Evaluating Sparse Attention-Based Multiple Instance Learning for Cancer Classification with Pre-Trained TCR Embedding Models

Jan Pytel[1]

BSc Computer Science

Supervisors:

Professor John Shawe-Taylor

Professor Benny Chain

Submission date: April 2025

Abstract

TODO: Summarise your report concisely.

Contents

Chapter 1

Executive Summary

TODO: Executive Summary Page 1

TODO: Executive Summary Page 2

Chapter 2

Introduction

This chapter introduces the research problem, motivation behind it, and the goals of the project. It outlines the scope of the work, the challenges involved, and the contributions made. It also provides an overview of the report structure to help guide through subsequent sections.

2.1 Problem Statement and Motivation

Detecting cancer at an early stage can significantly improve patient prognosis, yet it remains challenging, especially for asymptomatic individuals [2]. Current diagnostic methods often require invasive procedures and often fail to capture early-stage cancers [3]. Recent advances in immunology suggest that analysing immune responses, specifically, the diversity and specificity of T-cell receptors (TCRs), may offer a less invasive and more efficient means of cancer detection [4].

TCRs play a crucial role in adaptive immunity by recognising specific antigens presented by major histocompatibility complex molecules. Due to their very high diversity, understanding and predicting their antigen specificity and relevance in disease response is computationally challenging [5]. Machine learning models, particularly those utilising sequence embeddings and multiple instance learning, have emerged as promising tools for classifying TCR repertoires associated with cancer [5, 6].

Despite advancements, these models often demonstrate issues related to interpretability, robustness, and reproducibility. In particular, attention-based multiple instance learning methods applied to TCR embeddings have shown promise but require evaluation of their effectiveness [7, 8, 9, 10]. This project is motivated by the need to thoroughly evaluate these approaches, verify existing results, and improve the interpretability of model predictions.

2.2 Prior Work and Project Background

This project builds upon prior research conducted by Rudy C. Yuen as part of their MEng Computer Science dissertation at University College London, completed in 2024. Their report, titled *"Multi-Instance Transfer Learning on T cell receptor LLMs for Cancer Prediction"* [1], investigated the use of two TCR embedding strategies: one, simpler, based on physicochemical properties (referred to as *symbolic encodings*), and the other, more complex, using embeddings generated by pretrained language models (referred to as *subsymbolic encodings*). The latter were designed to capture richer biological patterns, such as the spatial locality of amino acids within a TCR

sequence. These embeddings were subsequently used as input to a multiple instance learning classifier for distinguishing cancer from control samples. The aim of their work was to demonstrate the superiority of the subsymbolic encodings in this task. The project was supervised by Professor John Shawe-Taylor from the Department of Computer Science (Faculty of Engineering Sciences) and Professor Benny Chain, along with Doctor Yuta Nagano from the Division of Infection & Immunology (Faculty of Medical Sciences).

The present project continues under the supervision of Professor John Shawe-Taylor and Professor Benny Chain, and directly extends Yuen's research by reproducing their results and further evaluating the performance of the attention-based multiple instance learning model using both symbolic and subsymbolic encodings. In particular, among the latter, this project focuses on the SCEPTR (Simple Contrastive Embedding of the Primary sequence of T cell Receptors) model, developed by Doctor Yuta Nagano, Professor John Shawe-Taylor, Professor Benny Chain, and other researchers from University College London and Princeton University [11]. SCEPTR is a TCR-specific protein language model pretrained using a combination of masked language modelling and autocontrastive learning, allowing it to generate highly informative embeddings from unlabelled TCR data. A detailed overview of SCEPTR and its architecture is provided in Subsection 3.2.3.

2.3 Aims and Goals

This project is driven by the following objectives that together address both reproducibility and interpretability in TCR-based cancer classification.

The first aim is to reproduce and confirm the findings of Yuen's research, which showed that TCR embeddings derived from pretrained language models such as SCEPTR outperform the symbolic encoding schemes in cancer classification tasks. This involves replicating the experimental setup and evaluating performance differences across the symbolic and subsymbolic embeddings in order to verify the superiority of the latter in capturing biologically relevant information.

The second aim is to thoroughly evaluate the effectiveness of the attention-based MIL model in combination with SCEPTR embeddings for cancer classification. The project seeks to assess the overall model performance, robustness to random initialisation, and differences in training datasets. It also explores the challenge of the differences between the two TCR chain types (alpha and beta, further described in Subsection 3.1.1).

Additionally, the project aims to enhance interpretability by visualising attention scores and bag-level TCR representations, evaluating the model's capacity to identify and prioritise the most informative TCRs for accurate cancer classification.

Together, these objectives aim to verify prior results and provide a deeper understanding of how sparse attention mechanisms interact with rich pretrained embeddings in the task of TCR-based cancer prediction.

2.4 Contributions

The main contributions of this dissertation can be summarised in the following points:

1. **Reproduction of Prior Work**
   Building on the framework introduced by Yuen [1], this project reproduces their key experiments, i.e. training a downstream classifier with TCR sequences represented using both symbolic and subsymbolic embeddings. This phase serves to validate the reported findings

and establish a baseline for subsequence evaluations and extensions. This phase involves a comprehensive analysis of the proposed attention-based multiple instance learning architecture, by investigating and documenting model's internal components and the underlying machine learning techniques used.

2. **Interpretability and Model Insights**
   Emphasis is also placed on interpreting model decisions. For example, attention distributions are analysed to to identify "high-scoring" TCRs, and then it is verified whether suc TCRs appear more frequently across cancer patient TCRs than "low-scoring" ones. Additionally, the alignment between the scoring and classifying layers in the model (which include model trainable parameters) is assessed to explore consistency in how the model is "learning". Finally, bag-level embeddings (model's representations of the entire patient's TCR repertoires) are visualised in a two-dimensional space to show the separability between cancer and control samples

Together, these contributions validate key prior findings, while providing a deeper examination of model behaviour.

2.5 Report Structure

The remainder of this report is structured as follows. Chapter 3 (Context) provides an in-depth literature review of prior work on TCR-based cancer prediction. It describes relevant encoding strategies of TCR sequences, and machine learning strategies, in particular, multiple instance learning. This chapter establishes the theoretical foundations essential to understanding the project's methodology. Chapter 4 (Requirements Analysis) defines the problem statement in more detail after having examined the relevant background literature. It outlines the requirements necessary to reproduce and extend the baseline work from [1]. Chapter 5 (Methodology and Results) describes the experimental setup, including data preprocessing steps, model architecture, and metrics collected throughout the training. It also presents the results obtained on the training and evaluation set for various TCR embedding strategies. Metrics of loss, accuracy, and area under the curve (AUC) are reported. Further, Chapter 6 (Evaluation) offers a critical analysis and deeper insights into these findings, discussing their implications, identifying patterns, and addressing potential limitations. Finally, Chapter 7 (Conclusions) summarises the main achievements of the project with comparison to the original goals. It also summarises the limitations and proposes areas for future work.

Chapter 3

Context

The first part of this chapter provides the necessary background and context to understand the relevance of evaluating different TCR embeddings for cancer classification with the multiple instance learning. Firstly, the biological foundations of TCRs and their role as cancer biomarkers are introduced. Next, the numerical representation methods of TCR sequences are discussed, including both symbolic and subsymbolic encodings used in this study. Finally, the methodological challenge of $\alpha\beta$ chain pairing is described, explaining its implications for computational analysis.

The second part of the chapter focuses on a detailed overview of related works. It begins by introducing the multiple instance learning (MIL) framework, outlining its relevance for repertoire-level classification tasks, and reviewing key studies that have successfully applied it to TCR data. This is followed by a discussion of recent developments in TCR sequence embeddings, including TCR-BERT and SCEPTR-two pretrained models this study utilises. Finally, the section presents a detailed summary of the prior work conducted by Rudy C. Yuen, which this project builds upon.

3.1 Background Information

3.1.1 T-Cell Receptors

T-cell receptors (TCRs) are specialised protein complexes located on the surface of T cells, a type of white blood cell that plays a key role in the immune system [12]. TCRs enable T cells to recognise and respond to specific antigens, fragments of proteins derived from pathogens or abnormal cells [13, 14], when these peptides are presented by major histocompatibility complex (MHC) molecules on antigen-presenting cells (APCs) [15]. Upon recognising these peptide-MHC (pMHC) complexes, TCRs initiate intracellular signaling cascades that activate T cells, triggering immune responses aimed at eliminating infected or malignant cells [16, 17].

Structurally, the majority of TCRs (approximately 95%) are composed of two polypeptide chains, alpha (α) and beta (β), which form a heterodimer linked by a disulphide bond [14]. Each TCR chain contains several complementarity-determining regions (CDRs) that directly interact with with antigens [14]. Among these, the CDR3 exhibits the highest variability and is primarily responsible for antigen specificity and binding [15].

The unique specificity of each TCR is established through a genetic process known as V(D)J recombination, which randomly assembles distinct gene segments during T-cell development in the thymus. For the TCR α chain, Variable (V) and Joining (J) gene segments recombine, while the β chain recombines Variable (V), Diversity (D), and Joining (J) segments, resulting in a vast

diversity of TCRs capable of recognising a wide range of antigens [17, 18, 19]. Figure 3.1 illustrates the overall structure and stages of the V(D)J recombination process for α and β chains.

