is a promising approach because pre-trained models have already acquired an understanding on 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 on why the use of output embeddings from large language models to quantify T cells 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 motifs that are frequently occurring. It has been found that most motifs have sizes from 2 amino acids to 16, with most motifs being 7 amino acids long. Although proteins operate in a three-dimensional space [35, 36] with contrast that languages are one-dimensional, there has 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 solidly 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 are also not captured in databases which stores the structure of naturally occurring proteins. This further proves that proteins exhibit a language-like structure, one 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, therefore 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 which they recognise [18]. Although proliferation and differentiation mostly occur in tissues, T cells migrate around the body in blood to seek for these foreign cells that may have spread around the body [37].

Hence, we can sample and measure peripheral TCRs that has been stimulated by antigen exposure, which can tell us about the type of attack that the body is under at the time 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 an evidence to this hypothesis, studies analyse differences with TCRs' motifs occurrences within a TCR repertoire for patients that are under the same immunological stimulus against a control group of patients. Since TCRs exhibit a grammatical structure as they are made up of 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 mice and unimmunised mice. This shows that there is a significant difference between the two populations' TCR repertoires. Although this study is not related with 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 such that it has minimal restrictions to identify tokens that can best represent the TCR sequence of any length. 16 motifs are identified within the TRACERx Lung Cancer Dataset [39] that are enriched in cancer patients when compared to healthy patients.

However, TCR frequency analysis is insufficient to 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 an ineffective approach. The differences in each human's T cell system further highlights 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 is able to demonstrate that there is a difference between cancer patients' against control patients' TCR repertoires, it is analogous to comment the use of TCR frequency analysis to classify whether a patient has cancer as classifying whether a sentence is workplace appropriate by filtering on swearwords.

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 be able 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 contains redundant information [24]. There are several efforts in reducing 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].

Taking these methodologies in representing amino acids numerically, DeepCAT is one of the earliest studies in predicting cancer using TCR repertoires. The authors used Convolutional Neural Networks to predict cancer. They have achieved an AUC of > 0.95 [15] after quantifying TCRs using 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

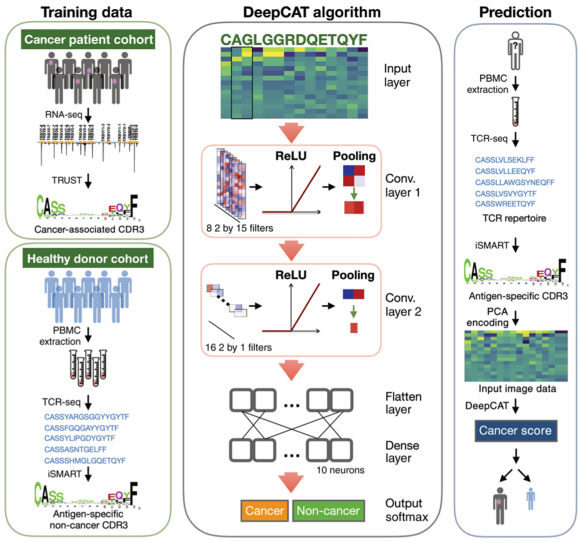

Figure 3.1: DeepCAT Training Paradigm [15]

two sampling methods, rather than the true relationship between TCR repertoires from cancer patients and control patients.

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. Accompanied with the fact that the AUC of 0.8 for a binary classification scenario is rather low, it is evident that 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 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 TCR beta chain CDR3s that are involved in binding. Subsequently, the authors encoded 4 amino acids within TCRs at a time using the Atchley factors [25] and studied the importance of each factor within the Atchley factors through the use of 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 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 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 attentions 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 - this is due to the fact that TCR datasets are arranged in repertoires, where each repertoire contains a different amount of TCRs. Within the repertoire, we will have a label of whether the repertoire comes from a cancer patient or a healthy patient, rather than whether each individual TCR targets cancer antigens. This dataset structure is due to the practical impossibility to label 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 through the use of 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 is created using the Sparsemax activation function [46], which is a variant of softmax but 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 is unable to 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 is able to output a zero probability, it amplifies the signal for relevant TCRs within the bag-representing vector, as irrelevant TCRs will be not be involved within the sum. Hence, all vectors in the weighted sum will be, to some degree, related to the classification label.

Not only does the usage of sparsemax increase the sparsity of the probability distribution generated, therefore increasing the speed of computation, sparsemax improves the explainability as well. This is due to sparsemax's ability in distinguishing the instances that matters and those that do not. In a trained model, it is expected that TCRs that are related to cancer will be assigned a non-zero probability; whereas the irrelevant instances will be assigned a probability of zero. This increases the interpretability of the model, as we can now see which instances are deemed to be important by the model, therefore we can explain what the model has learnt. We can also track if the model's knowledge is in line with what we know about the problem.

The Atchley matrices for each CDR3 sequence are processed into 30-dimensional vectors by TESSA, which is then passed into the deep network to obtain the components within the weighted sum. The processed vectors and weights from sparsemax together 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 10 types of cancer in balanced and imbalanced data scenarios respectively. This AUC performance might originate from the fact that the authors did not use a lot of data in this study. Therefore, MINN-SA is prone to overfit as the deep network that creates the high level feature vector contains much more trainable parameters than the amount of labels.

Although it is not convincing that MINN-SA has generalised the relationship between cancer and non-cancer patients by evaluating the AUC, there are many features within MINN-SA that stands out. We believe that the use of sparsemax and the weighted sum is a key aspect which could be learnt from MINN-SA.

3.4 Transfer Learning

To highlight the importance of using transfer learning, let's consider a new born baby 'Timmy' who is destined to be a radiologist. For Timmy to progress in his destined career, he will have to learn Biology, earn a medical degree, and pass his residency. 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 is able to 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 most probably true that he might not understand the underlying reasons of why one medical image correlates with a symptom.

This underpins one important problem with all models seen in the previous section. Models such as DeepCAT [15] have used randomly initialised models with no knowledge for TCRs whatsoever, and are directly trained to predict cancer using TCR sequences. Although these models have demonstrated good AUC results, this can possibly be improved by taking a more gradual approach in training models. This can be done by providing sufficient background knowledge of the problem to the model prior training 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 the randomly initialised model is unable to understand what these physico-chemical properties correspond.

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, which processes CDR3 sequences as if it were an image. This design overlooks the language-like structure of amino acid sequences, which exhibit language-like patterns instead of visual images.

This oversight becomes particularly problematic in tumour environments. The authors highlighted a key issue where the predictive power of TESSA diminishes due to the homogenisation of T cell functional patterns in cancerous contexts. Such environments weaken the correlation between TCR specificity and T cell phenotype, rendering 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 the sequential, language-like structure of TCR sequences, which is critical for accurately interpreting TCR functional implications in complex biological settings like tumours.

Hence, we argue that the usage of TESSA in detecting cancer is a domain mismatch. Domain mismatch is when we apply a model on a domain that the model has not been trained on, or

has been shown to be weak at. We have seen that TESSA is weak in tumour environments, such as cancer. Hence, applying TESSA in identifying 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 the use of pre-trained large language models (LLMs). LLMs are able to understand language-like properties of sequences, and can offer an insight into the complex relationships within TCRs. Leveraging LLM's ability in understanding intrinsic language-like properties of amino acids, a more thorough understanding on problems to do with TCR sequences can be acquired. Using LLMs to model TCRs can also avoid problems such as inability to generalise in particular scenarios like cancer in TESSA.

Pre-trained models (PTM) such as ChatGPT [48] are models that have already been 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 are able to inference on these specific tasks, they benefit from fine-tuning, a process that uses data from a smaller domain and specialises a PTM to it. An example of this specialisation process is when we fine tune a large language model like ChatGPT to classify whether an email is spam or not.

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 gives a significant improvement in understanding patient inquiries and providing more accurate consultations as opposed to 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 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 that are pre-trained on TCRs to generate an encoding for them. This approach can be promising as LLMs are able to create vectorised representations of the input based on the ordering of tokens and its surrounding tokens including ones that are positioned further away. This provides a strong promise to our proposal of using pre-trained TCR language models in cancer prediction, as they are able to 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 approaches of fine-tuning LLMs 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 a more gradual

process in 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.** Link here.
   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 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 prodominantly 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 other phenotypes such as the HLA alleles.

2. **Step 2: Antigen Classification.** Link here.
   Taking the model from the previous step, TCR-BERT is further trained on 4,365 β chains' CDR3 amino acid sequences to predict antigens that they bind to.

Through a gradual training process, it is promising to say that the model understands the underlying problem that it is tackling with. The authors verified this hypothesis by pairing TCR-BERT with downstream machine learning algorithms, to tackle supervised learning 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 different amount 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 showed a significant improvement in performance when compared against 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 demonstrated an improved performance in predicting antigen binding when given paired TCR CDR3 alpha and beta sequence, giving 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 that it gave 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 is able to 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 is able to transform the data such that downstream algorithms can better capture patterns within the data.

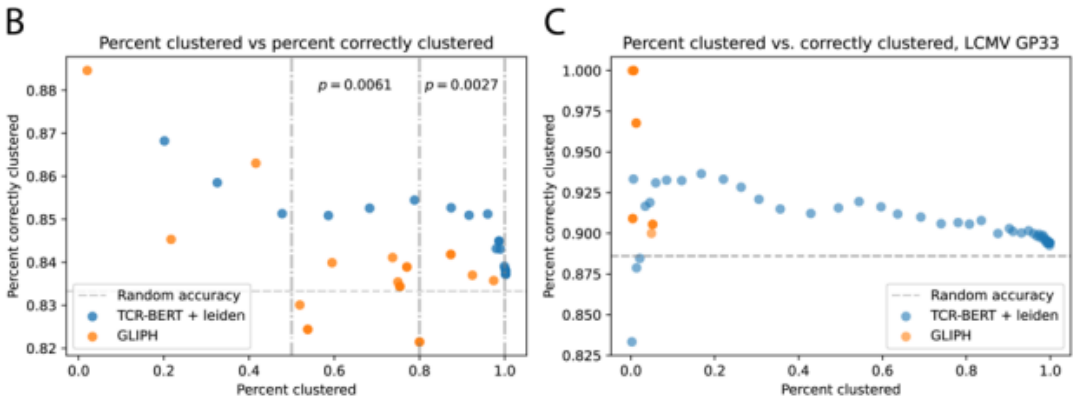

Figure 3.2: TCR-BERT compared against GLIPH [32]

With the robustness as well as the performance from TCR-BERT across both supervised learning and unsupervised learning tasks, TCR-BERT provides a promise that applying transfer learning on 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 is able to assist models in understanding a specialised domain of expertise by first learning a broader domain, we aim to utilise large language models pre-trained on TCRs to predict cancer by pairing it with a downstream classifier.

In this chapter, we demonstrate how the model architecture, training algorithm and training paradigm is designed. In particular, we focus on two different types of encodings: 'symbolic' and 'subsymbolic' encoding. The unconventional usage of these two phrases are inspired from 'symbolic AI' and 'subsymbolic AI'. The creation of these phrases are 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 through assigning meaningful values to each symbol. These values have a human understandable meaning towards it.*

Definition 2 (Subsymbolic Encoding) *Subsymbolic encoding is a method that assigns each symbol within a nonnumerical input with values that does 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, where this refers to the use of subsymbolic encodings from a TCR PTM, is a more effective means of training downstream classifiers to classify cancer as opposed to symbolic encodings.

To provide an evidence to our hypothesis, we will use 4 different symbolic encodings and compare their performances with 2 subsybmolic encodings. The 2 subsymbolic encodings are from TCR-BERT and an in-house pretrained model SCEPTR, which will be introduced in section 4.1.3.

An overview 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 amount of neurons in the neural network.

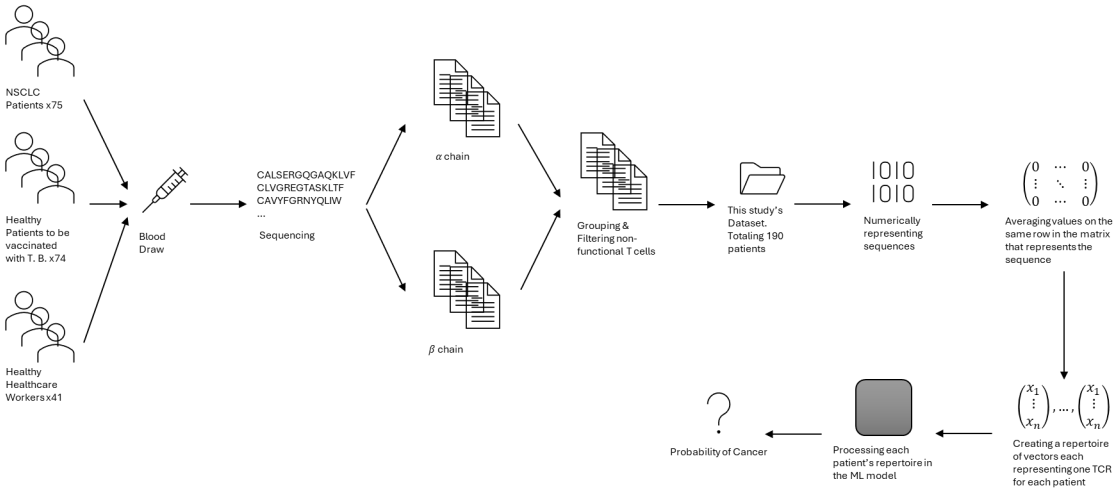

Figure 4.1: Data Pipeline

4.1 Data Processing

This section describes the data that we are using. We will introduce the training data, how we clean the data to how representing them numerically is done. 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 originality of SCEPTR.

4.1.1 Training data

We use 3 different datasets to train our model, where these 3 datasets are stored within the Chain's Lab Research Data Storage and is not publicly available yet. TCRs are sequenced using the same sequencing methodology in the 3 datasets to avoid a sampling bias within this study. We suggest referring to the articles for each dataset to know 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 stages 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 with an aim to study healthcare workers during the first wave of the Covid-19 epidemic [65], and another dataset was collected to investigate the effect of *M. tuberculosis* vaccination [66].

To maintain fairness, in that the control data is not obscured with T cell receptors that has a *M. tuberculosis* or a Covid-19 specificity, we will only use data from individuals who were not infected with Covid-19 or *M. tuberculosis*. All experimental subjects within the control set has 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. It is unethical to invasively sample TCR repertoires from a healthy patient's lung. However, since PBMC data can be sampled from a blood draw, it is both ethical and unbiased to use PBMC data for both classes. Note that when we use the phrase 'PBMC', we refer to only T cells whilst with appreciation that PBMC includes more cells than T cells.

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

4.1.2 Data Cleaning

We clean the data by removing all non-functional TCR sequences with the use of TidyTCells (TT) [67], a data cleaning library for TCR repertoires. We use TT to find non-functional TCRs for removal and standardises TCR annotations to be IGMT-compliant. To make sure that SCEPTR and TCR-BERT, the two pre-trained models perform properly as both of them 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 files for the cancer set, totalling to 379 files. 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 and another for their beta chain. It should be noted that these alpha and beta chains are not paired.

To help the model to 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 epitope specificity of a TCR, and is 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, with each file corresponding to the TCR repertoire from the same patient. Note that one of the pre-trained models, SCEPTR, can process V call, J call as well as CDR3 sequences, therefore we include the V and J call genes in the files that are used to train SCEPTR's downstream classifier.

4.1.3 SCEPTR

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

SCEPTR 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, J call and CDR3 sequences to create an embedding. Since SCEPTR is not trained with data that is used in this study, this avoids data leakage to the downstream model. Using this dataset, SCEPTR was first trained on masked amino acid modelling, and subsequently contrastive modelling to

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

assign spatial locality for similar input data. It is trained to encode TCRs onto a 64-dimensional hyperspace.

4.1.4 Symbolic Encodings

3 different physico-chemical encodings alongside with 1 'control' encoding will be used, which are collectively referred to as symbolic encodings. All 4 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 which serves as a 'control' encoding. For each amino acid, we use 5 features which is pre-generated from a random uniform distribution from 0 to 1. We are using 5 features in this encoding since the minimum amount 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 therefore assume that all 3 physico-chemical encodings will outperform the random encoding, as the random encoding's embedding space should be difficult to learn since it is random. On the contrary, if any of the 3 physico-chemical encodings has a similar, or worse performance than the random encoding, then this shows that the particular physico-chemical encoding is not an effective representation space.

We create a vector representation for each TCR CDR3 sequence from each of the symbolic encodings by first generating an embedding for each amino acid in the sequence using the TSV file. Then, we average the list of embeddings vectors, creating one vector which represents 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] seems to be taking in strings as an 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. The aim of this transformer is 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 gives 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 are many research that supports this claim, where it has been suggested that earlier layers within neural networks are not only applicable for one dataset, but seems 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 difficult 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 TCR, we are looking for an output embedding that can be beneficial for our prediction task - having established that earlier layers of PTMs contains 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 is 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 one is SCEPTR, as previously introduced in section 4.1.3. SCEPTR is a pretrained 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 embeddings based on a TCR's V call, J call and its CDR3 sequence. SCEPTR comes with a package that computes the vector representation of TCRs using the pre-trained model. Therefore the details for how we calculate the vector representation of TCRs are not covered here.

The second model that we use is TCR-BERT, which has been reviewed previously in section 3.4.3. We will use the masked amino acid modelling pre-trained variant of TCR-BERT, which is able to process alpha and beta chain CDR3 sequences. This can provide more information about the patient's TCR repertoire to downstream models as opposed to the other variant which can only process beta chains. Note that TCR-BERT is trained on VDJdb, which is independent to Tx and the 2 control datasets.

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

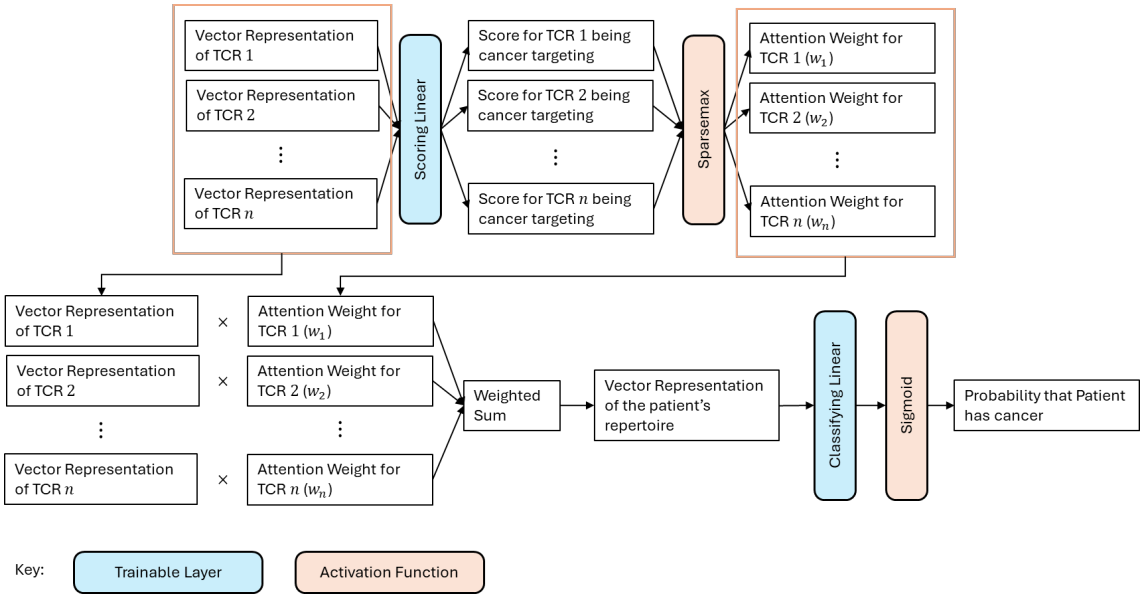

Figure 4.2: Downstream Model for Cancer Classification

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, to extract the embedding from a BERT structure, we should extract the embedding of the stop token. However, since TCR-BERT is a modified BERT model which is to tackle with classification rather than protein generation, 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 amount 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 that represents the CDR3 sequence.

During our training, we will freeze the parameters in both large language models, therefore they are not trainable.

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 subsymbolic encodings are able to inference better than 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 exact 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 to classify whether a patient has cancer using TCRs is a Multi-Instance Learning problem. Within each repertoire (often referred to as 'a bag' in the context of Multi-Instance Learning), we have a collection of TCRs from patients, whereby we have a label for whether this patient has cancer but we do not have a label of whether each TCR has a cancer specificity.

This further motivates why a simple model is important. During forward propagation, the input to each layer is stored for computing 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, and the number of TCRs per repertoire could be large. This means that forward propagating a whole TCR repertoire down a deep model could incur a large amount 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 usages, causing complications when training the model due to computational constraints.

Creating a model for this task is difficult as there are only a few TCR instances in a cancer bag that has a specificity with cancer. Moreover, the amount of TCRs in a bag is not a determined value, and it is also not possible to sort the TCRs in the bag 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 when compared to 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: Sort z as $z^{(1)} \geq z^{(2)} \geq \cdots \geq z^{(n)}$
3: Find $k(z) := \max \left\{ k \in [K] \,\middle|\, 1 + kz^{(k)} > \sum_{j \leq k} z^{(j)} \right\}$
4: Define $\tau(z) = \left(\sum_{j \leq k(z)} z^{(j)} - 1 \right) / k(z)$
5: **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 a very 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 the weights are generated by a softmax function, 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 making it hard for the downstream classifier to learn a relationship.

To use the sparsemax function, we will need to assign a 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 to be the scoring weights and b_s to be the scoring bias. We name this layer as the scoring layer, and the usage of 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 is able to put TCRs with the same specificity in a close distance.

After we obtained s_i, we can obtain p_i, the weighting outputted by the sparsemax function for instance i. We can think of p_i to be the probability that x_i is a TCR that has 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.

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 such that we can properly compare the difference between the quality of the embedding space for each encoding method, it is imperative that we do not introduce further information into the encoded vector through the use of kernels, nor increase the difficulty of the learning algorithm by training an unnecessary amount of parameters.

We also believe that the use of 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 out the vectors to obtain one vector that represents the TCR CDR3 sequence for fairness, since we are doing the exact same operation for TCR-BERT. Note that all values within the vector will be in range of 0 to 1 to ensure the range of the input does not affect the difficulty of the learning algorithm.

To avoid increasing the difficulty of the learning algorithm whilst ensuring that we evaluate the subsymbolic and symbolic encodings fairly, we believe that the symbolic encoding's downstream model should share the same model architecture than the subsymbolic encoding's downstream model. This is to ensure 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 are significantly lower than subsymbolic encodings, we create table 4.1 to demonstrate the amount of trainable parameters in our model

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

for each encoding means. Note that there are 2 trainable linear layers where both of them output a scalar, hence, the amount 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 used 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 has 24GB GRAM.

Due to resources limitations, it is impossible to correctly report the time taken for each training process as if they were ran 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 not accurate, which is why we do not report the training time in this report.

4.3.2 Hyperparameters

Since TCR-BERT has a significantly higher output embedding dimension as opposed to 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 due to the fact that TCR-BERT's downstream model has 1538 parameters, whilst we have only 190 labels to train it with. 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 a randomly initialised compression matrix by exploiting the fact that an expressive feature space should still be expressive if we were to downscale it using a linear transformation, 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, which is an extensive amount for 190 bags. We deem this amount of epochs as an extensive amount 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 each of the 50 epochs. The best model is selected by the highest test set AUC in the training instance, whilst the test loss at that epoch should not be increasing as compared to the previous few epochs, as this is a sign of overfitting.

The learning rate is set to 0.001 for all 6 embedding methods on the Adam optimiser [85]. This learning rate is relatively small and should allow the model to take enough steps throughout the 50 epochs to converge.

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 a 80%-20% train and test data split, rendering it not feasible to perform such an extensive search. The problem with time taken is more severe in training TCR-BERT's downstream model, where it took at least 72 hours to complete the 50 epochs of training.

To mitigate problems to do with class imbalance, we up scale the Binary Cross Entropy loss calculated. 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 be 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

For each encoding, we repeat the exact same training process for 5 times, unless otherwise specified,. Across these repeats, we will use different training and testing data splits as well as model parameter initialization. Note that the results we present 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 summarises the performance of the best checkpoints for each encoding's downstream model's training instance. Details can be found in Appendix A, where we attached the training and testing loss graphs, AUC graphs and the confusion matrices for each encoding method.

4.4.1 Evaluation Set

In machine learning, it is often considered a good practice to split the data into train-test-evaluation sets. The training set is used to train the model and the test set is used to validate the model in midst of training. The evaluation set is often withheld to observe the different models' performance on some mutually unseen data after training.

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

We have observed that SCEPTR's embeddings allow its downstream models to attain a very strong test set AUC, therefore we create an evaluation set by manually withholding 10% of the data which is equivalent to 19 patients' data. We 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 for 10 times instead of 5, where each of these training instances splits the remaining 90% of the data using a 80%-20% train-test split.

These 19 patients are composed of 10 control patients, where we extract 5 patients from each the 2 control datasets. The remaining are cancer patients. The patients extracted from each dataset have consecutive patient IDs for convenience. Since the patient ID and the difficulty of classifying the patient is not correlated, the methodology in choosing repertoires is not based on any prior knowledge of its difficulty in classifying. On the contrary, some of the files withheld to serve as the evaluation data have only CDR3 sequences and does not contain the V call and J call. This introduces an extra level of difficulty in classification as the TCR encodings are not created from a thorough view of the chain.

Taking the best checkpoint from each of the 10 repeats, we test the downstream models on the evaluation data. The evaluation set's AUC curves and confusion matrix across the 10 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 10 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

It is impossible for data leakage to happen during training as each TCR repertoire is cleaned independently to other repertoires; the evaluation data has been manually withheld, and therefore

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 onto SCEPTR as the subsymbolic encoding method. We also evaluate the weaknesses with all symbolic encodings as well as limitations to our investigation.

An outline of potential future works to this project will be provided. The future works will be in 3 directions: increasing the accuracy of cancer prediction using TCRs, extending this investigation between the expressiveness of subsymbolic encoding against 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 3 main evidences to 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 is significantly lower than the test loss, which is a sign of overfitting. We will propose methods to mitigate this issue in section 5.3.

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.

3 evidences will be provided to support this claim. Collectively, they reason on why physico-chemical encodings are not as expressive as subsymbolic encodings, therefore supporting the claim above.

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, which has been 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 show a gradual 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 this improvement is insignificant. All of SCEPTR's downstream model's training instances have succeed 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 that there are some easily learnable patterns within SCEPTR's and TCR-BERT's output embedding since most of their downstream models' training instances were able to eventually fit towards it. 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 difficult 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 $-\ln(0.5) \approx 0.693$. We see that smallest train BCE loss for the four symbolic encodings is 0.570 and smallest test BCE loss is 0.595, which is much closer to 0.693 as opposed to 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.

Confusion Matrix

Although we observe that the average test AUC for the four symbolic encodings' downstream classifiers are high in table 4.3, their confusion matrices in the train and test set as provided in Figure A.18 for Atchley Factors, Figure A.24 for Kidera Factors, Figure A.30 for AA Properties and Figure A.36 for Random Encodings demonstrate that the downstream model makes more mistakes than TCR-BERT's and SCEPTR's downstream model, which can be seen by their confusion matrix in Figure A.6 and A.12 respectively. It is worth noting that the confusion matrix for Atchley Factors' downstream model showed that the model predicted everything into the negative class despite efforts in mitigating class imbalance.

This demonstrates that it is difficult for the symbolic encoding's downstream models to find an optimal direction where all cancer targeting TCRs are located as all four symbolic encoding's downstream model make significantly more mistakes than SCEPTR or TCR-BERT's downstream classifier. On the other hand, we believe that SCEPTR's and TCR-BERT's downstream classifier's

high accuracy is achieved by placing similar TCRs together, therefore helping the downstream model to learn effectively.

Note that SCEPTR's confusion matrices in Figure A.12 are generated from models that are trained after taking 10% of the data away to serve as the evaluation set.

Difference with Random Encodings

Since random encodings are used as the control encoding, where this encoding carries no meaning whatsoever as opposed to the physico-chemical encodings. We would expect random encodings' downstream classifier to perform worse than 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. In certain occasions, we see that random encodings' downstream model is performing better than physico-chemical properties' downstream model in accuracy and AUC. For example random encodings' downstream model had a higher accuracy and AUC than AA Properties' and Atchley Factor's downstream model in both the train and test set.

This provides an evidence that random encodings are approximately as expressive than the physico-chemical encodings since the difference in performance of their downstream models are minimal.

On the other hand, we can see a significant difference between SCEPTR's and symbolic encoding's downstream model's performance in the test set. This observation cannot made when we compare the difference between the performance of physico-chemical properties' and random encodings' downstream model. 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 than 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) respectively. Note that these numbers do 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 in, 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 3 aforementioned studies. This is because some articles such as MINN-SA classified tumour

[1] The authors for MINN-SA did not provide the exact numbers, and these numbers are only an estimation from reading Figure 5.1

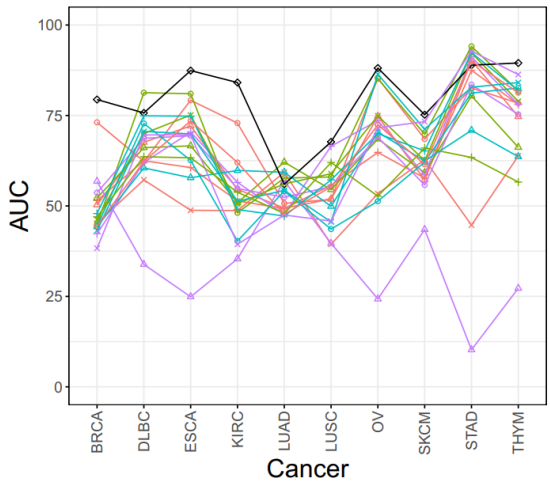

Figure 5.1: AUC Results for 10 Cancer Types on imbalanced dataset [23]

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 the usage of pre-trained TCR language models as the encoding method would be a good direction in researching into methodologies in predicting whether a patient is diseased.

5.2 Interpretability

We have seen success over SCEPTR's downstream model, whereby it 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 learnt through looking at the probability distribution which the sparsemax layer generated. We will focus only on the best checkpoint out of the 10 repeats that 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 demonstrates 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 is 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 into which TCRs corresponds to cancer by interpreting which TCRs have been assigned a non-zero weight. Subsequently, we can gather these TCRs for each correctly classified patient and attempt in understanding what the model has learnt, as well as evaluating whether this knowledge is in like with what we know in biology.

We attempt in understanding 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 instance is hereby referred to as 'the model'. In particular, we are most interested in whether if we see the exact same V call, J call and CDR3 sequence within different cancer patients that are correctly classified by the model.

Amongst the 19 patients in the evaluation set, the model has correctly classified 18 patients, where it incorrectly classified one control patient. Amongst the 9 correctly classified true positives, the model has assigned non-zero weights to 32 alpha chains and 292 beta chains. This means that the model believes that there is more signal in the beta chain than the alpha chain for cancer patients. 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 call and J calls within these non-zero weighted alpha chains and beta chains, as well as their occurrence frequencies. We can see that TRBV2 is the only beta V call that has been picked up by the model, 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.

Amongst all cancer targeting TCRs registered in VDJdb, we observe that there are a total of 135 TCRs that has a TRBV2 gene, and 294 TCRs that has a TRBJ2-2 gene. Amongst which, 25 TCRs have both TRBV2 and TRBJ2-2 as their beta chain V and J call genes.

We gather the CDR3 sequences that has been seen in multiple patients and placed it in Table A.3. It can be observed that there are multiple rows that has the same CDR3 sequences. This is due to the fact that the weighting assigned by sparsemax for the same CDR3 in different patients' will be different.

In total, SCEPTR's downstream model picked up 7 CDR3 chains that are seen across several patients. 6 of these CDR3 chains are from the beta chain and 1 is from the alpha chain. We can see that these 6 beta chains are all formed from the TRBV2 V call and either TRBJ1-2 or TRBJ2-2 J calls. We attempted in searching 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 4 patients, this chain had been expanded in 3 of them. The phrase 'expanded' refers to the scenario where the same chain has been spotted more than once within the same blood draw, which is caused by T cell proliferation during clonal selection. We also observe that the CDR3 chain 'CASSFRGGADEGYTF' is expanded within the two patients that had 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 signal 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 sequences that the model

picked up was an alpha chain which has CDR3 sequence 'CAVGGYNKLIF', with 8 occurrences within the same patient.

All in all, what we can see is 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 the cancer patients. In the non-zero weighted TCRs in true positives, the model prioritises beta chains, which were thought to carry more signal than alpha chain.

For the non-zero weighted TCRs in the true negatives, the model picks up a large selection of TCRs, which demonstrates that there is 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 is against every repeat of SCEPTR's downstream model. The scoring layer is the layer that assigns a score to each individual TCRs for assigning weights by sparsemax. We also evaluate the similarity of the classifying layer across different training instances.

The intuition for doing so is because if SCEPTR assigns all cancer targeting TCRs into one close region, then all models that are trained on SCEPTR's embedding space will have the scoring layer pointing to the same direction since there is only one region that has the features which help the model in understanding the relationship. Hence, we should expect that 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, therefore it should not be an expectation that SCEPTR assigns spatial locality by specificity.

Sparsemax assigns weights by seeing how large the score is. This means that only the instances that has a close distance with the scoring layer's weights will be assigned high scores. Therefore, we would assume that the scoring layer is similar as classifying layer. Using this argument, we would also expect the classifying layers' weights to be similar when compared against another training instance.

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

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

The euclidean distance measures the actual distance between two different points, and the euclidean distance is also referred 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 are created. The L2 Norm is defined as below:

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

Using these 2 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.

If our aforementioned hypothesis that SCEPTR assigns cancer targeting TCRs into one close region, then we should expect the euclidean distances of different training instances' classifying layers and scoring layers to be 0 and cosine similarity to be 1. Yet, we can see in both Figure A.39 and A.40 that the similarity between different training instances' classifying and scoring layer is low. In the same training instance, the similarity between the scoring and classifying layer is low as well. This suggests that SCEPTR does not assign spatial locality on TCRs by its specificity on the 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 demonstrate our observation.

5.3.1 Data Limitations

Although data from 190 patients is not considered as a small number in computational biology as clinical data is difficult to collect, it is insufficient 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 ethic backgrounds [20, 40, 86, 87]. Our data could be biased in the sense as our data is collected from patients with the same ethnicity, as data collection has been performed in the UK only. Thus, it is expected that a majority of patients involved in this study have similar ethic backgrounds.

Furthermore, we have seen that TCR-BERT overfitted as there are too many parameters to train given the amount of labelled data that we have. We propose two methods which could help overcoming 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 is to introduce more data, which can also help the downstream model for SCEPTR to better learn the relationship between TCRs and cancer.

5.3.2 Quality of Embedding Space

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

Although this is an expected phenomena 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 does not only apply on identifying whether a patient is diseased,

but also with other TCR problems such as pairing alpha and beta chains and pairing TCRs with antigens.

Yet, this limitation does not reduce SCEPTR's downstream model's performance as we have seen impressive results from them 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 that there are certain restrictions as to 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 same specification as mentioned previously in section 4.3.1 to run the training for 6 different encoding types, especially considering TCR-BERT takes a large amount of GPU memory to run.

It would definitely be more desirable if each encoding method has its own set of hyperparamters, so the choice of hyperparameters is not an influence to the model's performance which relates to the encoding space's expressivity.

We believe that this problem is particularly predominant within TCR-BERT. This is because 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 which is 0.25 is only a rough guess. It is possible that TCR-BERT could perform better than SCEPTR as the encoding model, however this is to be verified in future works. We will propose methods to compare TCR-BERT's and SCEPTR's expressivity in section 5.4.5.

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 to tackle with TCR-BERT overfitting.

5.4.1 Fine Tuning Encoding Models

During our experiments, we have not fine tuned TCR-BERT or SCEPTR to create a more desirable output embedding space. This is due to firstly demonstrating robustness with encoding models, and secondly limitation to GPU memory that we have. These embedding models are a generalised method in 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 so they are tuned towards disease identification can mitigate the problems with embedding space quality as discussed in section 5.3.2.