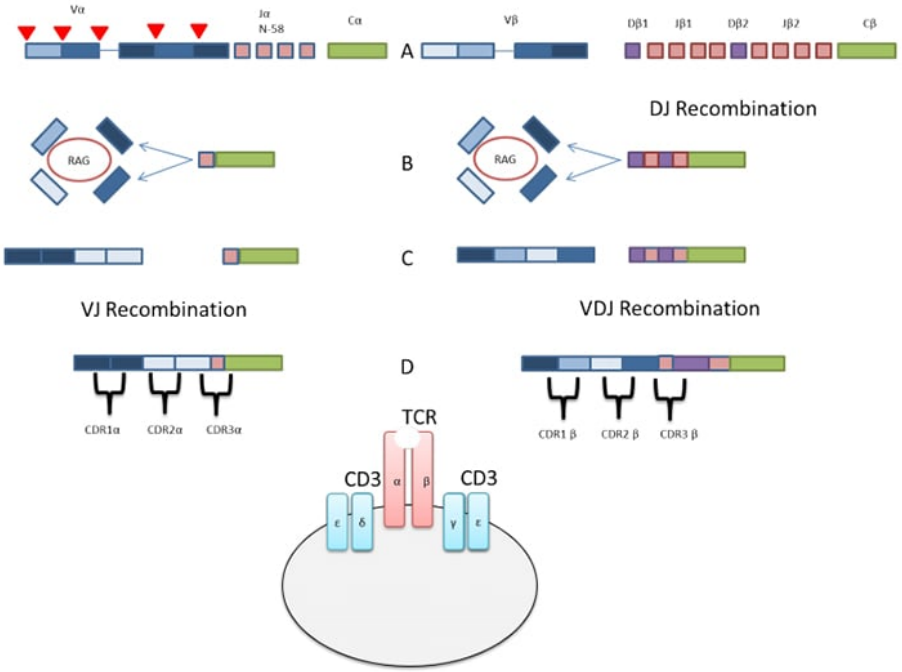


Figure 3.1: Schematic Representation of the V(D)J Recombination Process [14]

3.1.2 T-Cell Receptors as Cancer Biomarkers

A biomarker is defined as a biological characteristic that can measure and indicate a biological or pathological process, helping diagnosis, prognosis, or response prediction to treatment [20]. Due to their specificity for antigens and highly dynamic behavior, TCRs have increasingly gained attention as promising biomarkers for cancer diagnosis [21].

TCR repertoires refer to the entire collection of unique TCRs present in an individual's immune system [22]. They are particularly suitable biomarkers due to their sensitivity to antigen-driven expansion. During tumor growth, cancer-specific neoantigens stimulate selective T cell clones, leading to clonal expansion, which results in an increased frequency of T cells bearing identical TCRs. The composition and diversity of these repertoires vary significantly depending on immune status and disease progression, allowing researchers to use them as indicators of cancer presence and immune response effectiveness [23, 22].

The rationale behind using TCRs as biomarkers in cancer relies on two main repertoire characteristics: clonality and diversity. Clonality (the extent to which a few T cell clones dominate the repertoire) often reflects an ongoing immune response against tumor-specific antigens. For instance, one study found that high clonality among tumor-infiltrating lymphocytes (TILs) correlates positively with better outcomes in immunotherapy-treated melanoma patients [22]. High TCR diversity (variety of unique TCRs present in a patient) can also reflect an ongoing immune response, although its prognostic value varies depending on the cancer type and treatment context [22, 24].

Peripheral blood TCR analysis has further enhanced TCRs' utility as cancer biomarkers due to its minimally invasive nature, repeatability, and ease of sample collection compared to traditional

tissue biopsies. In lung cancer, peripheral TCR diversity and clonal expansions have even surpassed conventional biomarkers such as tumor mutational burden (TMB) in predicting responses to immune checkpoint inhibitors [25]. Additionally, studies on PD-1$^+$CD8$^+$ peripheral T cells indicate that TCR repertoire features from these cells strongly predict clinical outcomes in non-small cell lung cancer (NSCLC) treated with immunotherapy [26].

Several studies have demonstrated successful predictive applications of TCR repertoires across different types of cancer. For example, certain patterns of TCRs have been associated with better outcomes in patients with EGFR-mutant non-small cell lung cancer (NSCLC) who are treated with epidermal growth factor receptor tyrosine kinase inhibitors [27]. In cervical cancer, distinctive TCR repertoire metrics were shown to differ significantly between patients' cancer stages and outcomes, allowing precise monitoring of therapeutic responses and prognosis [24]. Similarly, in lung cancer, a TCR-based immunotherapy response (TIR) index was developed, measuring the overlap between tumor-infiltrating and circulating PD-1$^+$CD8$^+$ TCR repertoires, effectively stratifying patient responses to immunotherapy [28].

Understanding the role of TCR repertoires as cancer biomarkers underscores the importance of TCR diversity and clonality analysis. The insights gained from these previous studies highlight the diagnostic and prognostic potential of repertoire-level immune features across various cancer types and treatment contexts. Those findings provide a strong foundation for TCR-based cancer prediction studies, including the present work, which investigates how different TCR encoding strategies combined with machine learning can capture biologically meaningful signals for distinguishing between cancer and control repertoires.

3.1.3 Numerical Representations of TCR Sequences

TCRs can be represented as strings corresponding to the amino acid sequences of their complementarity-determining region 3 (CDR3), the critical region for antigen recognition [18]. For example, a CDR3 region of a TCR may appear as `CASSPFQGHDYEQYF`. Each character in this sequence uniquely identifies an amino acid, overall encoding the antigen-binding specificity of its respective T cell [29, 30].

For computational analysis and machine learning applications, these sequences must be converted into numerical representations. There are two approaches for encoding TCR sequences in this study, referred to as *symbolic* and *subsymbolic* encodings. Symbolic encodings map amino acids to predefined numerical values based on their physicochemical properties. For example, Atchley factors reduce each amino acid to five numerical values representing its hydrophobicity, size, charge, secondary structure, and heat capacity [31]. Symbolic encodings retain biological interpretability and can be useful for traditional statistical models.

In contrast, subsymbolic encodings are generated by pre-trained machine learning models, which are capable of capturing complex amino-acid sequence relationships beyond predefined biochemical properties. An example is TCR-BERT, a transformer-based model that learns context-aware embeddings by training on large-scale TCR datasets. Using subsymbolic encodings can often improve classification performance. Its limitation is the lack of direct interpretability, making it more difficult to extract biological meanings from the learned representations.

Building upon Yuen's findings, this study evaluates several encoding strategies, including the symbolic methods of random encoding, Atchley factors [31], Kidera factors [32], and Amino Acid Properties [33], along with subsymbolic encodings of TCR-BERT [34] and SCEPTR [11]. The aim is to confirm the reproducibility of prior results and demonstrate the superiority of subsymbolic

encodings [1].

3.1.4 The TCR $\alpha\beta$ Chain Pairing Problem

The antigen specificity of TCRs is defined by the pairing of their α and β chains. Each TCR chain is generated independently through V(D)J recombination, but their combined structure defines the receptor's unique antigen-binding properties. Both α and β chains contribute to the recognition of pMHC molecules. This emphasises the importance of accurately identifying native chain pairings in TCR repertoire analysis [35].

However, standard bulk sequencing techniques generally do not retain pairing information between α and β chains. These methods sequence RNA or DNA extracted simultaneously from thousands of T cells, producing separate lists of α and β sequences without indicating their original cellular pairings. The loss of this pairing information significantly complicates computational modeling tasks, because crucial biological context is missing [36].

Single-cell sequencing technologies have recently addressed this limitation, accurately capturing paired TCR $\alpha\beta$ sequences from individual cells. Studies leveraging these approaches have highlighted the complexity and partial stochasticity of TCR chain pairing. For example, in [37], it was observed that while pairing often appears random, certain combinations, such as `TRAV26-1` paired with `TRBV20-1`, occur at higher-than-expected frequencies, suggesting underlying structural or selective biases. Moreover, genetically identical individuals show notably increased sharing of specific TCR $\alpha\beta$ clonotypes, indicating that genetic factors influence pairing patterns [37].

Despite these technological advances, single-cell sequencing remains costlier and of lower throughput compared to bulk sequencing, limiting its widespread adoption. Many TCR studies relying on bulk sequencing data either use heuristic methods for approximating pairing or analysee α and β sequences independently. Common simplifications include focusing solely on the β chain, due to its greater diversity and therefore predictive power [36, 38].

In summary, the $\alpha\beta$ chain pairing problem poses a significant methodological consideration in TCR repertoire analysis. Although single-cell technologies offer solutions by capturing paired chain information, practical limitations often necessitatee simpler strategies. Due to the absence of paired data in this study, the α and β chains will be analysed separately, with an expectation that the β chain will yield superior performance.