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

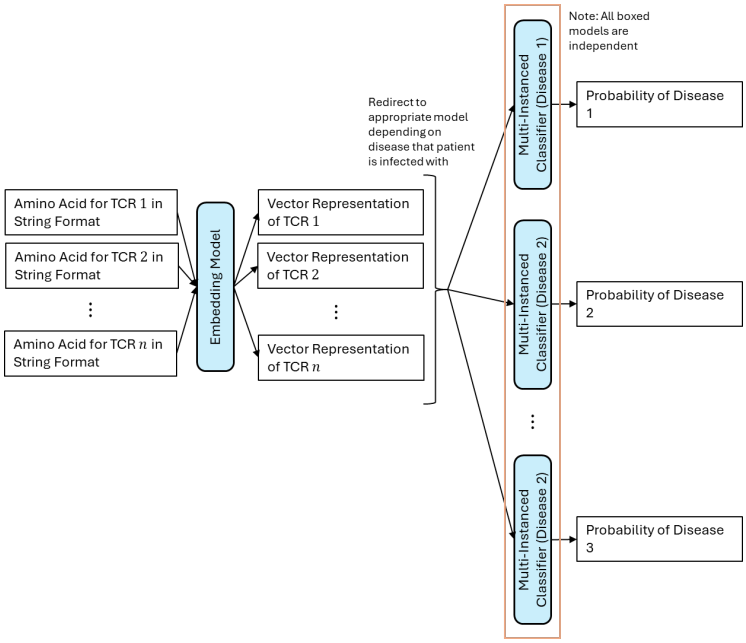

Figure 5.2: Fine Tuning Regime for Encoding Models

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 greatly 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 amount 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 that of 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 patient's repertoire data, it will most probably cause the model to overfit very quickly as we have seen that TCR-BERT's 1538 parameter downstream model overfitted quickly.

To mitigate overfitting, we propose that we can fine tune one LLM by incorporating more TCRs from patients who are infected with diseases other than cancer. Taking these data, we can train a series of downstream classifiers that are in the same architecture as what we used.

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 model as well as the embedding model. It is important that we update each classifier sequentially, rather than completing the training for one classifying model and going onto the next. This avoids the encoding model from forgetting knowledge from disease that was first processed. The reason why we believe that this training regime would work is that 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

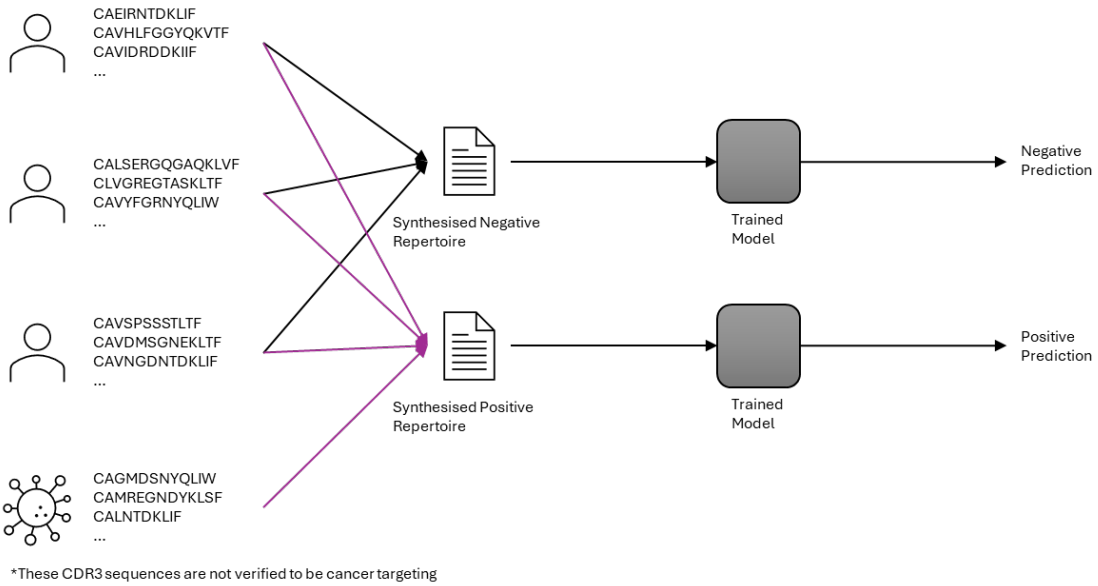

Figure 5.3: Classifying Test

mitigate the problem with GPU memory consumption.

5.4.2 Patient Background

It has been observed in [31] that when the patient's background such as ethnicity, age, gender and other factors are included into the consideration of classifying whether he 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 if we allow the model to know more information regarding the patient, it allows the model to make well-informed decisions regarding whether the patient has cancer. This also follows what is known in biology, since likelihood of having cancer has been shown to be 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 these information to our model, we could concatenate the vector that represents the patient's background onto the patient's bag-representing vector. This assumes that the patient background vector is scaled to be within an appropriate range.

5.4.3 Model Verification

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

We name the easier test as the classifying test, which has been 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 training of the model. We will then pass this augmented repertoire into the model, where we should expect the output of the model, p, i.e. the probability of having cancer, to be lesser than 0.5, which corresponds to a negative prediction.

Subsequently, we add the cancer TCRs into the augmented repertoire, which simulates a blood draw from a cancer patient. We would expect the model to output a probability higher than p and is also higher than 0.5, corresponds to a positive prediction. This is because we know that there are now cancer targeting TCRs within the repertoire, which can only occur when the patient has cancer.

The second test is the score test, which is more difficult 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 that the model outputted. 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 has no knowledge of TCR specificity, it will most 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 about TCRs and cancer.

5.4.4 Kernel Methods

We argued in section 5.3.2 that the embedding space generated by SCEPTR does not allocate TCRs with the same specificity into one location. This applies to all other encodings, as the downstream classifier should have a high accuracy and AUC if the embedding space puts TCRs with same specificity together, since there would now be a straightforward way to classify whether a patient has cancer or not with TCRs.

However, since SCEPTR does not have this property, and other encoding's downstream model is performing worse than SCEPTR, this demonstrates that all encodings cannot assign TCRs by specificity.

We could investigate whether using Kernel Methods can make the TCRs linear separable. This is because Kernel Methods transforms data to a higher feature space, whereby this feature space could make the features linearly separable, therefore the usage of one linear layer would be sufficient to capture the relationship between TCRs and cancer.

There are two major approaches of using kernel methods in this multiple instance learning scenario. We could use a traditional kernel method such as a support vector machine (SVM) to classify bag-representing vectors, or multi-instanced kernels paired with carefully designed models [92].

The former method is trivial: we fit an SVM with a chosen kernel, such as the radial basis kernel or the polynomial kernel to the bag-representing vectors. We back-propagate the loss of the SVM

to the scoring layer so we can obtain better representations of the bags.

The latter method is less trivial. In [92], the authors proposed two different ways to use kernel methods in a multi-instanced problem. One focuses on applying kernel methods on the bag, and another focuses on applying kernels on the instances. Set Kernels are kernels designed to be applied on a bag of instances. This includes the Earth Mover's Distance (EMD). The EMD measures the dissimilarity between two bags, where it considers each bag as a distribution of points in a feature space.

However, the usage of set kernels such as EMD causes a problem with time complexity, where it takes a long time to compute the kernel between two large bags.

On the contrary, instance level kernels would require pairing instances between two bags. This is not an effective method as we are unable to create meaningful pairs between two TCR repertoires. We could compute all possible pairs of TCRs between two bags, but this is not computationally feasible especially.

Therefore, we believe that using SVMs on the bag-representing vector, and updating both the SVM's dual parameters and the scoring layer will be the most effective way of using kernel methods.

On the topic of Kernel Methods, we argue that the symbolic encodings might not necessarily perform bad if we use kernel methods on the encoded TCRs. Yet, this introduces a bias to the usage of symbolic encodings. This is because we are introducing more information to the symbolic encoding's feature space by using kernel methods, which may make the classification problem easier for symbolic encodings.

5.4.5 Usage of TCR-BERT

As we are unable to create a convincingly generalisable performance when we use TCR-BERT as the upstream encoding model, we propose the use of an autoencoder to reduce the dimensionality of the feature space. This can reduce the trainable parameters in the downstream classifier.

Autoencoders are a pair of neural networks that are trained at the same time. The encoder takes in the raw embedding of 768 dimensions and downscale it to a lower dimension, whilst 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 has a similar representation of the TCR as the original output from TCR-BERT. Note that this vector will not be as expressive as the output from TCR-BERT.

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 exact same TCRs that we are going to use for training the downstream classifier is not going to be a cause of data leakage as long as we do not include the test set into training the autoencoder. This is due to the fact that it is unnecessary for the encoder to know the specificity of the TCR as its end goal is to encode TCRs onto a lower dimension. Hence, the encoder never know the specificity of the TCR, thus cannot assign spatial locality for TCRs with same specificity.

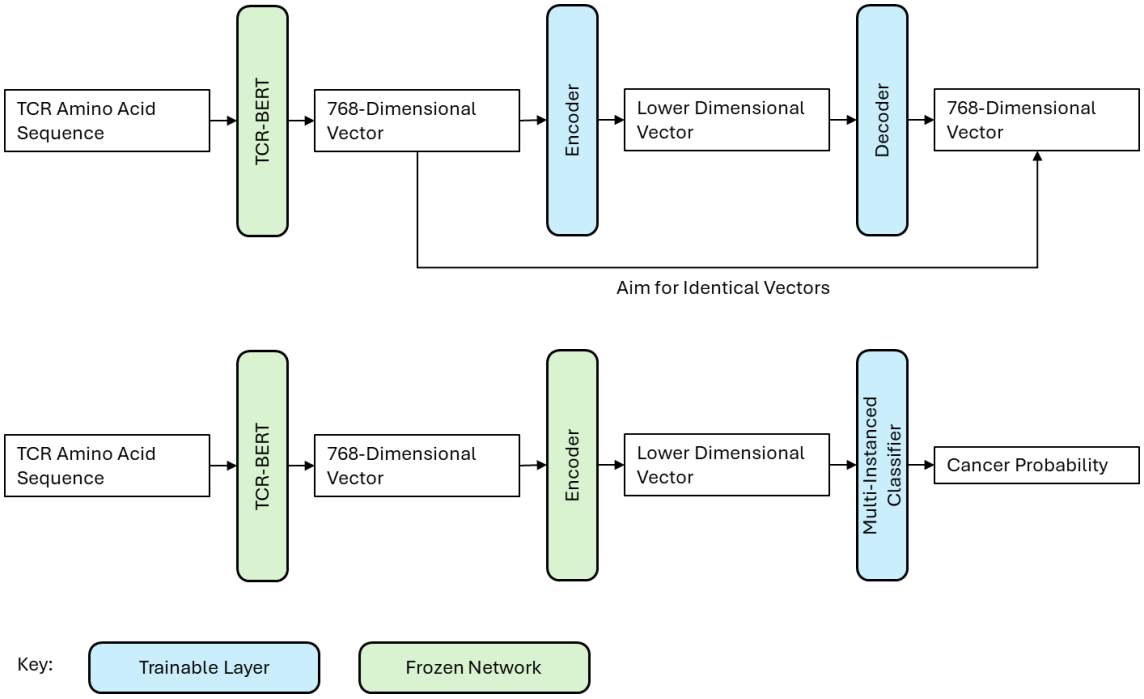

Figure 5.4: TCR-BERT's Encoder Training Paradigm

5.4.6 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 in 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 that we used, utilising them to encode the matrix that represents the CDR3 sequences into a vector could be a better way than simply taking an average.

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

Chapter 6

Conclusions

We conclude this report by objectively reviewing how much we met the goals as set in our Research Aims in Section 2.3. We will also summarise our achievements in this research project as well as summarising 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 through the use of large language models' embedding hyperspace provides a more robust and better performances than using physico-chemical properties.

We showed this through the use of a simple 2-layered neural network, where we spotted 3 significant differences between the performance of downstream classifiers that uses subsymbolic and symbolic encoding spaces.

1. The classifier which uses TCRs encoded from physico-chemical properties is unable to descent in training loss as effectively as the model which takes in large language model embeddings. This demonstrates that either there is no minima to reach in this optimisation problem, or the minima is 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. The confusion matrices for the best checkpoint of symbolic encodings' downstream models have higher values down the minor diagonal than subsymbolic encodings. This demonstrates that using subsymbolic encodings reduce the difficulty of the model in distinguishing cancer and non-cancer patients.
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 the random encoding is not designed to provide any meaning towards each amino acid as opposed to physico-chemical properties which provides high-level meanings towards each amino acid, we expect the random encoding to have the worst performance amongst all symbolic encodings.

   However this has not been observed, whereby the random encoding occasionally performs better than physico-chemical encodings. This suggests an 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, we observed that SCEPTR does not assign spatial locality by TCR specificity. This can be concluded as different training instances' best performing checkpoint has significantly different model parameters, which can be verified by the cosine similarity and euclidean distance. We showed that in 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 are better performing 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 as opposed to the symbolic encodings. This is because TCR-BERT's downstream classifier had too much trainable parameters when compared to the training data. On the other hand, SCEPTR, a model which casts TCRs onto a lower dimension space, performed extraordinarily, whereby one of its downstream models had a 100% AUC in the test set and the evaluation set, 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 than alpha chains, which is in line 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 that is trained on SCEPTR's encodings have achieved state-of-the-art AUC when we average out its AUC across the 10 training instances, where the testing AUC and evaluation set AUC are 95% and 98% respectively on early stages of non-small cell lung cancer.

Although it is arguable that we did not use the same dataset as previous works, the simplicity of our model as well as extreme results such as a 100% AUC on both the test set and evaluation set has showed robustness in our work.

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

6.2 Summary of Future Works

To further our research in this paper, we have proposed 6 different directions as the future works. These 6 directions are in 3 directions: improving the accuracy and predictability of our algorithm in TCR cancer prediction (Point 1, 5), evaluating whether subsymbolic encodings are more expressive than symbolic encodings (Point 4, 6) and increasing the robustness of our study (Point 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 in understanding how to better encode TCRs. This could subsequently help the encoding model in creating 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 as well on patients that has a significantly different genetic background. We propose that collecting more data from different patients with a wider background could improve the robustness of the classifier.

3. We propose that we could 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 that is known to be cancer targeting, and another test focuses on making sure the score for the cancer targeting TCR is high.

4. Whilst the usage of the encoding space without further projection demonstrates that symbolic encodings are less expressive than subsymbolic encodings, we argue that it might be possible to cast the symbolic encodings to a hyperspace such that the symbolic encodings could be as expressive, or more expressive than subsymbolic encodings. With the use of the kernel trick, we believe that this is achievable through a combination of deep learning and kernel methods.

5. 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 a lot of trainable parameters when compared against the labels that we have. We propose the use of an autoencoder to downscale the encoded vectors to a lower dimension, which could in turn, fit a classifier with lesser trainable parameters. We believe that we can use the training data for the encoder can be the same as the data that trains the classifier, since there are no data leakage of disease specificity to the encoder.

6. 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 that we could use an autoencoder trained specifically to transform the matrix that represents the CDR3 sequence into a vector. However, we assert that this autoencoder has to capture the language-like structure of amino acids.