3.2 Related Works

3.2.1 Multiple Instance Learning

Multiple instance learning (MIL) is a form of supervised machine learning where training examples are grouped into sets, known as "bags", and labels are provided only at the bag-level rather than for individual instances within each bag. Each bag is labeled positive if at least one of its instances is positive. Otherwise, the bag is labeled negative. This setup reflects real-world situations where data inherently comes grouped and precise instance-level labeling is difficult or infeasible [39].

MIL is a suitable approach to the TCR-based cancer prediction, due to the structure of TCR repertoires. Each patient provides a TCR repertoire (bag) containing thousands of TCR sequences (instances), out of which only a subset may recognise the tumor antigens. The label "true" or "false" is assigned to the entire repertoire, rather than an individual TCR.

[40] conducted a comprehensive comparative study on various MIL methods for cancer detection using TCR repertoires. They analysed traditional MIL approaches such as mi-SVM and mi-

Net alongside modern deep learning methods and found that the deep learning-based approaches significantly outperformed traditional methods in distinguishing cancerous from non-cancerous repertoires. Their evaluation on different cancer types demonstrated that convolutional neural networks (CNNs) integrated into MIL frameworks provided particularly robust predictive performance, suggesting that deep architectures better capture subtle motifs and interactions within TCR sequences.

Building upon the success of deep learning in MIL, [41] introduced DeepLION, a deep multi-instance learning model designed specifically to identify cancer-associated TCR sequences for accurate cancer detection. DeepLION employs an attention mechanism to weigh each TCR sequence according to its predicted relevance to cancer diagnosis, improving the interpretability and accuracy of the predictions compared to previous methods. DeepLION achieved an Area Under the Curve (AUC) exceeding 0.94 in distinguishing healthy from cancerous repertoires across multiple cancer types, validating the strength of deep neural MIL models for TCR-based diagnostic tasks.

[42] proposed DeepLION2, an advanced version of DeepLION that uses multiple instance contrastive learning and motif-based attention strategies to further improve the discriminative power of learned representations. This approach effectively captures subtle cancer-specific motifs within TCR repertoires, achieving even higher accuracy than its predecessor. The improved interpretability from motif-focused attention mechanisms additionally facilitated the identification of biologically relevant TCR sequence patterns associated with cancer.

Attention-based neural architectures have gained popularity in MIL, particularly due to their ability to identify influential instances within each bag. [8] developed Multiple Instance Neural Networks with Sparse Attention (MINN-SA), tailored for cancer detection from TCR repertoires. MINN-SA uses sparse attention to automatically select the most relevant TCR sequences within a repertoire, discarding irrelevant sequences, enhancing both interpretability and performance. The model demonstrated superior results across ten different cancer types, emphasising the potential of attention-based MIL methods for reliable and interpretable cancer diagnostics from TCR data.

Several additional studies have explored the predictive capabilities of MIL frameworks in different cancers using TCR data. [43] demonstrated that specific TCR repertoire features, particularly TCRβ V-J pairing frequencies, can be effectively used in a MIL context for predicting postoperative recurrence in non-small cell lung cancer (NSCLC), achieving an impressive AUC of around 0.9. Their results further underscore the predictive value of immune repertoire features within a MIL setting.

Similarly, [5] demonstrated the feasibility of predicting cancer-associated TCRs in ovarian and endometrial cancers using MIL. They used a deep learning model trained on TCR sequences derived from peripheral blood, highlighting the practical potential of MIL-based TCR analysis as a minimally invasive diagnostic tool, achieving accurate predictions.

Together, these studies confirm that MIL approaches are well-suited for TCR-based cancer classification. They offer both biological interpretability of individual TCRs within a repertoire and strong predictive performance. The success of recent MIL models, particularly those leveraging attention mechanisms, contrastive learning, and deep neural architectures, demonstrates the effectiveness of this paradigm in identifying cancer-associated patterns within high-dimensional, unlabeled sequence data. Building on this, the present project adapts a sparse attention-based MIL framework and evaluates its performance and interpretability.

3.2.2 TCR-BERT

TCR-BERT (T-cell Receptor Bidirectional Encoder Representations from Transformers) is a deep learning approach developed to generate biologically meaningful numerical representations of TCR sequences. Introduced in [34], TCR-BERT leverages the transformer-based BERT architecture, originally designed for natural language processing, to capture the sequential and contextual patterns within TCR sequences through self-supervised learning on large-scale, unlabeled datasets. TCR-BERT addresses the challenge of predicting TCR-antigen interactions given the diversity and complexity of TCR repertoires.

The innovation of TCR-BERT lies in its ability to exploit vast amounts of unlabeled TCR data through self-supervised masked amino acid prediction, enabling the model to learn the underlying "grammar" of valid TCR sequences. In this masked amino acid (MAA) pre-training stage, random residues in TCR sequences are hidden, and the model learns to predict these missing amino acids based only on the surrounding context. Subsequently, a supervised antigen classification pre-training step further refines the embeddings by predicting antigen specificity using a smaller set of labeled TCR sequences. This combination of self-supervised and supervised pre-training allows TCR-BERT to generate representations capable of capturing structural and functional attributes critical for antigen recognition [34].

Architecturally, TCR-BERT processes amino acid sequences through multiple layers of self-attention transformer blocks, effectively modeling complex residue interactions and sequence dependencies. This allows the model to encode TCR sequences into high-dimensional embeddings that reflect biologically relevant patterns, including antigen specificity, structural interactions, and evolutionary constraints. By utilising these embeddings, TCR-BERT supports a variety of downstream applications such as antigen binding prediction, sequence clustering, and even computational design of novel TCR sequences with predetermined antigen affinities [34].

Benchmarking studies have demonstrated that TCR-BERT embeddings significantly outperform traditional sequence encoding methods, as well as general-purpose protein language models, in predicting antigen-specific TCR interactions. Specifically, TCR-BERT achieves superior classification performance in challenging scenarios such as cross-patient generalisation. This highlights its robustness and potential clinical utility. Furthermore, TCR-BERT embeddings enable efficient exploratory analyses by facilitating clustering and visualisation of TCR repertoires, providing intuitive insights into shared antigen specificities and changes in repertoire composition over time [34].

Overall, TCR-BERT represents a substantial advancement in computational immunology, providing a flexible framework for encoding and analysing TCR sequences. Its ability to learn generalisable representations from unlabeled data, combined with state-of-the-art supervised performance, positions it as an important tool for future research and clinical applications.

3.2.3 SCEPTR

SCEPTR (Sequence Contrastive Encoding of Protein T-cell Receptors) is a novel deep-learning approach specifically designed for generating meaningful representations of T cell receptors (TCRs). Developed by a group of researchers from University College London and Princeton University, SCEPTR leverages contrastive learning to encode TCR sequences into embeddings that capture the biological and functional diversity from the immune repertoires [11]. Similarly to TCR-BERT, SCEPTR also addresses the challenge of representing high-dimensional TCR sequence data in a biologically meaningful numerical format.

The core innovation of SCEPTR lies in its application of contrastive learning, a paradigm in

machine learning where representations are learned by distinguishing between pairs of samples labeled as similar (positive pairs) and dissimilar (negative pairs). In the context of TCRs, positive pairs consist of TCR sequences derived from the same or similar antigenic specificities, while negative pairs involve sequences recognising distinct antigens. Through this contrastive framework, SCEPTR learns embeddings that cluster functionally similar TCR sequences closely together in the representation space, while placing dissimilar sequences further apart. This structured embedding space significantly enhances the model's ability to predict TCR functionality and antigen specificity [11].

Architecturally, SCEPTR employs a transformer-based encoder to process amino acid sequences of TCRs, capturing complex sequential dependencies and contextual relationships within them. The model was trained on a large-scale dataset of paired and unpaired TCR sequences, enabling it to generalise across different biological contexts and making it a useful tool in downstream tasks such as antigen specificity prediction, disease diagnosis, and cancer detection [11].

In benchmarking studies, SCEPTR demonstrated superior performance compared to conventional embedding techniques, including both symbolic encoding (e.g. Atchley factors) and other subsymbolic methods (e.g. TCR-BERT). Specifically, SCEPTR embeddings significantly improved accuracy in classifying antigen-specific TCR sequences and enhanced predictive capability in multi-instance learning scenarios for cancer-associated TCR identification [11].

Figure 3.2 visually summarises the core mechanism behind SCEPTR. Panel (a) depicts how SCEPTR converts TCR sequences, specifically their six complementarity-determining regions (CDR loops), into numeric embeddings. Each amino acid residue is initially represented as a 64-dimensional embedding vector (see panel b). These residue embeddings, alongside a special classification token (`<cls>`), pass through multiple layers of self-attention. SCEPTR uses the contextualised embedding of the `<cls>` token as the overall representation of the TCR, rather than employing average-pooled representations common in other models. Panel (b) provides details of the initial embedding stage, where each amino acid is encoded based on its identity (one-hot encoding), its respective CDR loop, and its relative positional index within that loop. Panel (c) illustrates the contrastive learning strategy, which explicitly encourages embeddings of similar (antigen-specific) TCRs to cluster together while pushing apart dissimilar ones. Panel (d) highlights how contrastive learning applies to supervised (matched TCR pairs with known shared specificity) and unsupervised settings (two independent augmented views of the same TCR sequence), allowing SCEPTR to generalise and capture biological relevance within its embedding space [11].