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

A.1.1 TCR-BERT

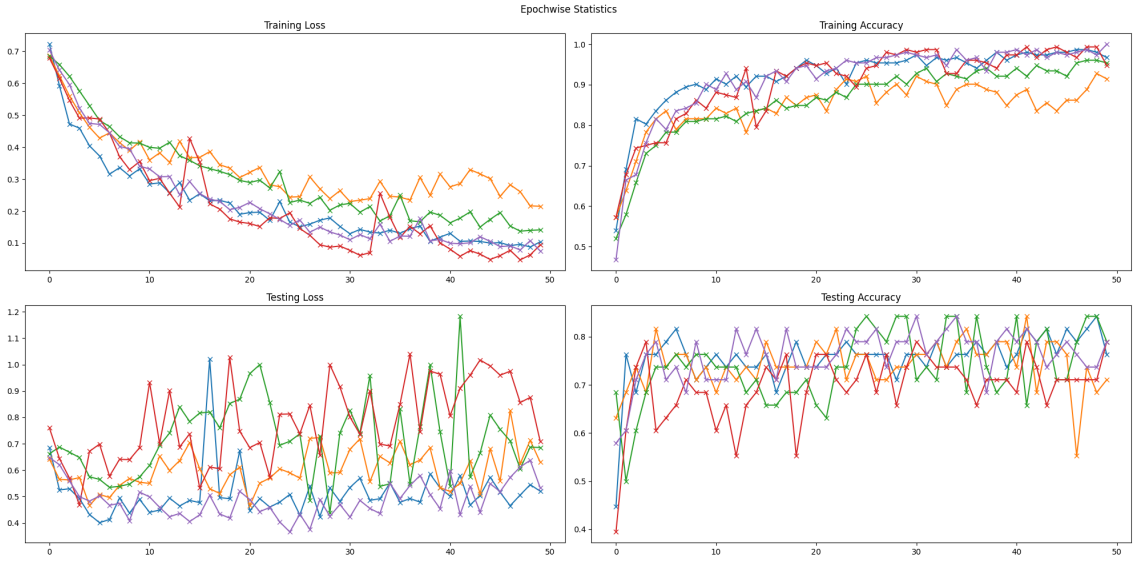

Figure A.1: Loss & Accuracy on Training and Test Sets for each Epoch

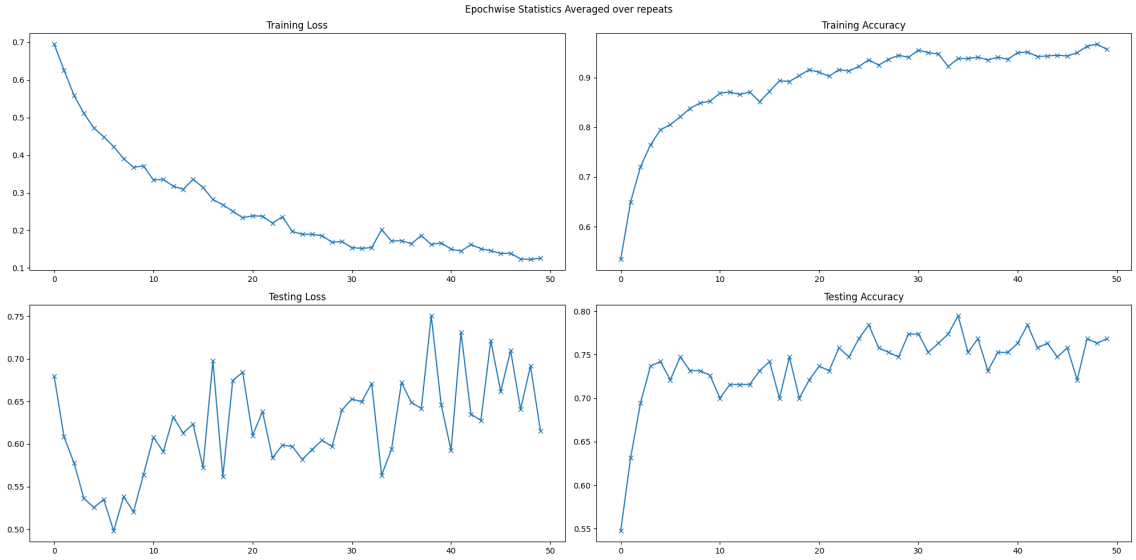

Figure A.2: Loss & Accuracy on Training and Test Sets Averaged on Repeats

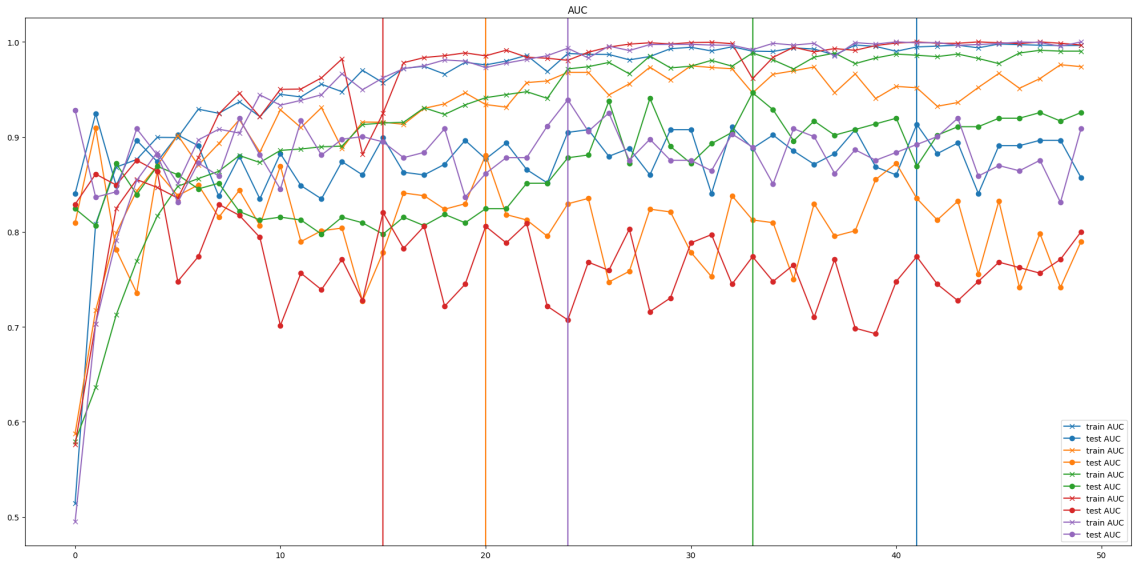

Figure A.3: AUC on Training and Test Sets

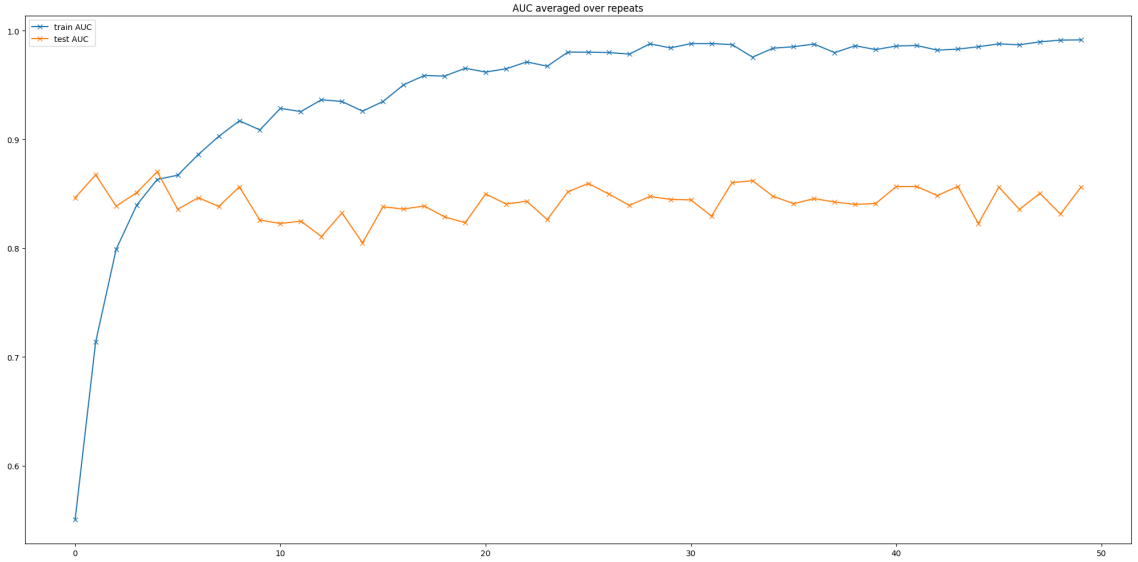

Figure A.4: AUC Averaged on Training and Test Sets

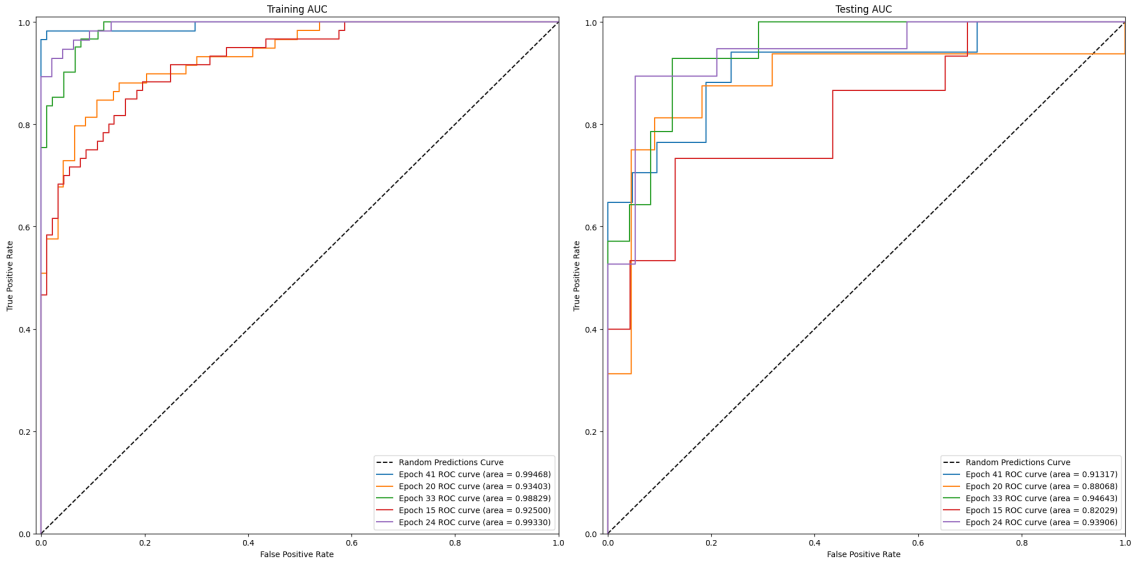

Figure A.5: AUC Curve for each repeated runs

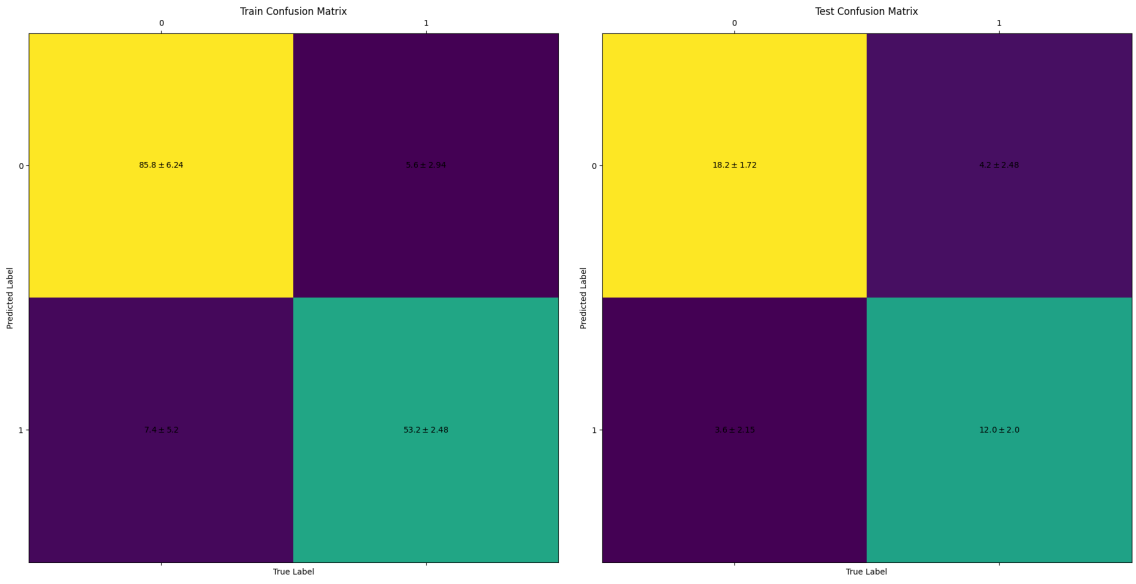

Figure A.6: Confusion Matrix with Mean and Standard Deviation Across Repeats

A.1.2 SCEPTR

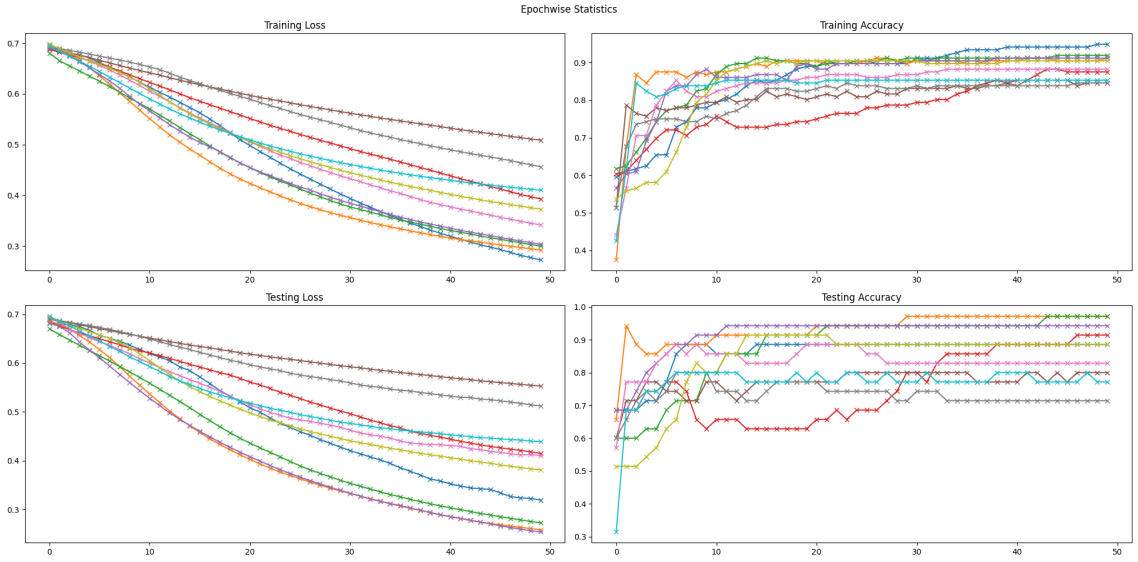

Figure A.7: Loss & Accuracy on Training and Test Sets for each Epoch

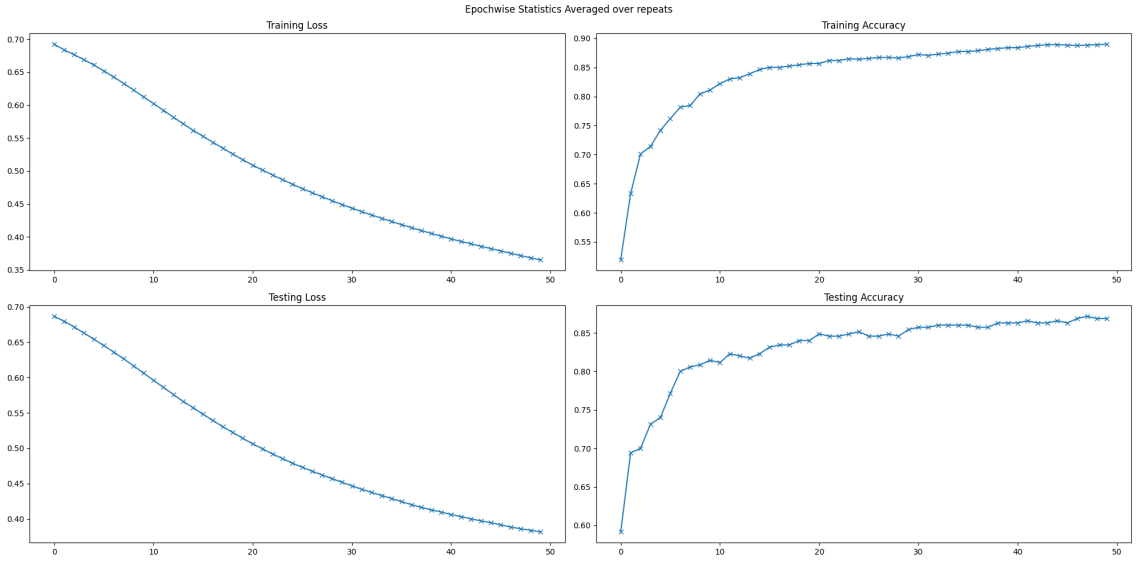

Figure A.8: Loss & Accuracy on Training and Test Sets Averaged on Repeats

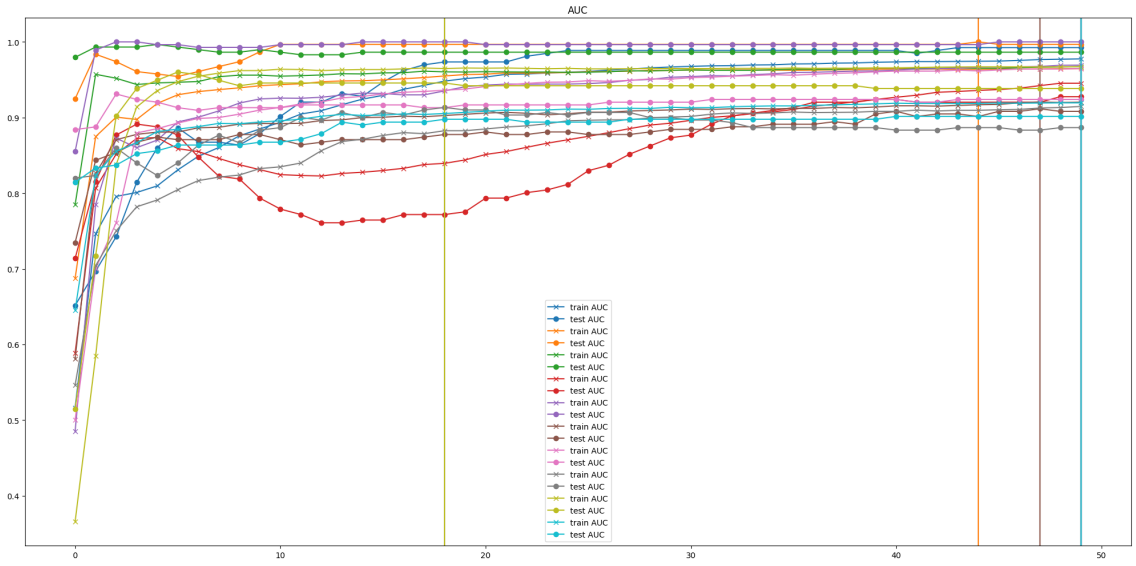