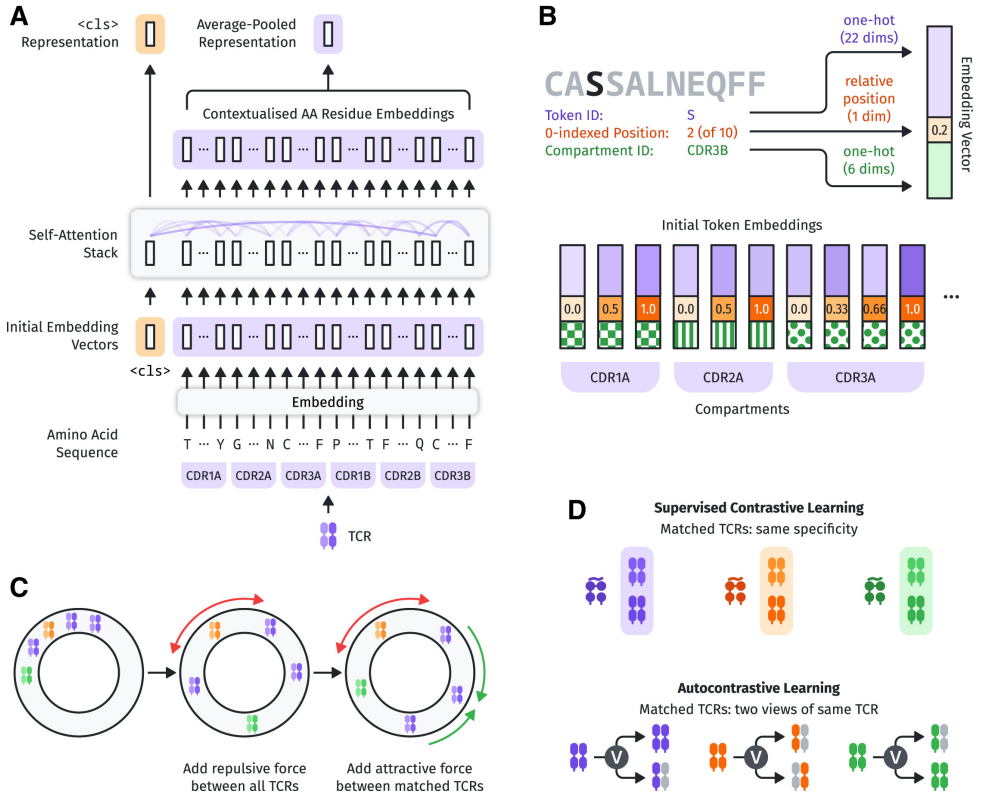


Figure 3.2: Overview of the SCEPTR model architecture and its contrastive learning framework. TCR sequences are processed through embedding and self-attention mechanisms (a, b). Contrastive learning strategies (c, d) guide the embeddings to cluster sequences with similar antigen specificities closer together (adapted from [11]).

Overall, SCEPTR's ability to generate rich and biologically relevant TCR embeddings positions it as another promising tool with applications in TCR-based cancer classification.

3.2.4 *Multi-Instance Transfer Learning on T cell receptor LLMs for Cancer Prediction* [1]

As introduced earlier in Section 2.2, Rudy C. Yuen's Master of Engineering dissertation at University College London, supervised by Professor John Shawe-Taylor, Professor Benny Chain, and Dr. Yuta Nagano, explored the use of large language models (LLMs) for TCR-based cancer classification. Yuen's work focused on contrasting the effectiveness of symbolic numerical encodings of TCRs with subsymbolic encodings obtained from pre-trained LLMs. The hypothesis of the study was that the latter provide superior numerical representations compared with the former, which are simply based on the physicochemical properties of amino acids within the TCRs [1].

Relying on the multiple instance learning framework, they designed and implemented a downstream classification model. Their architecture included a sparse attention-based pooling mechanism, allowing the model to focus on the TCRs most predictive of cancer presence. For training the model, they obtained data from the TRACERx lung cancer dataset [44], consisting of TCR repertoires from both symptomatic lung cancer patients and healthy individuals [1].

In their experiments, they compared four symbolic encodings, Random, Atchley Factors, Kidera Factors, and Amino Acid Properties, against two subsymbolic encodings, TCR-BERT and SCEPTR. Each encoding was tested under the same downstream MIL classifier, allowing for di-

rect performance comparison. The results demonstrated exceptional performance, particularly using SCEPTR, achieving perfect AUC scores of 100% on both evaluation and test datasets. Subsymbolic encodings substantially outperformed the traditional symbolic methods, supporting the initial hypothesis that pre-trained LLMs provide richer representations, capable of capturing the underlying biological patterns relevant to immunological cancer response [1].

They, however, acknowledged several limitations of the study. Despite the remarkable performance of subsymbolic embeddings, they raised concerns regarding the generalisability and potential overfitting, given the small data set size and complexity of high-dimensional embeddings, especially with the TCR-BERT [1].

Their dissertation laid the groundwork for the present project. The results obtained within this study will be compared against theirs to assess the generalisability and reproducibility.

Chapter 4

Requirements Analysis

This short chapter revisits the problem statement, and outlines the requirements that should be met to consider the project successful.

4.1 Problem Statement

This project addresses the problem of repertoire-level cancer classification based on TCR sequence data. Given a repertoire of TCRs from a patient, the task is to determine whether the patient has cancer or not. As discussed in section 3.2.1, the nature of this problem aligns with the multiple instance learning framework, where each patient is treated as "bag" of instances (TCRs), and the model learns to infer the overall label from the sequences within the bag.

The two primary goals of the projects are:

1. **Reproduction and verification results of prior work**: The goal is to establish a strong baseline for futher experiments. This involves training the models with cancer and control data using the same lightweight sparse attention MIL model with symbolic and subsymbolic encodings. The symbolic encodings used will be Random Encodings, Atchley Factors, Kidera Factors, and Amino Acid Properties. The subsymbolic encodings used will be TCR-BERT and SCEPTR The reproduced results will be compared against those reported in [1] to assess reproducibility. As stated in that study, it is expected to confirm that subsymbolic encodings outpeform the symbolic methods, reaffirming their effectiveness in capturing the underlying biological patterns within the TCR sequences.

2. **Design and implementation of exploratory evaluation strategies to enhance model interpretation**: After reproducing, comparing and evaluating the results against those from the prior work, the project will continue by designing and implementing a set of targeted experiments aimed at evaluating the internal behaviour of the model. This can aim at e.g. enhancing the explainability of model decisions or analysing the TCRs that are most relevant for cancer prediction.

4.2 Requirements

The requirements are divided into two categories: those required to reproduce prior work, and those necessary to extend it. Tables 4.1 and 4.2 summarise these requirements, which serve as targets throughout the project.

ID	Requirement
R1	Understand the architecture and design of MIL model from [1]
R2	Clone and understand the codebase from [45]
R3	Install and configure all dependencies and models (Python packages, TCR-BERT, SCEPTR)
R4	Obtain and preprocess the dataset to be used for model training
R5	Train the model with Random embeddings and store the results
R6	Train the model with Atchley Factor embeddings and record training metrics
R7	Train the model with Kidera Factor embeddings and record training metrics
R8	Train the model with Amino Acid Property embeddings and record training metrics
R9	Train the model with TCR-BERT embeddings and record training metrics
R10	Train the model with SCEPTR embeddings and record training metrics
R11	Validate that reproduced metrics (e.g. training/test accuracy, AUC) match the results from the prior work
R12	Evaluate all trained models on an evaluation set and record final performance metrics for comparison

Table 4.1: Reproduction Requirements

ID	Requirement
E1	Implement attention weight analysis to identify the most influential TCRs within each repertoire
E2	Investigate vector alignment of two trainable layers (scoring and classifying) within the model
E3	Generate 2D visualisations (e.g. UMAP) of bag-level embeddings for insights into the separability of cancer and control repertoires
E4	Discuss clinical relevance and limitations of interpretability analyses to guide future biological or diagnostic applications.

Table 4.2: Extension Requirements

Chapter 5

Methodology and Results

This chapter describes the experimental methodology and presents the results obtained from reproducing the work of Yuen [1]. It begins with the data preprocessing pipeline, which explains how raw TRACERx lung cancer and healthy control TCR data files are cleaned, split, and prepared for subsequent analysis.

Following this, the chapter provides a technical description of the symbolic and subsymbolic encodings used, emphasising their dimensional characteristics and relevance for model training. It then describes the attention-based multiple instance learning model architecture, highlighting why sparse attention, minimal parametrisation, and "frozen" TCR-BERT and SCEPTR embeddings are important to assessing the intrinsic power of each encoding type.

Next, the chapter outlines the implementation details, including the project's codebase, training environment, and hyperparameters. The chapter concludes with a detailed presentation of the experimental results across the training, testing, and evaluation sets, providing a basis for subsequent discussion.

5.1 Data Preprocessing

This section describes how the raw data is collected, cleaned, and split into training and evaluation sets before training the model.

5.1.1 Data Origin

The source of the dataset is the TRACERx (TRAcking non-small cell lung Cancer Evolution through therapy) study [44, 46, 47, 48, 49], a large-scale project investigating genomic and immunological correlates in non-small cell lung cancer (NSCLC). TRACERx focuses on understanding intratumor heterogeneity and evolution over time, providing sequencing information for both tumour and matched control samples.

Patients recruited in TRACERx undergo surgical resection of the primary tumour (stages I–IIIa). Alongside tumour tissue, peripheral blood is drawn to obtain peripheral blood mononuclear cells (PBMCs), allowing for a paired analysis of each individual's immune repertoire. For the purpose of this work, TCR sequences were extracted from PBMC samples of lung cancer patients (referred to in this study as `pbmc_cancer`) and from healthy donors (age-matched controls, referred to as `control`). This gives the two main categories of data:

- `control`: blood-derived TCRs from healthy individuals (age-matched).

- `pbmc_cancer`: blood-derived TCRs from NSCLC patients (the TRACERx cohort)

Each patient in either group typically contributes two data files: one containing alpha-chain TCRs and one containing beta-chain TCRs. This means that each patient's TCR repertoire is split into separate chains (i.e. there is no alpha-beta chain pairing). In total for the `control` set, there are 111 alpha-chain files and 111 beta-chain files, and for the `pbmc_cancer` set, there are 55 alpha-chain files and 55 beta-chain files. Some patients in the TRACERx study have multiple time points or follow-up samples, but this project uses only the baseline (pretreatment) data, excluding the future time points.

5.1.2 File Formats and Columns

All raw TCR sequencing data are provided as tab-separated value (TSV) files. For each TCR read, several fields capture V(D)J alignment information, sequence identifiers, and extra metadata. Key columns in `pbmc_cancer` TSV files include:

- `sequence_id`
- `v_call`, `d_call`, `j_call`
- `junction_aa`
- `duplicate_count`
- `sequence`, `junction`
- `decombinator_id`, `rev_comp`, `productive`
- `sequence_aa`, `cdr1_aa`, `cdr2_aa`
- `vj_in_frame`, `stop_codon`, `conserved_c`, `conserved_f`
- `sequence_alignment`, `germline_alignment`
- `v_cigar`, `d_cigar`, `j_cigar`
- `av_UMI_cluster_size`
- `LTX_id`

Control data files have the same general structure but omit certain cancer-specific fields such as `LTX_id`. For modeling, the most important fields include the raw amino acid sequence of the CDR3 region (from `junction_aa` or `sequence_aa`), along with the V/J calls.

5.1.3 Filtering and Trimming

Although many columns are available, not all are required for the encoding steps. Four of the six TCR-encoding methods tested in this study require only the CDR3 amino acid sequence. Hence, the files are trimmed to retain the minimal required columns. For symbolic methods (i.e. Random, Atchley, Kidera, AA Properties) the pipeline only uses the CDR3 sequence (`junction_aa`). TCR-BERT also requires only the TCR CDR3 sequence, though optionally it can also incorporate CDR1 and CDR2. For SCEPTR, the pipeline requires the V gene, J gene, and the CDR3 region, specifically columns `TRAV`/`TRAJ`/`CDR3A` or `TRBV`/`TRBJ`/`CDR3B` must be produced, depending on whether the file is alpha or beta.

The raw data may potentially contain TCRs with nonstandard amino acids, partial sequences, or ambiguous characters (e.g. `X`). If such exist, these are removed or replaced according to standard mapping rules before encoding. If a TCR sequence is fully unrecognised (i.e. it contains only invalid characters), that row is discarded.

For SCEPTR, gene names must follow standard IMGT conventions. To ensure this, the tidytcells package [50] is used to clean and standardise V/J gene symbols and CDR3 sequences.

5.1.4 Train–Evaluation Splits

After filtering out incomplete or invalid TCR entries, the repertoire for each file is retained as a single bag. To ensure unbiased performance estimates, the entire dataset is randomly split into 80% training and 20% evaluation sets at the file level. In total:

- 89 `control` files and 44 `pbmc_cancer` files go to the training split,
- 22 `control` files and 11 `pbmc_cancer` files go to the evaluation split.

This partition ensures that no patient's alpha or beta chain file is duplicated between training and evaluation. The final distribution helps maintain comparable representation across lung cancer and control repertoires in both sets.

5.1.5 Alpha/Beta Chain Handling

In Yuen's work [1], TCR alpha and beta chain sequences were concatenated into a single input file per individual. While this approach simplifies data handling, it obscures meaningful chain-specific differences and complicates encoding, especially when using pretrained models or embeddings that expect single-chain sequences as input. This study maintains separate handling of α- and β-chain data throughout training and evaluation. The decision aligns with practical considerations, as robust chain-pairing information is generally unavailable in large-scale bulk TCR-seq datasets [36, 35]. Prior literature suggests that TCR β chains typically exhibit greater diversity and are more informative for repertoire-level analysis [47, 48], justifying independent analysis of each chain type.

Table 5.1 presents the number of unique TCR sequences for both alpha and beta chains, separated by cancer and control groups, before and after applying the 80-20 train-eval split.

Dataset Split	**Alpha TCRs**	**Beta TCRs**
Cancer (Before Split)	612,770	1,194,309
Control (Before Split)	2,625,693	4,391,894
Cancer (TrainTest 80%)	509,023	992,826
Control (TrainTest 80%)	2,115,248	3,558,204
Cancer (Eval 20%)	103,747	201,483
Control (Eval 20%)	510,445	833,690

Table 5.1: Unique TCR Counts for Alpha and Beta Chains Before and After 80-20 Split

5.1.6 Data Preprocessing Flow Summary

In summary, the data preprocessing flow for this project consists of: (1) retrieving the raw TRACERx PBMC and healthy control TCR files, (2) removing unknown columns or incomplete entries, (3) splitting the dataset at the file level into training and evaluation sets, (4) retaining only the

chain-specific CDR3 sequences needed by each encoding method. This pipeline, applied across alpha and beta chain data, produces a well-defined input for subsequent symbolic and subsymbolic TCR-encoding methods.

5.2 Model Analysis

5.2.1 Symbolic Encodings

Introduced in subsection 3.1.3, symbolic encodings provide a way of converting TCR amino acid sequences into fixed-dimensional vectors, by using their physicochemical properties. In this study, four such encodings are used:

- **Random**: Assigns each amino acid a 5-dimensional vector drawn uniformly from $[0,\ 1]$. It is designed to serve as a baseline and expected to perform rather poorly compared with the other symbolic encoding methods.
- **Atchley** [31]: Transforms each amino-acid into a 5-dimensional vector capturing the following 5 biochemical properties: polarity, secondary structure, molecular volume, codon diversity, and electrostatic charge.
- **Kidera** [32]: Transforms each amino-acid into a 10-dimensional feature capturing 10 biochemical properties of amino-acids.
- **AA Properties** [33]: Transforms each amino-acid into a 14-dimensional feature capturing 14 biochemical properties of amino-acids.

Table 5.2 summarises all the symbolic encodings used alongside the two subsymbolic methods (TCR-BERT and SCEPTR) for completeness. It highlights each encoding's dimensionality and whether it is hand-crafted or learned.

Encoding	Dimensions	Type	Description
Random	5	Symbolic	Uniformly random vectors
Atchley	5	Symbolic	5 physicochemical properties [31]
Kidera	10	Symbolic	10 physicochemical properties [32]
AA Properties	14	Symbolic	14 physicochemical properties [33]
TCR-BERT	768	Subsymbolic	Transformer trained on TCR data [34]
SCEPTR	64	Subsymbolic	Contrastive learning for TCRs [11]

Table 5.2: Summary of both symbolic and subsymbolic encodings used in this study

Each symbolic method uses a transformation per each amino acid within a TCR sequence. For example, if an amino acid `G` is encoded via Kidera factors, the results will be a 10-dimensional real vector:

$$\mathbf{v}_{\mathtt{G}} = (1.46,\ -1.96,\ -0.23,\ -0.16,\ 0.1,\ -0.11,\ 1.32,\ 2.36,\ -1.66,\ 0.46) \in \mathbb{R}^{10}.$$

Since a length of a full TCR can vary, a mean-pooling approach is applied to transoform the entire sequence:

$$\mathtt{TCRseq} = (\mathtt{AA}_1, \mathtt{AA}_2, \ldots, \mathtt{AA}_L),$$

each amino acid $\texttt{AA}_i$ is mapped to a vector $\mathbf{v}_i \in \mathbb{R}^d$, then the full TCR embedding is given by:

$$\textbf{TCR\_enc} \;=\; \frac{1}{L}\sum_{i=1}^{L}\mathbf{v}_i.$$

For example, with Atchley factors ($d = 5$) and a CDR3 sequence like `CASSPFQGHDYEQYF`, the method computes one 5-dimensional vector per each amino acid and averages them to obtain a single 5-dimensional TCR embedding.

5.2.2 Subsymbolic Encodings

Introduced in Subsection 3.1.3, subsymbolic encodings rely on large-scale data and neural networks to learn the numerical representation of a TCR, rather than following a predefined biochemical table. This study examines two subsymbolic models: TCR-BERT and SCEPTR.