Figure A.9: AUC on Training and Test Sets

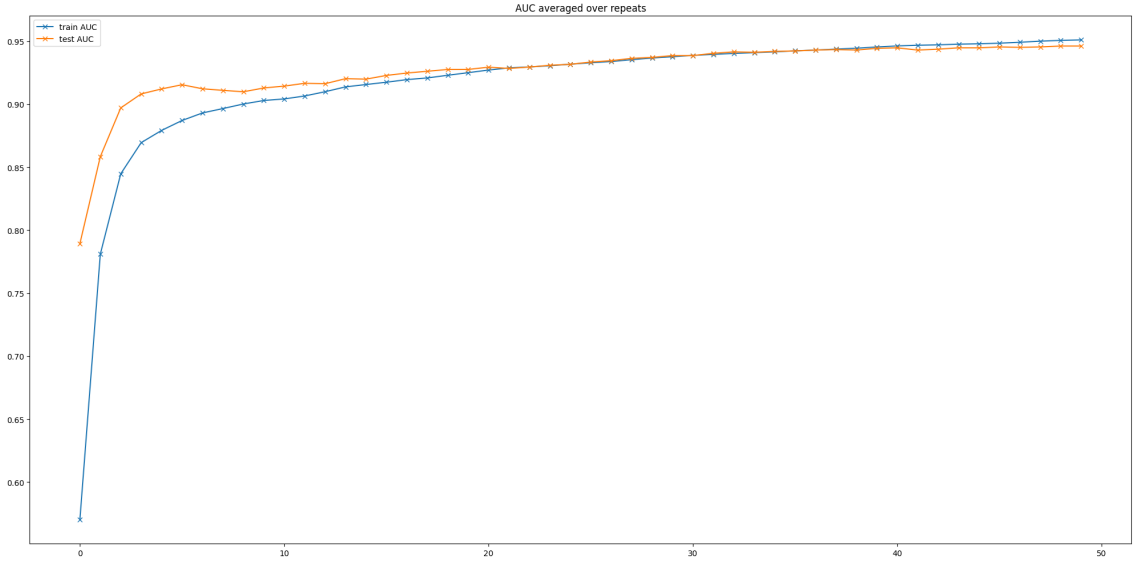

Figure A.10: AUC Averaged on Training and Test Sets

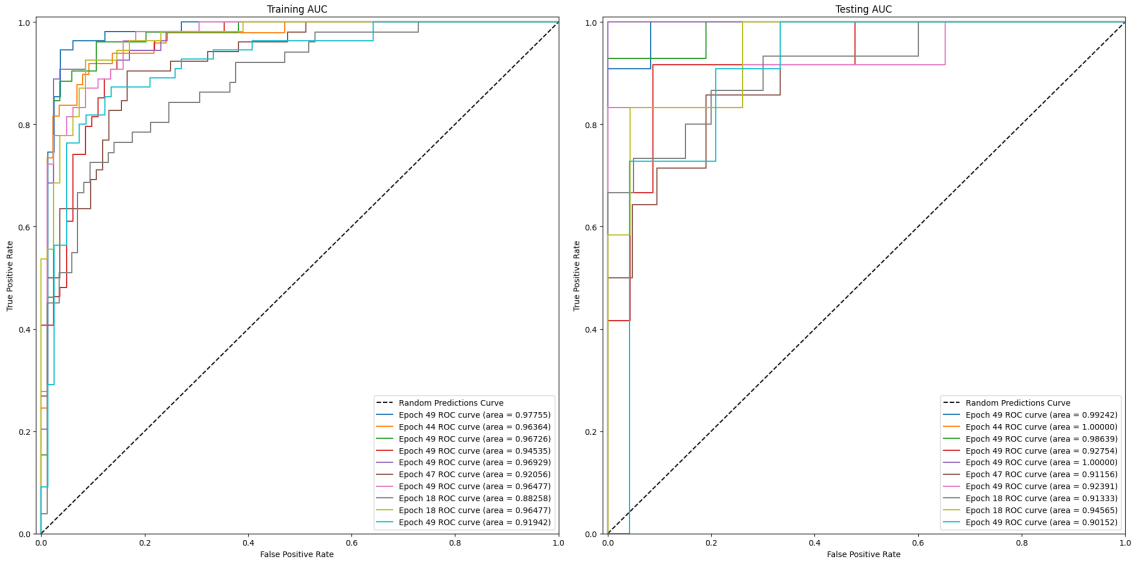

Figure A.11: AUC Curve for each repeated runs

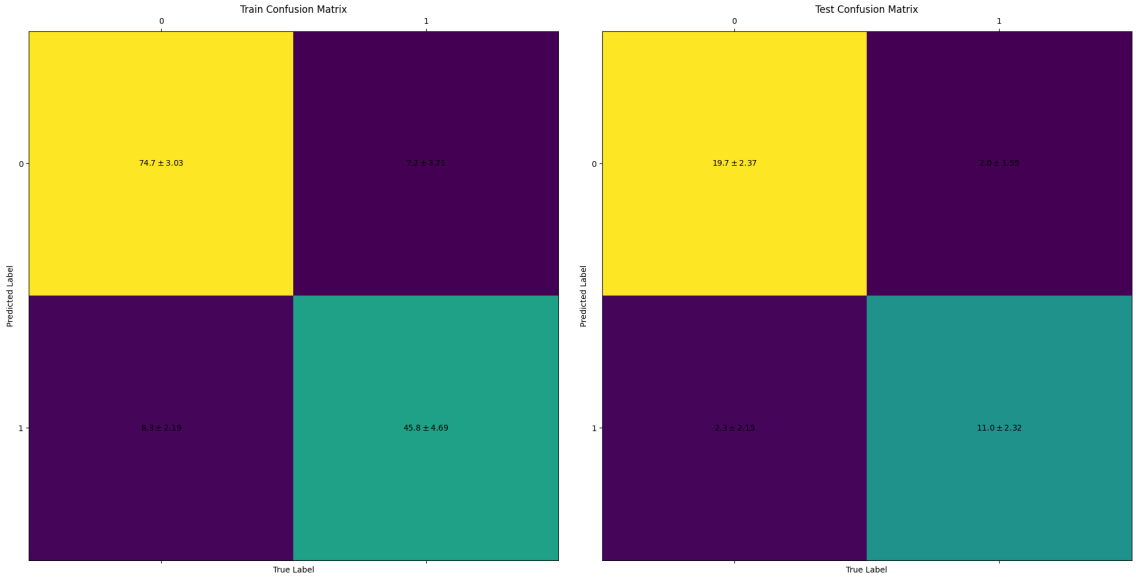

Figure A.12: Confusion Matrix with Mean and Standard Deviation Across Repeats

A.1.3 Atchley Factors

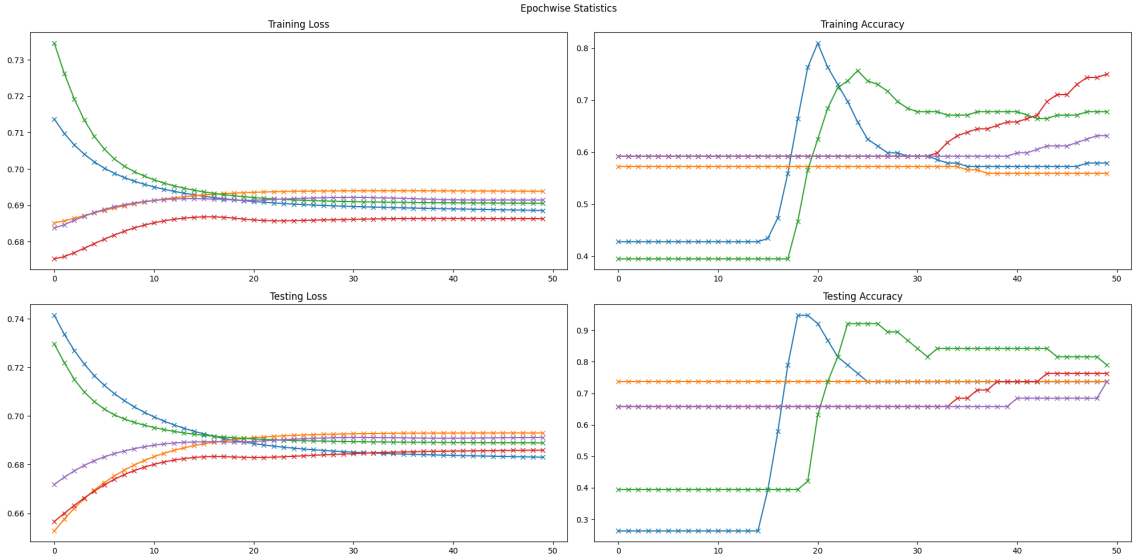

Figure A.13: Loss & Accuracy on Training and Test Sets for each Epoch

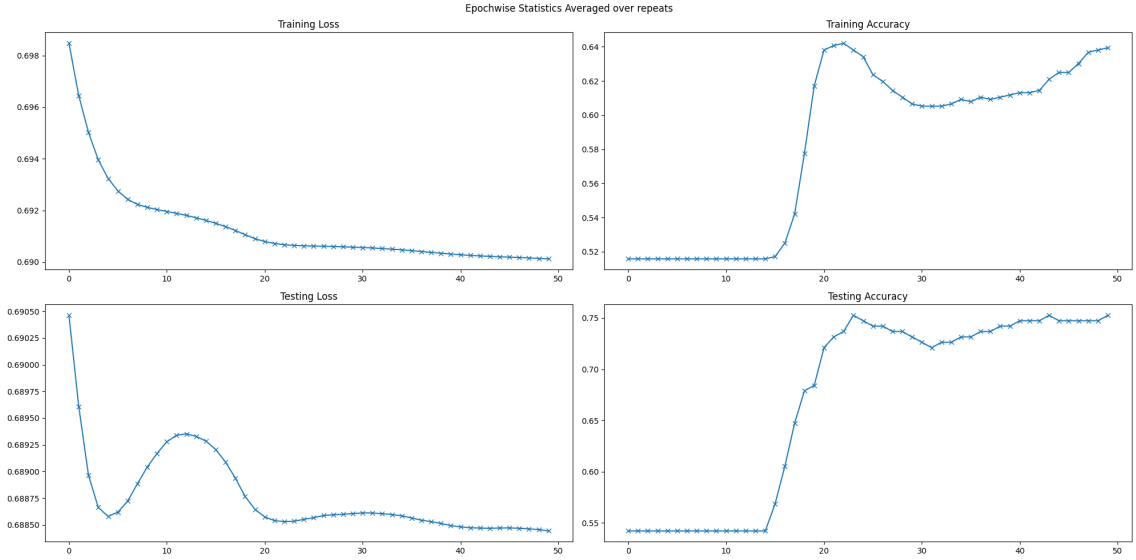

Figure A.14: Loss & Accuracy on Training and Test Sets Averaged on Repeats

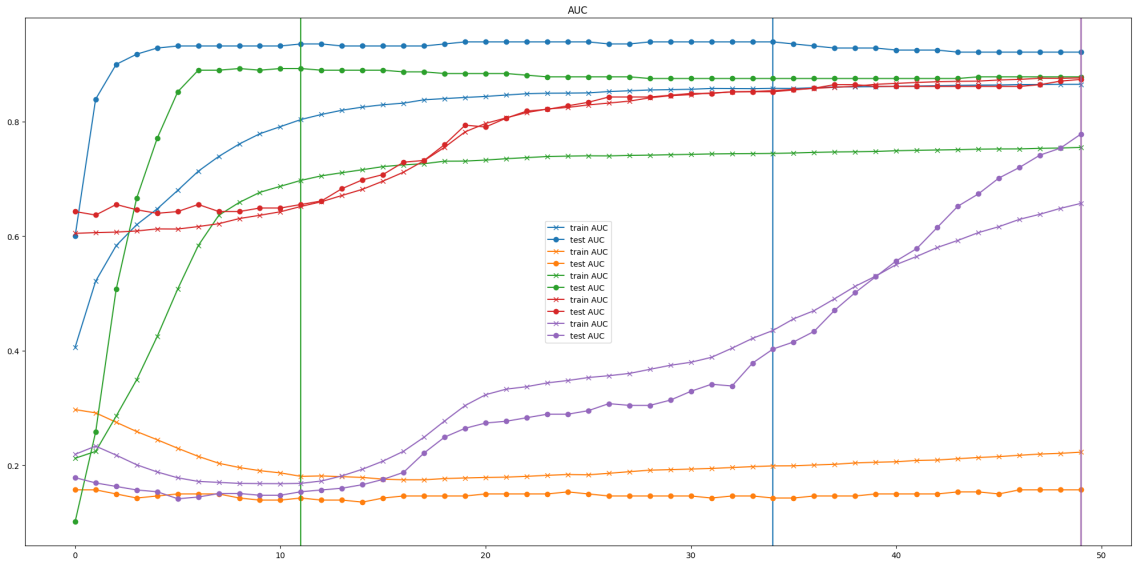

Figure A.15: AUC on Training and Testing Sets

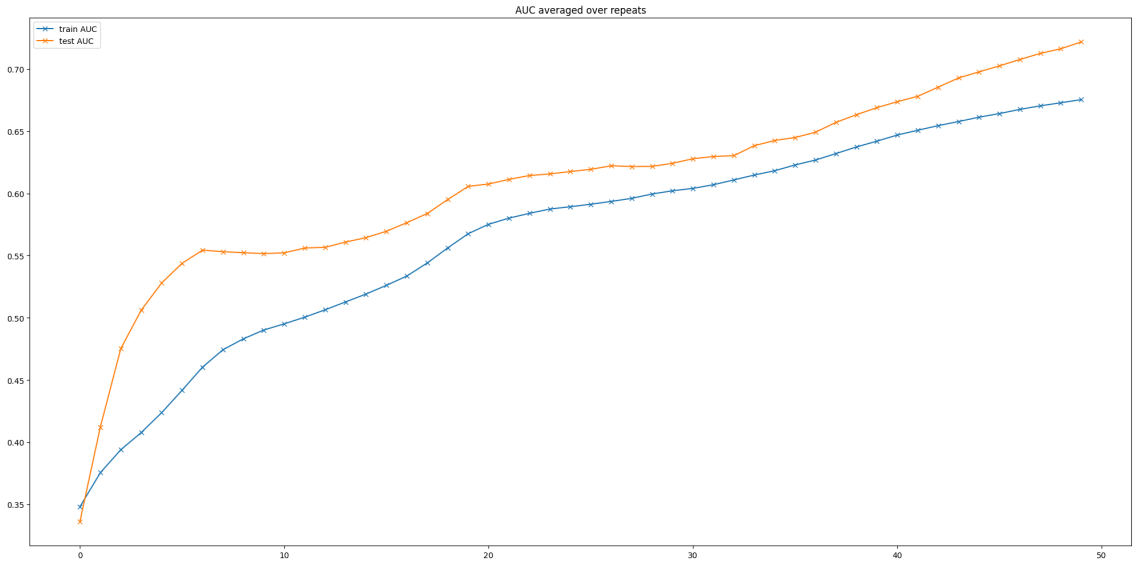

Figure A.16: AUC Averaged on Training and Test Sets

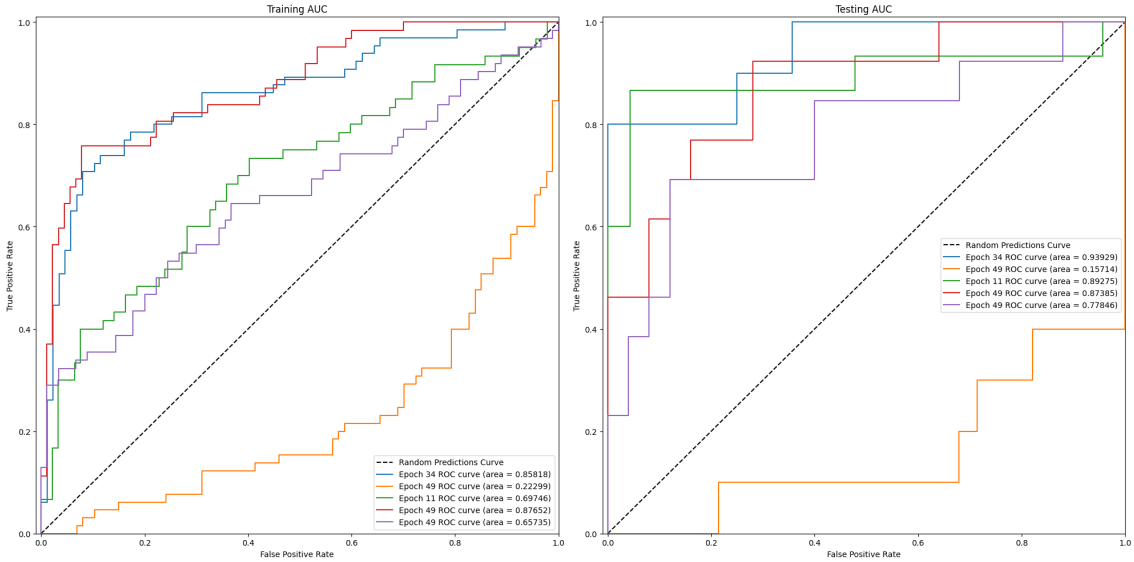

Figure A.17: AUC Curve for each repeated runs

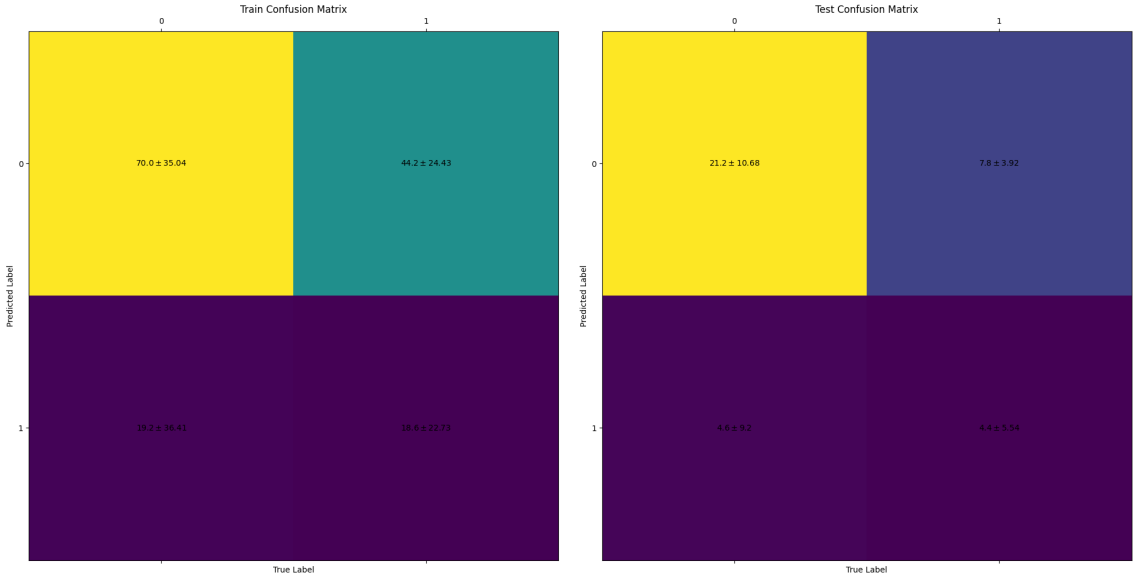

Figure A.18: Confusion Matrix with Mean and Standard Deviation Across Repeats