TCR-BERT [34] (see Subsection 3.2.2) is a transformer-based model trained on unlabeled TCR data via masked amino acid prediction. By treating each amino acid residue like a token in a "language", TCR-BERT learns context-aware embeddings, encoding subtle information about amino-acid spatial locality. For each TCR sequence, TCR-BERT produces a series of hidden states, typically of dimension 768, which are reduced to a single 768-dimensional vector by mean pooling.

SCEPTR [11] (see Subsection 3.2.3) utilises a contrastive-learning approach to push "functionally similar" TCR sequences closer in the embedding space, while separating dissimilar pairs. Each amino acid is tokenised, embedded, and passed through multiple self-attention layers. The final vector is either taken from a special token (`<cls>`) or from an average of all hidden states. SCEPTR's learned embedding dimension is smaller (64-dimensional) than TCR-BERT's.

Both TCR-BERT and SCEPTR yield high-dimensional vectors that encode each TCR's sequence context. Unlike symbolic approaches, these embeddings are trained representations: the model's internal parameters adapt to maximise performance on masked amino acid prediction or contrastive objectives, rather than explicitly encoding fixed physicochemical properties. As a result, they can represent intricate patterns within TCRs, but are less interpretable.

5.2.3 Architecture Overview

Figure 5.1 provides a high-level illustration of the multi-instance learning model proposed in [1]. The architecture is deliberately kept fairly simple, to highlight the potential and superiority of subsymbolic encodings, being able to achieve high cancer-predicting accuracy even with a simple model. The architecture aggregates per-TCR embeddings into a single bag-level vector via a sparse attention mechanism (sparsemax), followed by a classification layer giving the final probability of a TCR bag belonging to a cancer or control patient.

A TCR repertoire (bag) can contain thousands of TCR sequences, yet only a small part of those may be cancer-specific. This is a challenge for memory usage and training stability: a more complex or deeper network would require storing large intermediate tensors for every TCR in a bag. To overcome this, a lightweight network with only two trainable linear layers is employed, similar to MINN-SA [8] and other sparse-attention MIL methods. [1] emphasises that restricting the model capacity helps demonstrate whether learned embeddings (TCR-BERT and SCEPTR) indeed capture richer information than handcrafted encodings.

Following five steps describe the model architecture in more detail:

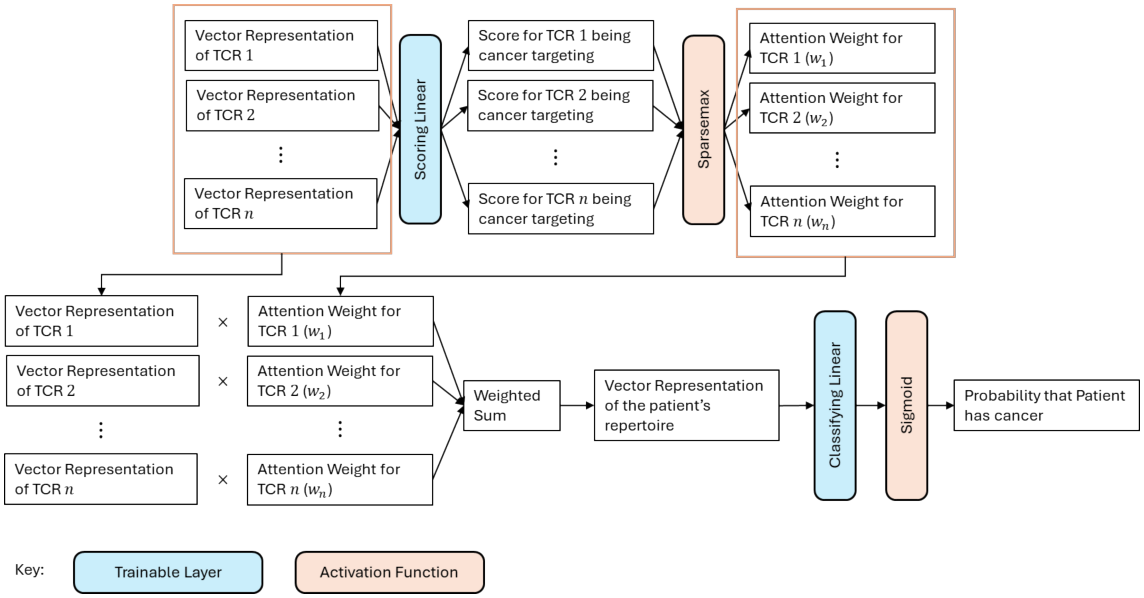


Figure 5.1: High-level overview of the multiple instance learning architecture adapted from [1], highlighting the sparse attention mechanism and final binary classifier. Each TCR is encoded into a vector (symbolically or subsymbolically), then scored and aggregated into a single bag-level representation.

Step 1: Per-TCR Embeddings

Each TCR i is mapped to a vector $\mathbf{e}_i \in \mathbb{R}^d$. The dimension d depends on the encoding strategy:

- **Symbolic encodings** produce $d \in \{5, 10, 14\}$ depending on the encoding (see Subsection 5.2.1)
- **Subsymbolic encodings** procude $d = 768$ (TCR-BERT) or 64 (SCEPTR) (see Subsection 5.2.2)

These embeddings are "frozen" when training the model, meaning no gradient updates are applied to the symbolic tables or the weights of TCR-BERT/SCEPTR.

Step 2: Scoring Layer

After obtaining $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n$ for a single bag (full TCR repertoire), a scoring linear layer assigns each TCR embedding a scalar score:

$$s_i \;=\; \langle \mathbf{w}_s,\, \mathbf{e}_i \rangle \;+\; b_s, \quad i = 1, \ldots, n.$$

The trainable parameters here are $\mathbf{w}_s \in \mathbb{R}^d$ and a scalar bias b_s. Intuitively, $\mathbf{w}_s$ points in the "direction" of TCR embeddings that the network finds most relevant for identifying cancer.

Step 3: Sparsemax Attention

An important innovation drawn from MINN-SA [8] and further explained by [1] is the use of a sparsemax [51] activation function instead of softmax. Sparsemax processes the scores $\{s_i\}$ to produce a probability-like weight $p_i \geq 0$ with $\sum_i p_i = 1$, but with the ability to assign zero weights to uninformative TCRs. Denoting the sorted score vector as $s_{(1)} \geq s_{(2)} \geq \cdots \geq s_{(n)}$, Sparsemax essentially solves :

$$\max_{p\in\mathbb{R}^n} \quad \sum_{i=1}^{n}(p_i\, s_i) \quad \text{subject to} \quad p_i \geq 0,\ \sum_i p_i = 1,\ \text{ and many } p_i \text{ can be zero.}$$

This yields weights $p_1, \ldots, p_n$ that are often sparse, amplifying only a portion of high-scoring TCR embeddings. The result is a more interpretable attention distribution and a stronger focus on TCRs that correlate with cancer.

The softmax algorithm could be applied to the same problem, however it's difference is that it would assign every TCR sequence a non-zero weight [52], meaning it would be incapable of "filtering out" non-informative TCRs.

Step 4: Bag-Level Aggregation

Given weights p_i and embeddings $\mathbf{e}_i$, the bag-level embedding is

$$\mathbf{b} \;=\; \sum_{i=1}^{n}\big(p_i\, \mathbf{e}_i\big),$$

which has the same dimension d as each TCR embedding. Because $\sum_i p_i = 1$, $\mathbf{b}$ is essentially the weighted average of the TCR vectors found most relevant by the scoring layer.

Step 5: Final Classification

A second linear layer $\mathbf{w}_c \in \mathbb{R}^d$ plus bias b_c performs a logistic classification:

$$\hat{y} \;=\; \sigma\big(\langle \mathbf{w}_c,\ \mathbf{b}\rangle \;+\; b_c\big),$$

where $\sigma(\cdot)$ is the sigmoid function. The scalar $\hat{y} \in (0, 1)$ represents the network's predicted probability that the patient's repertoire is cancerous.

Only two linear layers (scoring and classification) are trainable. Hence, the total parameter count is $2(d + 1)$, because each layer has d weights plus a single bias. For example, if $d = 64$ (SCEPTR), there are $2 \times (64 + 1) = 130$ trainable parameters. This fairly small size results in a reasonable memory usage while training the model.

The approach is particularly important for large n, where n is the number of TCRs in a repertoire. Each TCR must pass through the network's forward pass to compute s_i, but only $\mathbf{b}$ (of dimension d) is stored for the final classification. This helps overcome the possibility of memory explosion, which would occur if deeper layers maintained full intermediate activations for every TCR simultaneously.

In summary, the key design choices of the model architecture include: **simplicity**: only two linear layers ensure minimal overhead, and help fairly assess the performance of different TCR encodings rather than model capacity; **sparse attention (sparsemax)**: weights are zeroed out for irrelevant TCR embeddings, making the resulting bag-level vector easier to interpret; **frozen encodings**: symbolic or learned embeddings (e.g. TCR-BERT) remain unchanged during training, isolating the effect of each encoding method.