A.1.4 Kidera Factors

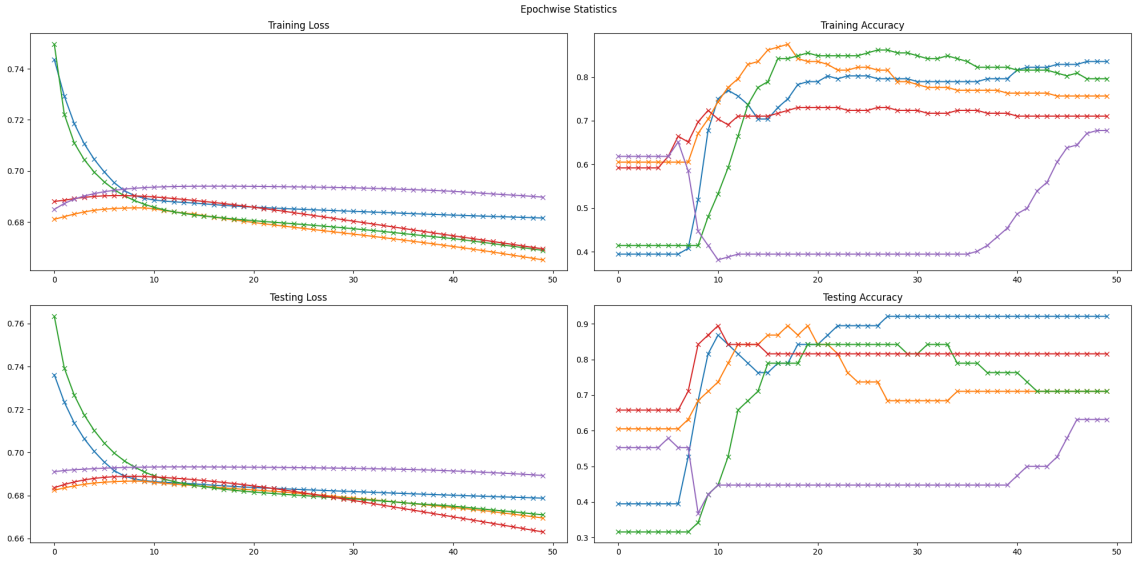

Figure A.19: Loss & Accuracy on Training and Test Sets for each Epoch

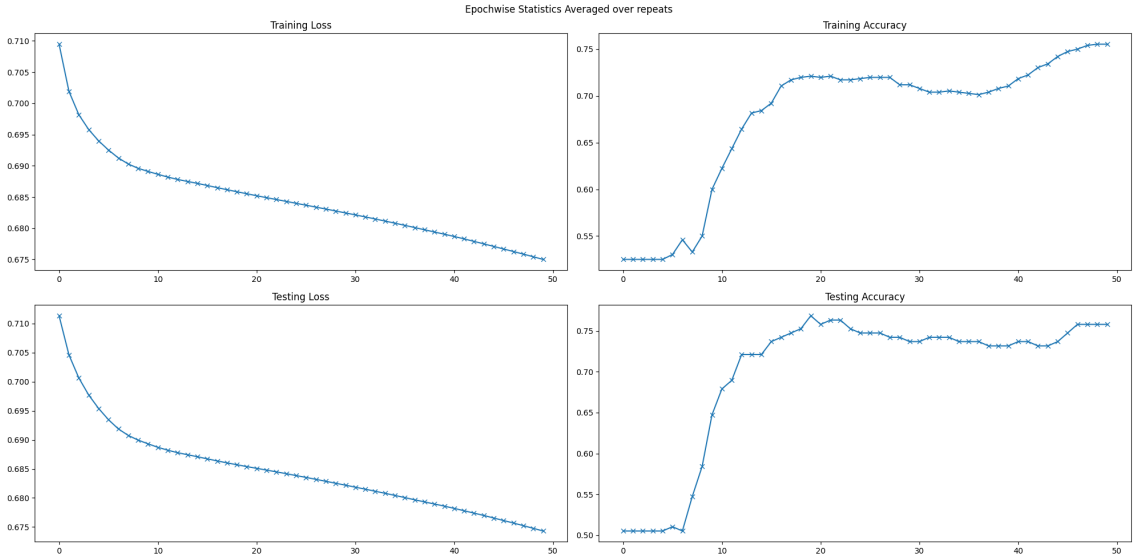

Figure A.20: Loss & Accuracy on Training and Test Sets Averaged on Repeats

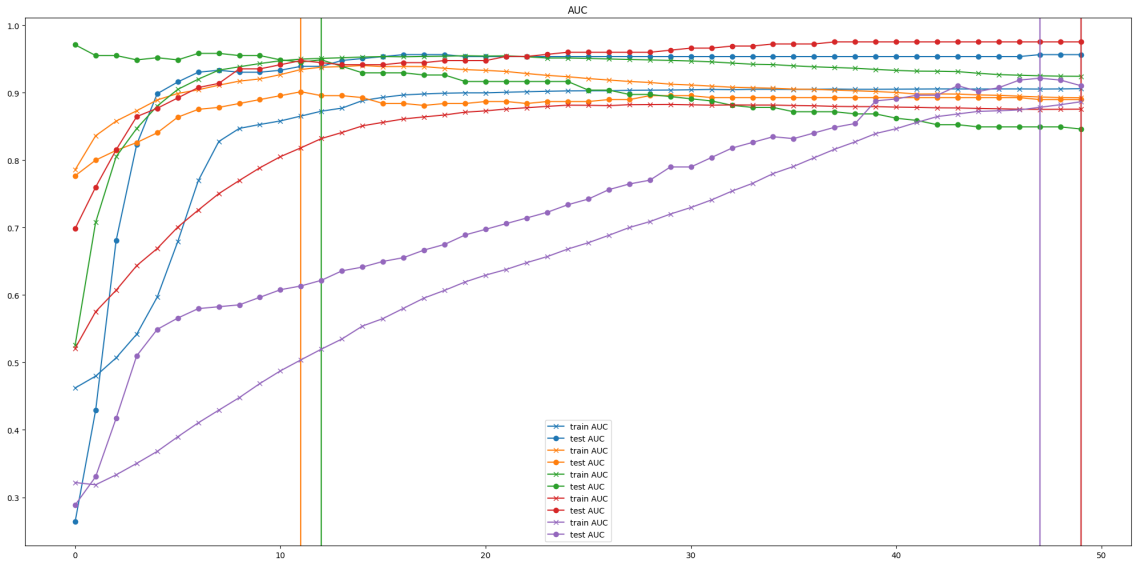

Figure A.21: AUC on Training and Test Sets

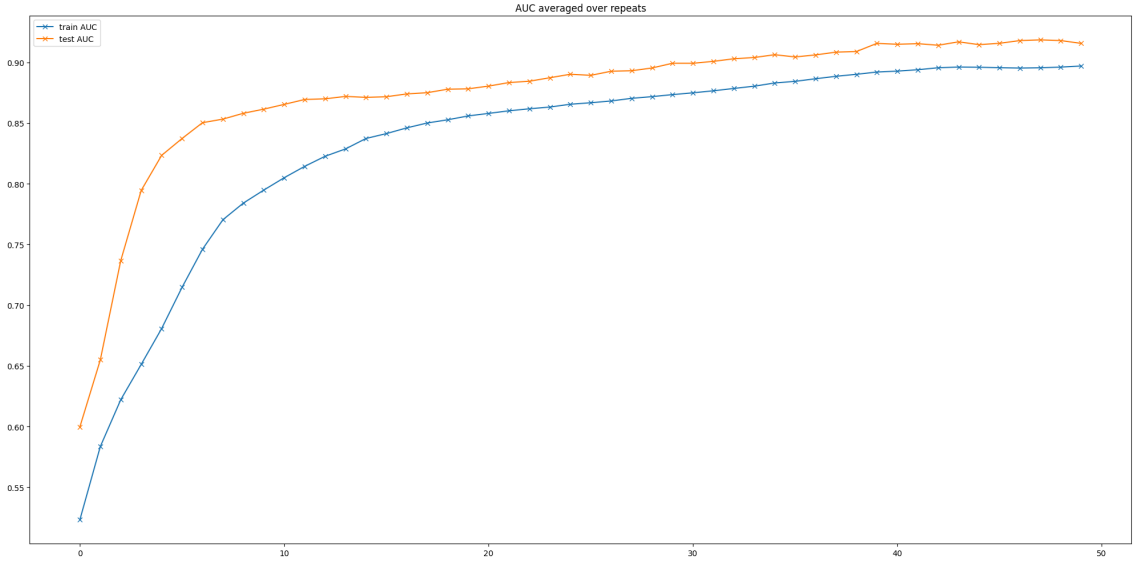

Figure A.22: AUC Averaged on Training and Test Sets

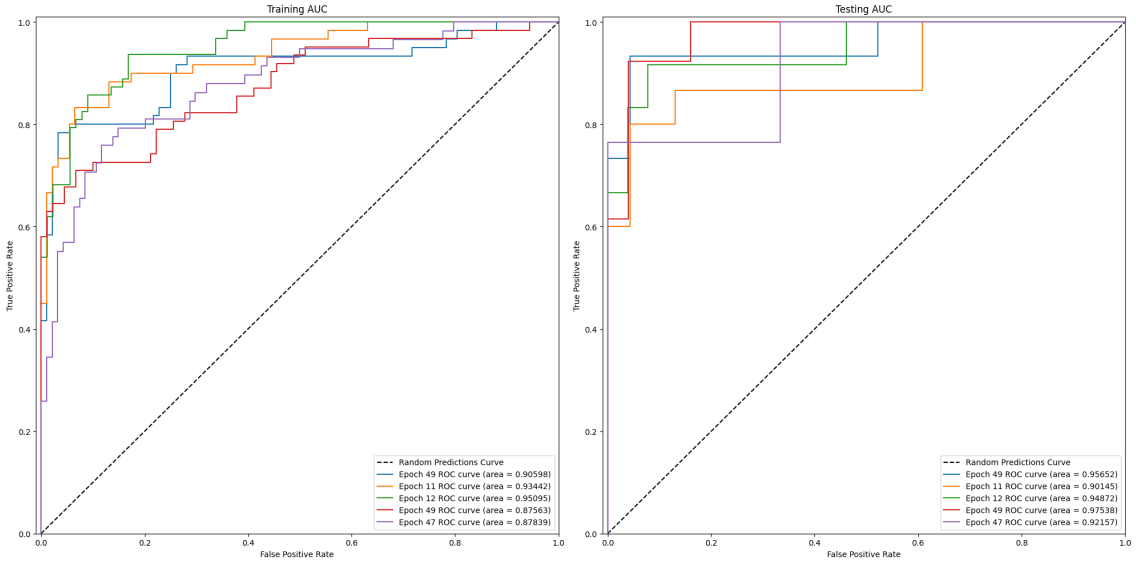

Figure A.23: AUC Curve for each repeated runs

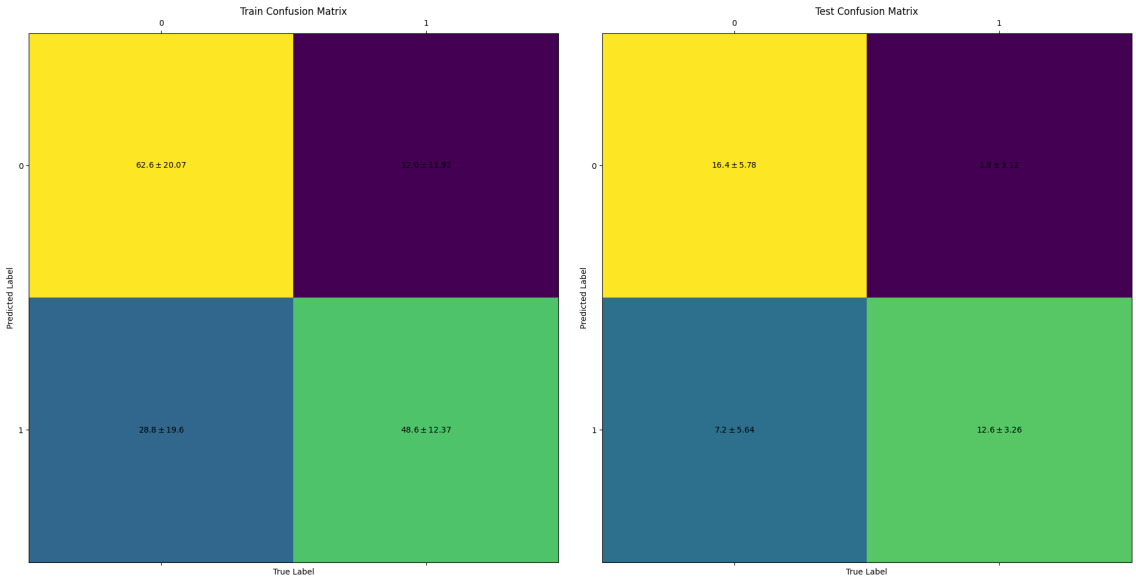

Figure A.24: Confusion Matrix with Mean and Standard Deviation Across Repeats

A.1.5 Amino Acid Properties

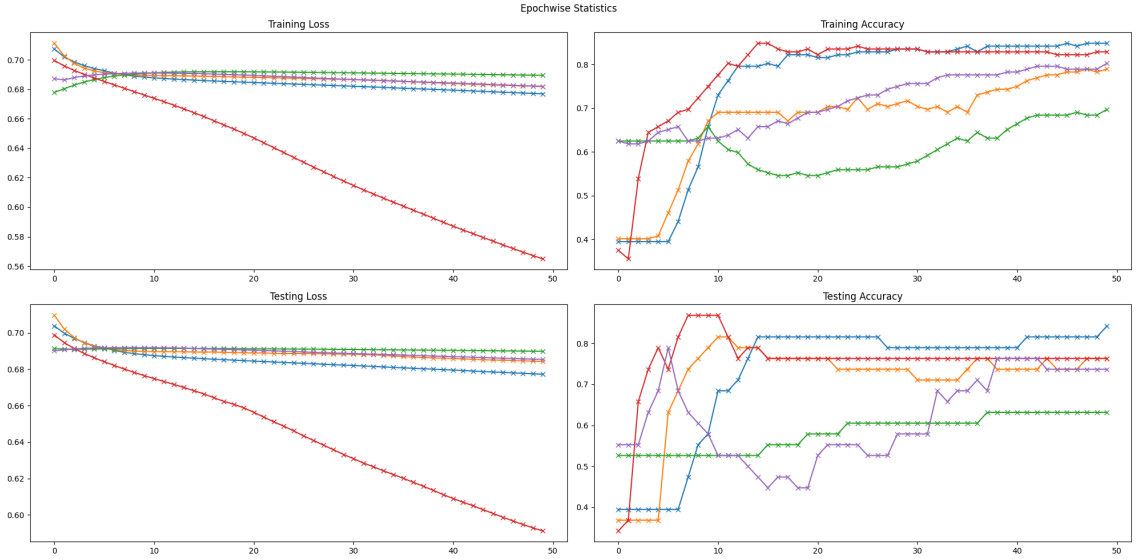

Figure A.25: Loss & Accuracy on Training and Test Sets for each Epoch

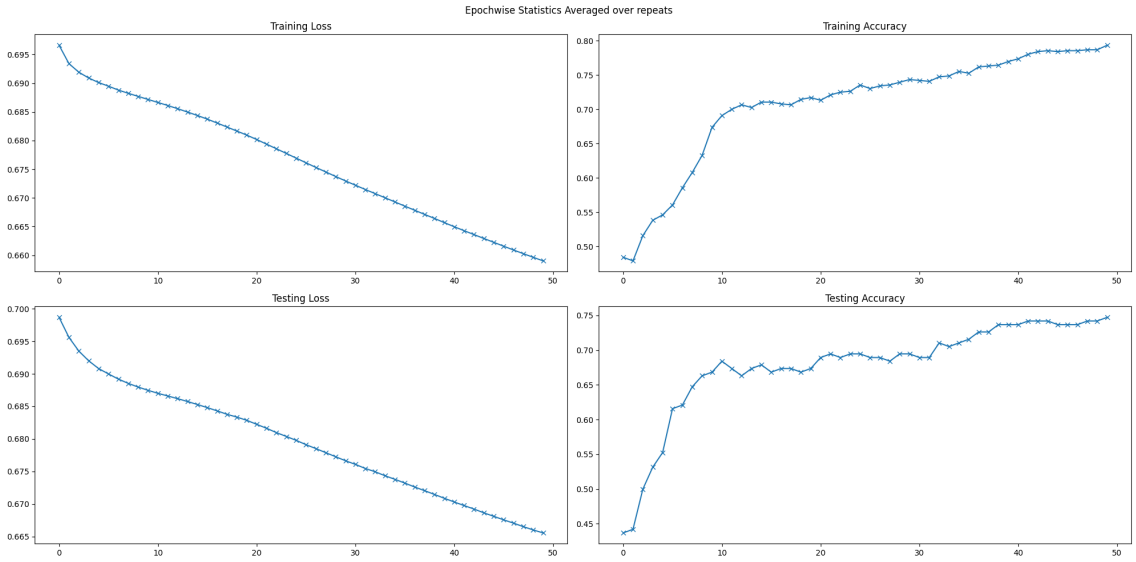

Figure A.26: Loss & Accuracy on Training and Test Sets Averaged on Repeats

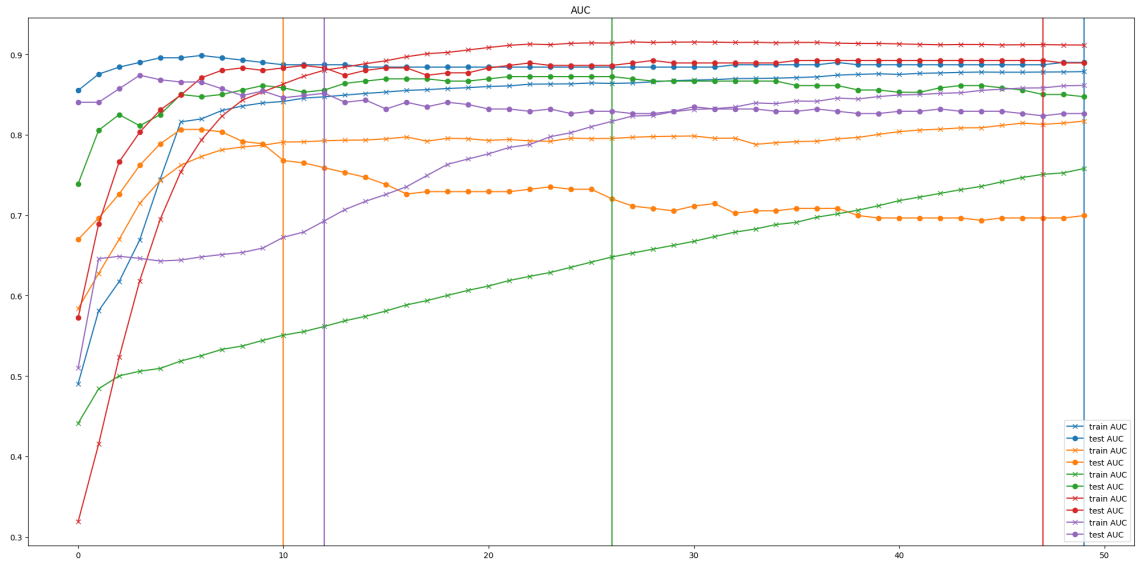

Figure A.27: AUC on Training and Testing Sets

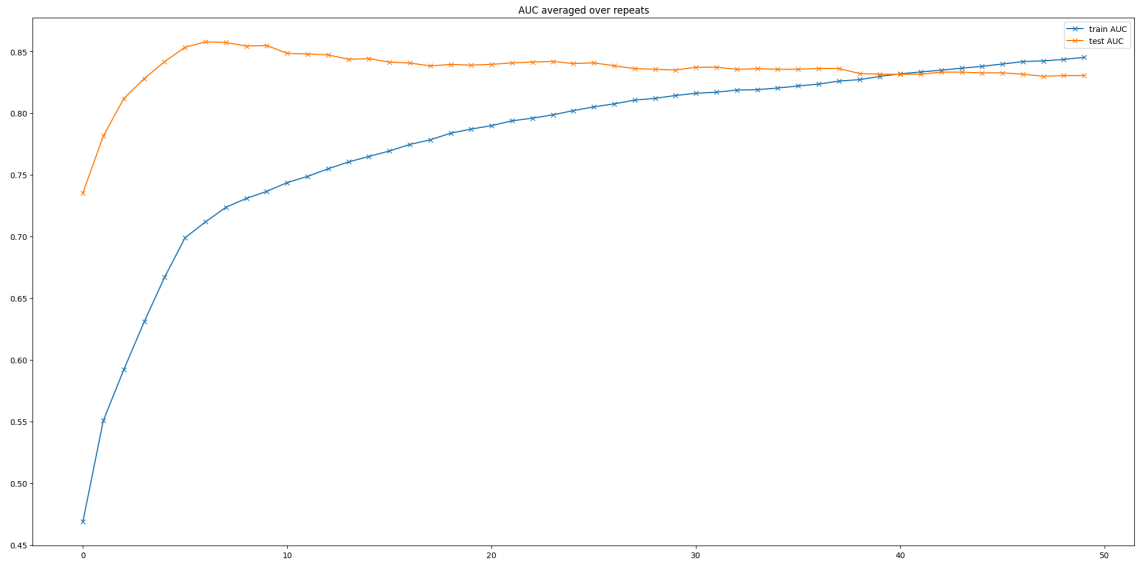

Figure A.28: AUC Averaged on Training and Test Sets

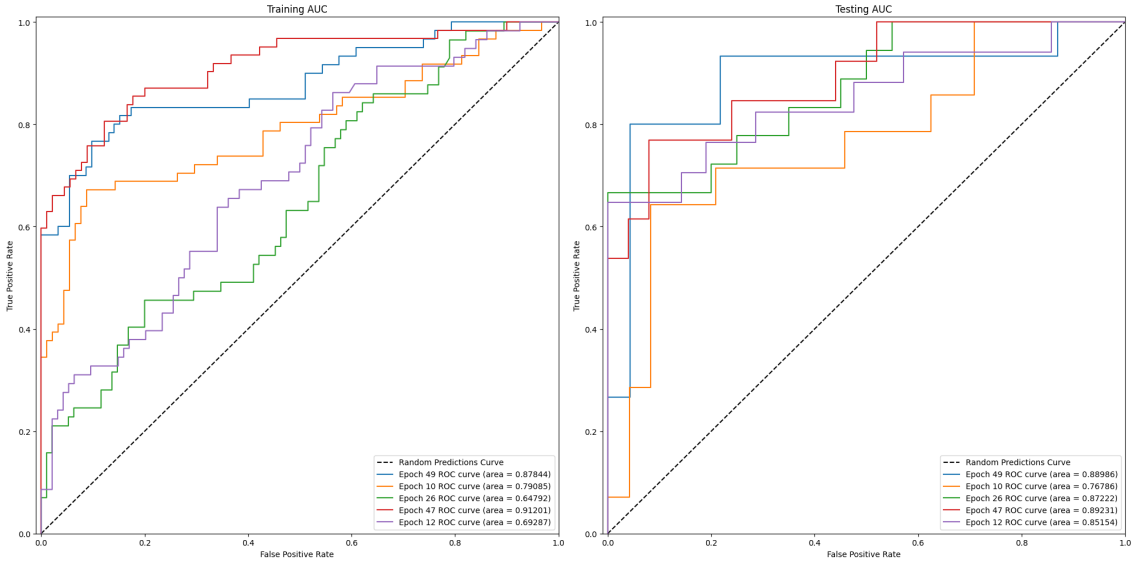

Figure A.29: AUC Curve for each repeated runs

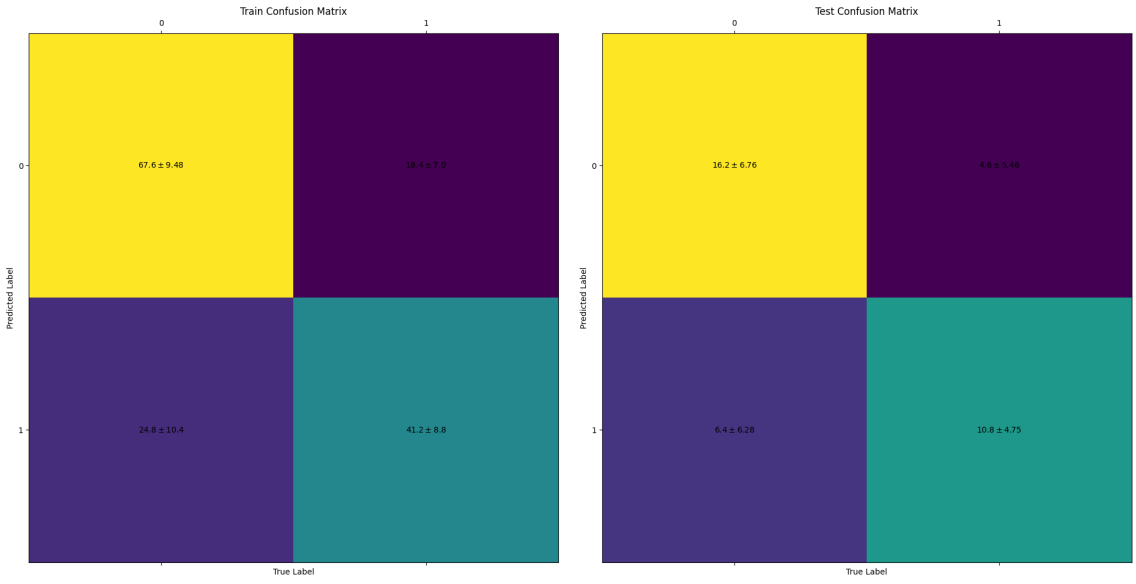

Figure A.30: Confusion Matrix with Mean and Standard Deviation Across Repeats

A.1.6 Random Embedding

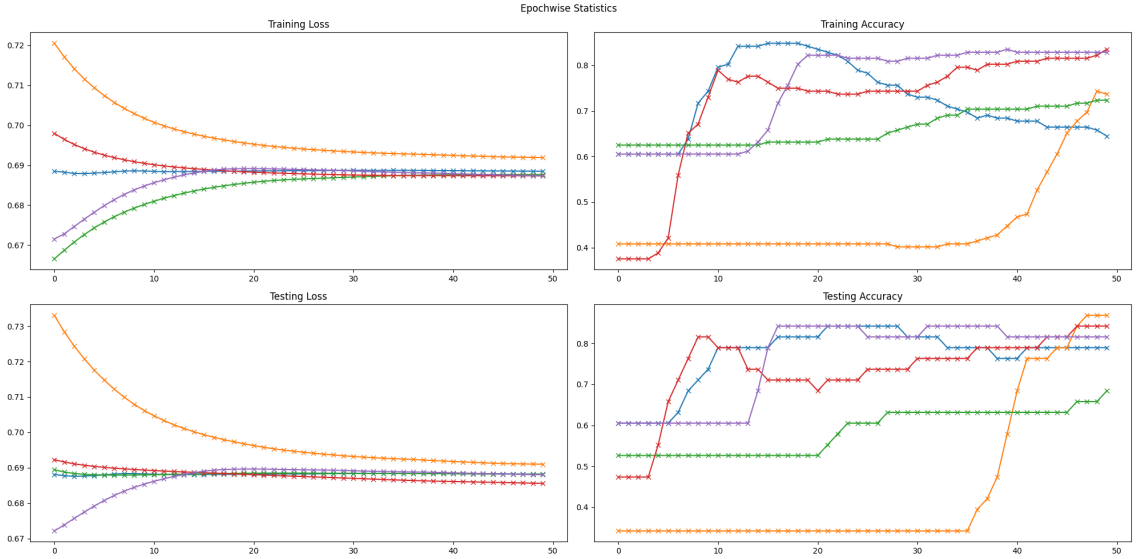

Figure A.31: Loss & Accuracy on Training and Test Sets for each Epoch

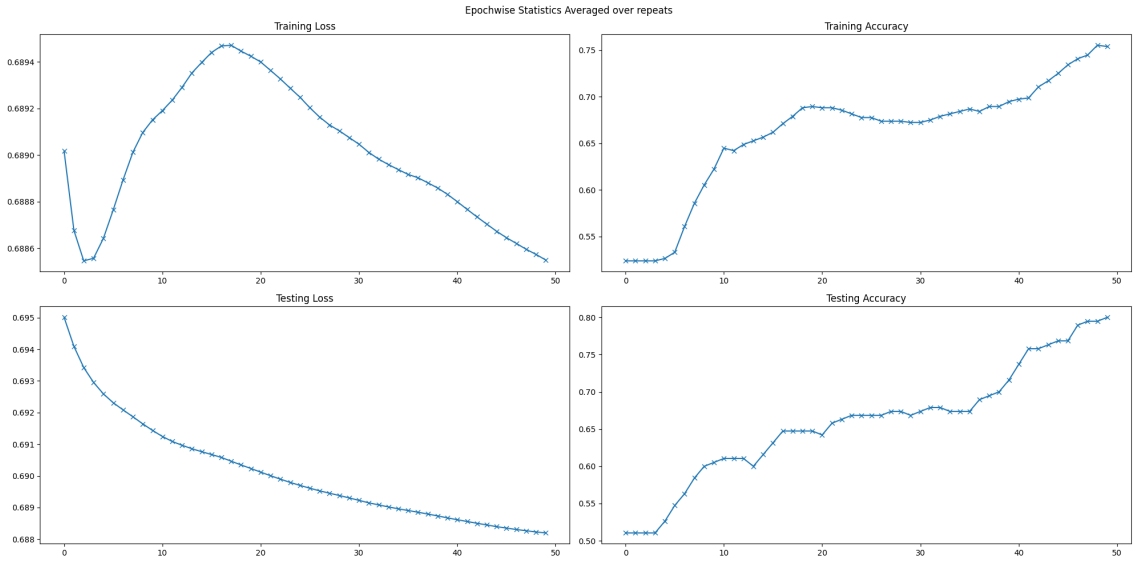

Figure A.32: Loss & Accuracy on Training and Test Sets Averaged on Repeats

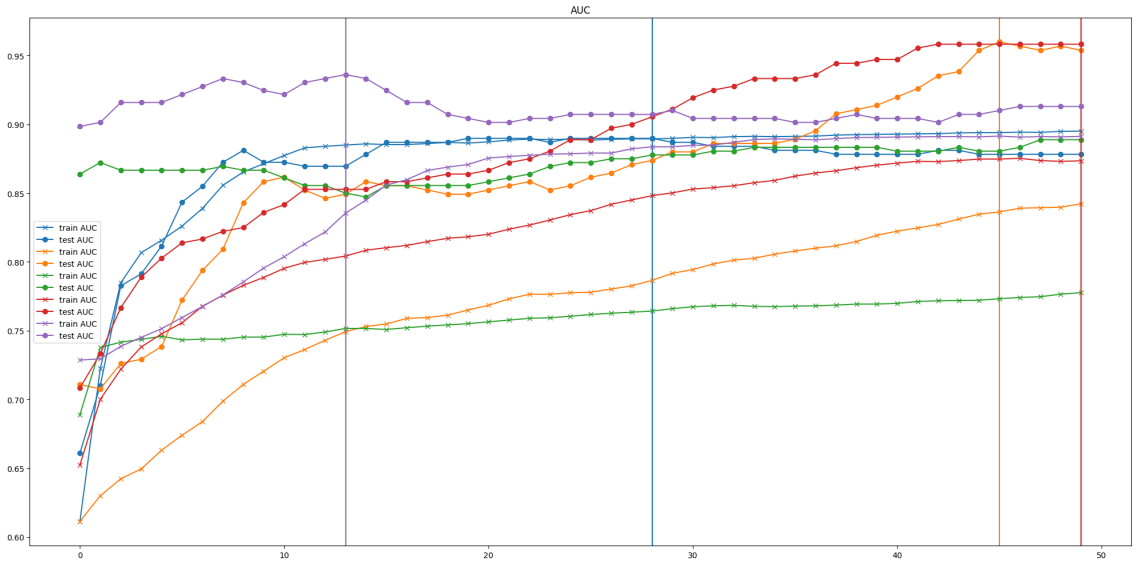

Figure A.33: AUC on Training and Testing Sets

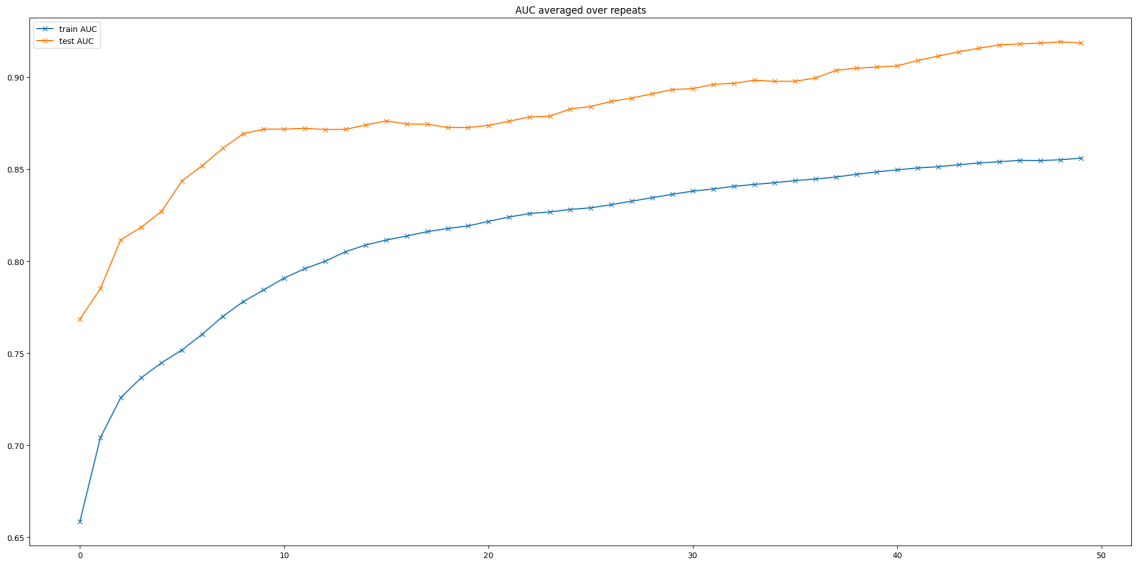

Figure A.34: AUC Averaged on Training and Test Sets

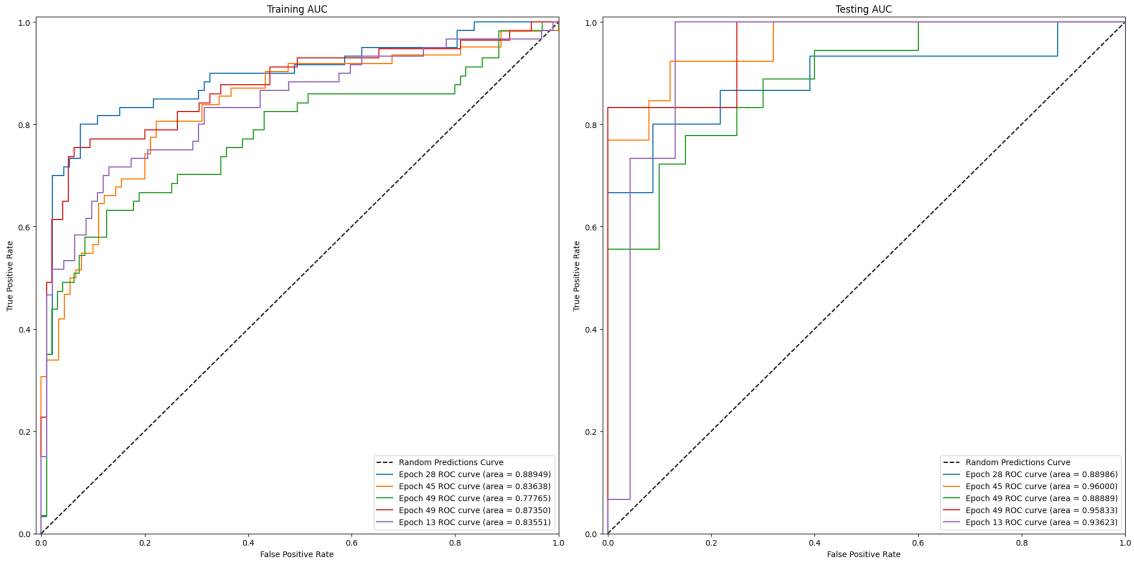

Figure A.35: AUC Curve for each repeated runs

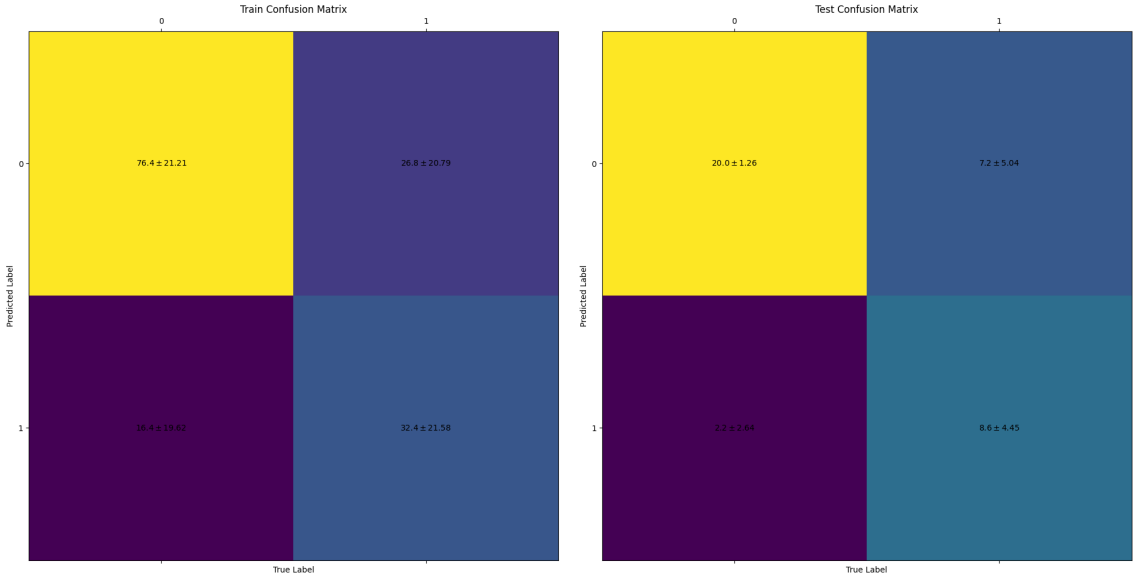

Figure A.36: Confusion Matrix with Mean and Standard Deviation Across Repeats

A.2 Evaluation Set (SCEPTR)