5.3 Codebase and Training Setup

This section describes the practical setup and software environment required to reproduce the multiple instance learning pipeline introduced in earlier sections. It covers the motivation for forking Yuen's repository [1, 45], the structure of that forked repository, the process of installing dependencies and pretrained models, the GPU environment the models are being trained on, the logging format for runs, and the final hyperparameter configurations including the number of repeated training runs per encoding.

5.3.1 Forking GitHub Repository

In order to reproduce and extend the multi-instance TCR classification framework from [1], the codebase was initially forked from Yuen's public GitHub repository at `https://github.com/RcwYuen/TCR-Cancer-Prediction` [45].

Forking, as opposed to rewriting from scratch, offers several advantages. First, it directly supports comparability with [1] by preserving the same downstream logic and enabling a fair environment for verifying previous results. Second, it ensures the MIL architecture is already configured for the symbolic and subsymbolic TCR encodings. Third, the repository contains useful scripts for e.g. logging, checkpointing, and Jupyter notebooks for post-training analysis, which can be adapted and extended for the newly proposed experiments further in this study.

5.3.2 Repository Structure

After forking, the project remains organised in a directory layout similar to the original. The `data/` folder stores the `pbmc_cancer` and `control` files preprocessed for training the models. Since these contain sensitive patient information, these are not tracked by version control. In Yuen's approach, cancer and control data were retrieved automatically from the Chain Lab RDS database at UCL [45] using Python scripts located in `loaders/`. This study, however, relied on data provided manually by Professor Benny Chain, requiring a rewrite of the preprocessing pipeline. The newly written scripts are placed under `data-preprocessing/`.

The directory `scripts/` contains two `.txt` files with dependencies that need to be installed (separate files for Linux and Windows dependencies). SCEPTR model is also simply installed as a Python package, while the TCR-BERT needs to be downloaded with the `load_ptm.py` script. The `src/model.py` file defines the actual PyTorch modules for the multiple instance classifier, including the sparse attention mechanism and linear layers. The main training flow is implemented in `trainer-symbolic.py`, `trainer-tcrbert.py`, and `trainer-sceptr.py` (for all 4 symbolic encodings, TCR-BERT, and SCEPTR respectively). Finally, there is a number of Jupyter notebooks (`.ipynb`) used for plotting training and evaluation statistics.

5.3.3 Training Environment

All experiments were conducted using Python 3.9.21 and PyTorch compiled for CUDA 12.6. The steps to install the library stack mostly follow `requirements.txt` (or `requirements-linux.txt`). The training was performed on the UCL Department of Computer Science Lab 105 PCs, each equipped with an Nvidia RTX 3090 GPU, with 24GB of VRAM and 10,496 cores [53]. Job progress was monitored via `nvidia-smi` to ensure GPU memory is not saturated when running high-dimensional models such as TCR-BERT.

Each training script (`trainer-symbolic.py`, `trainer-tcrbert.py`, `trainer-sceptr.py`) writes progress to a `.log` file. Each epoch's output includes metrics such as training and testing losses, batch-level accuracies, AUC values. After training, model performance is evaluated on the held-out 20% set with the `src/calculate_evals.py` script.

5.3.4 Repeats and Hyperparameters

All encodings were trained multiple times to capture variability from random initial trainable parameters. Table 5.3 summarises the number of repeats per method: five repeats each for Random, Atchley, Kidera, and AA Properties, two repeats for TCR-BERT, due to its long training times, and ten repeats for SCEPTR, as that is aimed to be evaluated in more detail.

Hyperparameters match those described by Yuen [1]. Each model trains for 50 epochs using the Adam optimiser with a learning rate of 0.001. To prevent bias from class imbalances, the positive (`cancer`) and negative (`control`) classes are weighted by the inverse of their proportions in the training set. L2 regularisation (`weight_decay=0.25`) is applied only to TCR-BERT's linear layers, as the 768-dimensional embeddings can overfit when trained on only around 190 patient bags. No other regularisation is applied to smaller encodings.

Repeats	Encoding Method
5	Random, Atchley, Kidera, AA Properties,
2	TCR-BERT
10	SCEPTR

Table 5.3: Number of repeats taken for each encoding method

5.4 Results

This section presents quantitative outcomes obtained from the training using all six encoding strategies (symbolic and subsymbolic). The metrics reported are binary cross-entropy (BCE) loss, accuracy, and AUC (Area Under the ROC Curve).

5.4.1 Metrics Reported

To define the metrics measured, let $\hat{y}_i \in (0, 1)$ be the predicted probability of the positive class for instance i, and let $y_i \in \{0, 1\}$ be the true label.

Binary Cross-Entropy (BCE) Loss

BCE loss, given by

$$\text{BCE Loss} \;=\; -\frac{1}{N}\sum_{i=1}^{N}\Big[y_i \log(\hat{y}_i) \;+\; (1-y_i)\log\big(1-\hat{y}_i\big)\Big].$$

is the standard loss function for binary classification tasks. It compares the predicted probability ($\hat{y}_i \in (0, 1)$) against the true binary label ($y_i \in \{0, 1\}$), penalising incorrect or overconfident predictions [54]. A lower BCE loss indicates that the model's predicted probabilities are closer to true labels. The loss should generally decrease as training progresses, indicating that the model is "learning" and making more accurate predictions with increasing number of epochs.

Accuracy

Accuracy measures the proportion of correct predictions when the predicted probability $\hat{y}_i$ is threshold at 0.5:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP, TN, FP, FN stand for True Positives, True Negatives, False Positives, and False Negatives, respectively. Accuracy therefore serves as a reasonable measure of model performance. However, it does not differentiate between types of errors (e.g. false positives vs false negatives) and should be evaluated alongside other metrics [55].

Area Under the ROC Curve (AUC)

AUC measures the model's ability to distinguish between the two classes across all classification thresholds. It is calculated from the Receiver Operating Characteristic (ROC) curve, which plots the True Positive Rate (TPR) against the False Positive Rate (FPR):

$$\text{TPR} = \frac{TP}{TP + FN}, \quad \text{FPR} = \frac{FP}{FP + TN}$$

AUC represents the probability that a randomly chosen positive instance is ranked higher than a randomly chosen negative one:

$$\text{AUC} = \int_0^1 \text{TPR}(x)\, dx$$

For example, an AUC of 1.0 implies perfect separability and 0.5 implies random guessing. AUC is particularly informative in classification settings, making it a preferred metric to evaluate the results of this study [56].

5.4.2 Train and Test Set Results

During training, the model partitions the training dataset into a training and test split using an 80/20 ratio. This split is fixed for each run and is handled automatically by the training script. The training set is used to optimise the model parameters, while the test set is used to evaluate generalisation performance at each epoch. For every training run, the best-performing checkpoint is identified based on the AUC.

Tables 5.4 and 5.5 present the model's performance metrics, BCE loss, accuracy, and AUC, for both the training and test sets. Each row represents the range of (minimum, maximum) and mean (μ) across all runs for a specific encoding method and chain type (α or β). The "best" epoch of each run is the one that gave the highest AUC on the test set during training. Only metrics from these chosen epochs are presented. Table 5.4 shows training set performance, while Table 5.5 shows test set performance for the same epochs.

Encoding	Train Loss	Train Accuracy	Train AUC
Random (α)	0.656 - 0.726 (μ: 0.691)	0.336 - 0.701 (μ: 0.536)	0.563 - 0.698 (μ: 0.614)
Random (β)	0.678 - 0.695 (μ: 0.689)	0.321 - 0.689 (μ: 0.543)	0.187 - 0.645 (μ: 0.413)
Random [1]	0.687 - 0.692 (μ: 0.689)	0.612 - 0.836 (μ: 0.716)	0.778 - 0.889 (μ: 0.843)
Atchley (α)	0.685 - 0.697 (μ: 0.692)	0.318 - 0.673 (μ: 0.465)	0.334 - 0.709 (μ: 0.568)
Atchley (β)	0.689 - 0.695 (μ: 0.692)	0.349 - 0.717 (μ: 0.617)	0.295 - 0.696 (μ: 0.559)
Atchley [1]	0.686 - 0.696 (μ: 0.691)	0.395 - 0.750 (μ: 0.583)	0.223 - 0.877 (μ: 0.663)
Kidera (α)	0.691 - 0.692 (μ: 0.692)	0.505 - 0.664 (μ: 0.615)	0.607 - 0.766 (μ: 0.674)
Kidera (β)	0.690 - 0.693 (μ: 0.692)	0.368 - 0.717 (μ: 0.608)	0.635 - 0.736 (μ: 0.680)
Kidera [1]	0.670 - 0.690 (μ: 0.682)	0.664 - 0.836 (μ: 0.732)	0.876 - 0.951 (μ: 0.909)
AA Properties (α)	0.687 - 0.693 (μ: 0.691)	0.514 - 0.692 (μ: 0.606)	0.573 - 0.737 (μ: 0.651)
AA Properties (β)	0.690 - 0.691 (μ: 0.691)	0.594 - 0.745 (μ: 0.660)	0.578 - 0.729 (μ: 0.630)
AA Properties [1]	0.570 - 0.691 (μ: 0.664)	0.566 - 0.849 (μ: 0.716)	0.648 - 0.912 (μ: 0.784)
TCR-BERT (α)	0.040 - 0.241 (μ: 0.140)	0.935 - 1.000 (μ: 0.967)	0.972 - 1.000 (μ: 0.986)
TCR-BERT (β)	0.110 - 0.121 (μ: 0.116)	0.972 - 0.972 (μ: 0.972)	0.995 - 0.998 (μ: 0.997)
TCR-BERT [1]	0.105 - 0.353 (μ: 0.221)	0.836 - 0.980 (μ: 0.914)	0.925 - 0.995 (μ: 0.967)
SCEPTR (α)	0.256 - 0.658 (μ: 0.577)	0.692 - 0.981 (μ: 0.813)	0.804 - 1.000 (μ: 0.876)
SCEPTR (β)	0.579 - 0.667 (μ: 0.629)	0.689 - 0.858 (μ: 0.816)	0.773 - 0.970 (μ: 0.876)
SCEPTR [1]	0.273 - 0.601 (μ: 0.396)	0.824 - 0.949 (μ: 0.886)	0.883 - 0.978 (μ: 0.948)