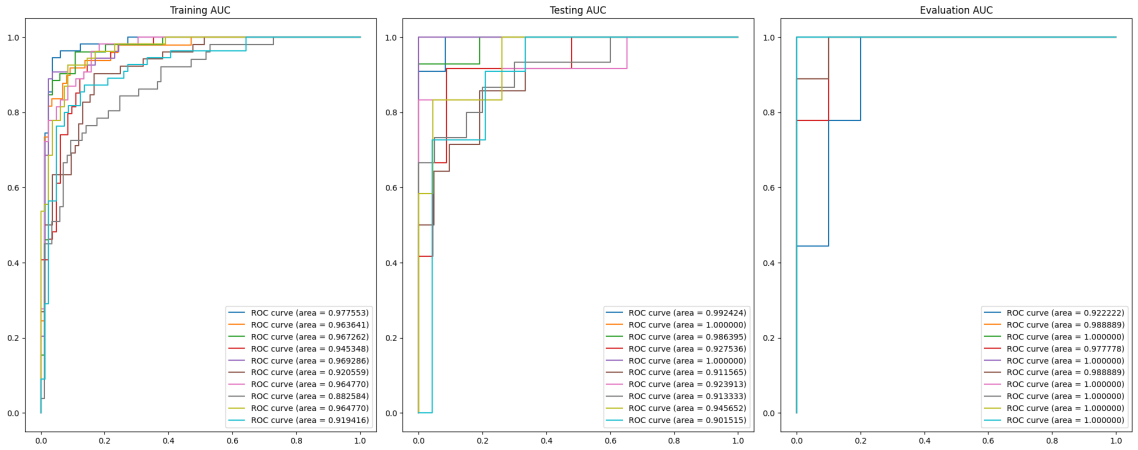

Figure A.37: AUC Curve with Evaluation Set

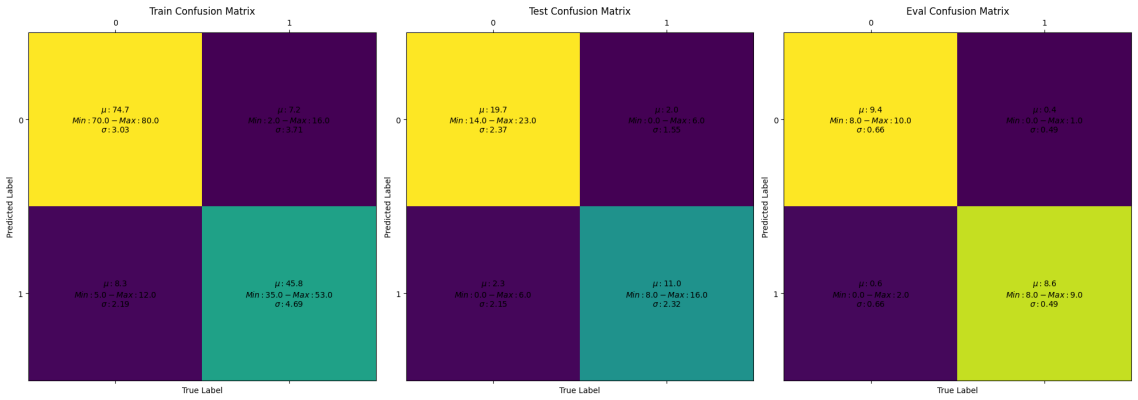

Figure A.38: Confusion Matrix with Evaluation Set

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 same V Call, J Call and CDR3 sequences are 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 are 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 same V Call, J Call and CDR3 sequences are extracted from different patients

A.3.3 Similarity

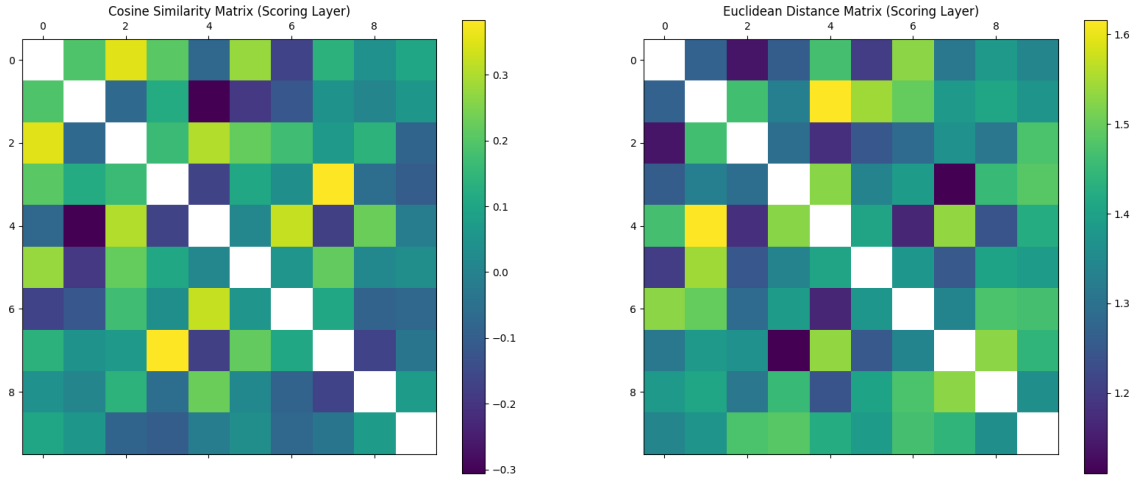

Figure A.39: Similarity between different instances' scoring layer's weights

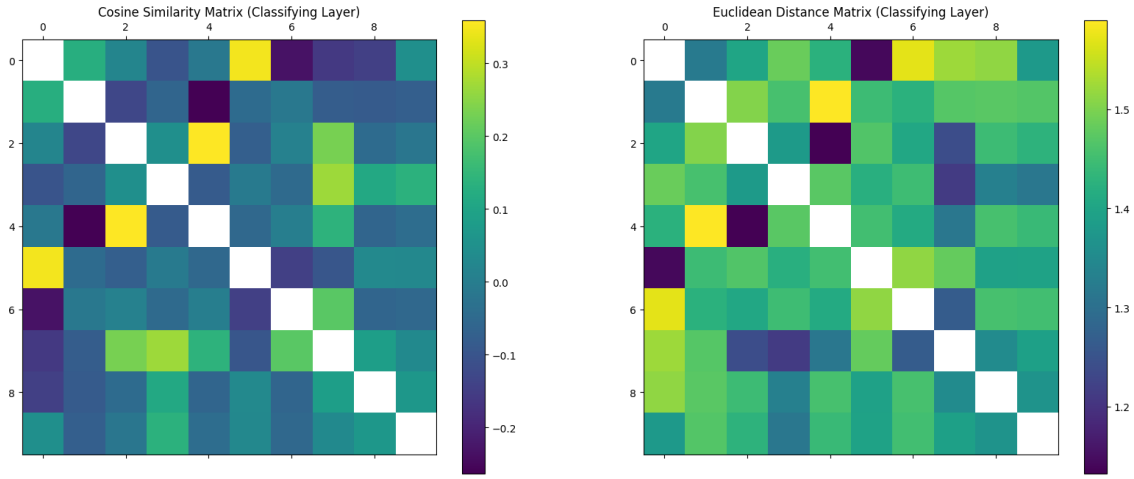

Figure A.40: Similarity between different instances' classifying layer's weights

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 is in the `README.md`

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

Appendix C

Project Plan

Student Candidate Number
Rudy C. Yuen

Title
TCR Repertoire Fine Tuned Protein LLM for Early Cancer Prediction

Supervisors
Professor Benny Chain, b.chain@ucl.ac.uk
Professor John Shawe-Taylor, j.shawe-taylor@ucl.ac.uk
Mr Yuta Nagano, yuta.nagano.17@ucl.ac.uk

Note: All supervisors have received this document on 31st October 2023 and have provided feedback and acknowledged my progress accordingly. I have advised Professor Benny Chain to submit the supervisor confirmation form for this report on behalf of all supervisors.

C.1 Problem Statement

Early Cancer Identification has been proved to decrease the mortality rate for cancer. Current approaches to predict or identify cancer in early stages using Machine Learning utilises Liquid Biopsy data, and recently TCR (TCR) Data is used in this respect.

However, many articles of research that correlates cancer with TCR Repertoires has been widely seen by the academic community as difficult to replicate, and in my perspective, does not provide a grounding basis for the model to learn the underlying structure of the input.

Within all types of TCRs, Blood TCRs have been shown to be hard to be used to predict cancer due to the low level of concentration of T Cells in blood. It is even harder to find the TCR from the Repertoire of TCRs that is different between Healthy and Cancer Patients. Yet, Blood T Cell are the easiest and least invasive to extract which makes the research topic of using Blood TCRs to identify patients in their early stages of cancer interesting.

C.2 Aim

In this project, we aim to create a LLM based classification algorithm for cancer patients using Blood TCRs. The LLM will be a pre-trained LLM using Protein Data, which should hypothetically understand the grammatical structure of Proteins, and this knowledge base can be used to serve as a grounding knowledge for the classification.

Taking this algorithm, we benchmark its result against the exact same experiment but with input data as TCRs sampled from the cancer tissues and healthy patient blood TCRs.

C.3 Deliverables

By the end of the project, we aim to create a Machine Learning Algorithm that classifies cancer patients using Blood TCRs. Dependent on the success, we may write a research paper that details the approach and results. If this project has been shown to be not successful, we aim to demonstrate areas which could be improved or places which can suggest how this project can be refined to complete the task.

C.4 Current Progress

We break down the progress by each chapter of the project type-up:

- **Introduction**
  Completed
  As an executive summary will be provided prior to the introduction chapter to summarise all the results and methodologies, the introduction chapter is relatively short. Currently, the introduction gives a brief overview of Early Cancer Prediction, delivers the project objective and highlights the structure of the whole report.

- **Research and Related Works**
  Completed
  I have completed the review of the research that uses motif enrichment and frequency differences to analyse the differences between TCR Repertoire. Since there are multiple research that states that there are semantic differences between cancer patient's and healthy patient's TCR Repertoire, this proves that it is possible to create a classification algorithm to distinguish cancer patients and healthy patients.

  In Progress
  I am currently reading into different Machine Learning approaches in Early Cancer Prediction using TCRs as a means of reference as to what kind of algorithms could be used to classify the TCR embedding outputted by the Pre-Trained Network.

  I have also identified 2 Pre-Trained Protein LLMs (namedly ProtGPT2 and ProGen) which are available in HuggingFace. I will explore several more algorithms in due course.

  To be Completed
  The effectiveness of grounding in fine-tuning LLMs, and the approaches to the set cover problem which helps correlating TCRs and cancer will be investigated.

- **Methodology and Results**
  There are no progress beyond this chapter yet.

The plans for the remaining of the project are as follows:

End of	Goal
November	Completion of Related Works and Methodology, Identifying some algorithms that might be able to used for Blood TCR Classification
December	Complete the procedures to import the Large Language Models into a Jupyter Notebook, and complete Data Visualisation Functions after obtaining results for one of the proposed algorithms
January	Interim Report, Finished Obtaining Results for the best 3 Algorithms for Blood TCR Classification and their corresponding results for Cancer Tissue TCR vs Healthy Patient TCR
February	Refine Methods based on Results, and building Up more complicated algorithms or models for more generalisable results
March	Complete Project Type-up, submit all work to supervisors by start of April
April	Final Touch-ups on Report, and type-up publication if the project is successful

Appendix D

Interim Report

Student Candidate Number
Rudy C. Yuen

Title
Transfer Learning on TCR LLM for Cancer Prediction

Supervisors
BC - Professor Benny Chain, b.chain@ucl.ac.uk
JST - Professor John Shawe-Taylor, j.shawe-taylor@ucl.ac.uk
YN - Mr Yuta Nagano, yuta.nagano.17@ucl.ac.uk

Note: All supervisors have received this document on March 31, 2024, have provided feedback and acknowledged my progress accordingly. Therefore, only one confirmation form has been submitted to avoid confusion.

D.1 Progress made to Date

We break down the progress by each chapter of the project type-up:

- **Introduction**
  This section has been completed and reviewed by BC and YN. If there are no significant changes, this chapter will be left as is until the project is completed for a final review before submission.

  Completed
  Whilst this section has been largely completed last term, I have added content that provides a peek into future chapters and the kind of work will be done. I have also clarified sections within my introduction to make sure all concepts are as clear as possible.

- **Research and Related Works**
  This section has been completed and reviewed by BC and YN. If there are no significant changes, this chapter will be left as is until the project is completed for a final review before submission.

Completed

I have completed an extensive review on several models on cancer prediction using TCR, such as MINN-SA, DeepCat and argued the generalisability of large language models on proteins.

I have also reviewed the strengths and benefits of transfer learning and reason as to why transfer learning is needed for this task. TCR-BERT has been identified to be a large language model that is useful for this task, as it is trained on TCR α and β chains.

- **Methodology and Results**

  Completed

  TCR-BERT will be paired with downstream algorithms for the prediction task. Both variants of TCR-BERT has been downloaded and imported into Python (difficulties explained in the *tbc* section).

  Taking on ideas from MINN-SA as reviewed in the previous chapter, I have designed a neural network architecture that is going to be deployed to the prediction task, where JST has provided some feedback. We have also had a sketch of the training paradigm. The writing up is in progress.

  Implementation-wise, I have gained access to the Chain's Lab's Research Data Server (RDS). I have written Jupyter Notebooks that scrapes data from the RDS to my computer locally, and a data cleaning script.

  To be Completed

  We have downloaded the binary file from HuggingFace but the tokenizer is nowhere to be seen. We will look into how we can find a tokeniser for this task, and if it cannot be found we will use one-hot encoding instead.

- **Discussion & Conclusion**

  No work has been done as this requires results from the previous chapters.

D.2 Pending Work

We review the pending work by seeing to what extent the aims are completed, and how much change has been decided from the Project Plan submitted in October:

End of	Goal
November	Completion of Related Works and Methodology, Identifying some algorithms that might be able to used for Blood TCR Classification. **Changes and Progress** This has been met.
December	Complete the procedures to import the Large Language Models into a Jupyter Notebook, and complete Data Visualisation Functions after obtaining results for one of the proposed algorithms. **Changes and Progress** This has been partially met. Since there are some delays due to the Christmas holiday, I have not been able to get access to the data, instead I have accessed and downloaded the weights of the LLM (TCR-BERT). We previously planned on using Kernel Methods but this has been changed - we have decided not to use the them as changing them to their Multi-Instance Learning Variant will introduce complications which cannot be completed within time. We have also identified a deep learning architecture and training paradigm, however we have not commenced implementing it as of yet.
January	Interim Report, Finished Obtaining Results for the best 3 Algorithms for Blood TCR Classification and their corresponding results for Cancer Tissue TCR vs Healthy Patient TCR **Changes and Progress** Since we are not working on Kernel Methods, we are not going to use 'best 3 Algorithms'. Instead, we are using a baseline approach against the neural network. The baseline approach will search for hyperspheres within the output embedding space from TCR-BERT which consists of TCR sequences from cancer patients only. Considering that we are going to use a Deep Neural Network in this section, we will request access to the GPU Cluster within the department to support this research.
February	Refine Methods based on Results, and building up more complicated algorithms or models for more generalisable results **Changes** We will not be building more complicated algorithms but only refine models by finding the best set of hyperparameters for the neural network.
March	Complete Project Type-up, submit all work to supervisors by start of April
April	Final Touch-ups on Report, and type-up publication if the project is successful **'Typing Up Publication if successful' can be removed from the goals as this does not come into part of the assessment criteria for this project.**

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 & 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] T. Gärtner, P. A. Flach, A. Kowalczyk, and A. J. Smola, "Multi-instance kernels," in *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, (San Francisco, CA, USA), p. 179–186, Morgan Kaufmann Publishers Inc., 2002.