Table 5.4: Results for the best-performing checkpoint on the train set (α-only vs β-only vs [1])

Encoding	Test Loss	Test Accuracy	Test AUC
Random (α)	0.642 - 0.724 (μ: 0.688)	0.296 - 0.704 (μ: 0.533)	0.664 - 0.790 (μ: 0.724)
Random (β)	0.685 - 0.696 (μ: 0.691)	0.259 - 0.667 (μ: 0.541)	0.290 - 0.877 (μ: 0.498)
Random [1]	0.686 - 0.691 (μ: 0.688)	0.605 - 0.842 (μ: 0.753)	0.889 - 0.960 (μ: 0.927)
Atchley (α)	0.686 - 0.697 (μ: 0.692)	0.185 - 0.741 (μ: 0.459)	0.517 - 0.794 (μ: 0.684)
Atchley (β)	0.690 - 0.694 (μ: 0.692)	0.222 - 0.630 (μ: 0.548)	0.525 - 0.873 (μ: 0.635)
Atchley [1]	0.684 - 0.694 (μ: 0.690)	0.395 - 0.763 (μ: 0.674)	0.157 - 0.939 (μ: 0.728)
Kidera (α)	0.691 - 0.694 (μ: 0.692)	0.333 - 0.778 (μ: 0.615)	0.632 - 0.835 (μ: 0.751)
Kidera (β)	0.690 - 0.693 (μ: 0.692)	0.444 - 0.704 (μ: 0.637)	0.645 - 0.859 (μ: 0.747)
Kidera [1]	0.663 - 0.690 (μ: 0.681)	0.632 - 0.921 (μ: 0.763)	0.901 - 0.975 (μ: 0.941)
AA Properties (α)	0.686 - 0.694 (μ: 0.691)	0.370 - 0.815 (μ: 0.644)	0.684 - 0.842 (μ: 0.766)
AA Properties (β)	0.689 - 0.692 (μ: 0.691)	0.556 - 0.741 (μ: 0.652)	0.563 - 0.802 (μ: 0.659)
AA Properties [1]	0.595 - 0.692 (μ: 0.669)	0.526 - 0.842 (μ: 0.711)	0.768 - 0.892 (μ: 0.855)
TCR-BERT (α)	0.090 - 0.388 (μ: 0.239)	0.778 - 0.963 (μ: 0.870)	0.940 - 1.000 (μ: 0.970)
TCR-BERT (β)	0.284 - 0.704 (μ: 0.494)	0.741 - 0.852 (μ: 0.796)	0.690 - 0.976 (μ: 0.833)
TCR-BERT [1]	0.366 - 0.579 (μ: 0.497)	0.737 - 0.842 (μ: 0.795)	0.820 - 0.946 (μ: 0.900)
SCEPTR (α)	0.258 - 0.669 (μ: 0.595)	0.630 - 1.000 (μ: 0.793)	0.704 - 1.000 (μ: 0.830)
SCEPTR (β)	0.641 - 0.685 (μ: 0.662)	0.556 - 0.778 (μ: 0.681)	0.520 - 0.893 (μ: 0.689)
SCEPTR [1]	0.254 - 0.606 (μ: 0.406)	0.771 - 0.971 (μ: 0.877)	0.902 - 1.000 (μ: 0.950)

Table 5.5: Results for the best-performing checkpoint on the test set (α-only vs β-only vs [1])

Detailed visualisations of training and testing metrics measured are provided in Appendix B.1 Together, these results form a basis for discussion in Chapter 6.

5.4.3 Model Evaluation Results

While the preceding section detailed the training and testing set performance across all encoding methods, this section focues on the final evaluation set (the 20% of data retained for post-hoc assessment). Again, BCE loss, accuracy, and AUC are reported. Table 5.6 presents the outcomes for SCEPTR, both for the α-chain and β-chain models, and compares them against the results reported in [1].

Encoding	**BCE Loss**	**Accuracy**	**AUC**
SCEPTR (α)	0.240 - 0.676 (μ: 0.590)	0.667 - 1.000 (μ: 0.773)	0.657 - 1.000 (μ: 0.814)
SCEPTR (β)	0.624 - 0.676 (μ: 0.656)	0.576 - 0.788 (μ: 0.685)	0.570 - 0.860 (μ: 0.698)
SCEPTR [1]	0.215 - 0.545 (μ: 0.326)	0.842 - 1.000 (μ: 0.947)	0.922 - 1.000 (μ: 0.988)

Table 5.6: Result for SCEPTR on the evaluation set (α-only vs β-only vs [1])

Again, the "best" checkpoints for each run were selected based on the best AUC on the test set. These were then used to generate predictions on the evaluation set. Hence, each row in Table 5.6 aggregates the performance metrics for these best checkpoints across ten repeated runs for SCEPTR. All three metrics reported are calculated from the final CSV outputs (generated by `calculate_evals.py`), and a separate Jupyter notebook uses `scikit-learn` functions to compute BCE loss, accuracy, and AUC.

The evaluation results are presented only for SCEPTR for two reasons. Firstly, TCR-BERT is expected to overfit, because of the limited data it's being trained on [1] (this is also discussed and confirmed in Chapter 6), SCEPTR yields more stable results. Second, SCEPTR is a model developed at UCL, by a team of researchers that including Professor John Shawe-Taylor, and Professor Benny Chain, who are supervising this work, which aims to extend Yuen's findings [1] by adding interpretability analysis in Chapter 6.

Chapter 6

Evaluation

This chapter offers a comprehensive evaluation and interpretation of the model results introduced in the previous section. It begins by comparing the symbolic and subsymbolic encodings of TCR sequences, highlighting the strengths and limitations of each approach. The results are then contrasted against those reported by Yuen [1] to assess the reproducibility and impact of methodological differences. Next, the chapter discusses the discrepancies in performance between the α and β chain models, providing an interpretation with regards to their diagnostic relevance.

The second part of this chapter focuses on model interpretability. It analyses which TCR sequences are most influential in classification decisions by inspecting the sparse attention weights assigned to individual TCRs by the MIL architecture. This is followed by an assessment of model's representation space, by analysing the alignment between the scoring and classifying layers within and across training runs. Finally, the chapter provides visual exploration of the learned bag-level embeddings, using UMAP to project patient repertoires into two dimensions. This helps evaluate the model's ability to separate cancer from control groups in the latent space.

6.1 Discussion

6.1.1 Comparison of Symbolic and Subsymbolic Encodings

Among the symbolic encodings, Random performed the worst overall, aligning with the expectation. This encoding assigns random numerical values to amino acids, avoiding any biological meaning and therefore lacking any informative structure. While the model trained on random encodings for α chains achieved a seemingly reasonable mean AUC of 0.724 on the test set, Figure B.3 reveals that the training loss actually increases over time, indicating that the model fails to converge. Notably, in Yuen's work [1], the reported performance of Random was unusually strong and comparable to other symbolic encodings. However, after inspecting the training logs in their GitHub repository [45], it was discovered that those results were produced using Atchley encodings, most likely by mistake. Thus, the poor performance observed here for true random encodings is both expected and consistent with theoretical reasoning.

The other three symbolic encodings (Atchley, Kidera, and AA Properties) achieved better performance than the random baseline, with mean test AUCs of 0.684, 0.751, and 0.766 respectively for α chains, and 0.635, 0.747, and 0.659 for β chains. These results indicate that the model is able to capture some degree of separability between cancer and control patients using biologically meaningful, though static, representations. Figures B.11, **??**, and **??** show increasing and plateau-

ing trends in training and testing accuracy, alongside increasing training and testing losses, which is a typical pattern indicative of model learning. However, both training and testing AUCs remain lower than those reported by Yuen [1].

It is also worth noting that among the Atchley, Kidera, and AA Properties symbolic encodings, which represent each amino acid within a TCR sequence using 5-dimensional, 10-dimensional, and 14-dimensional vectors respectively, the latter two consistently produced slightly higher AUCs than Atchley. This trend suggests that higher-dimensional representations, which encapsulate a broader range of physico-chemical properties, may improve model performance by providing a richer and more informative feature space.

Despite the acceptable classification performance observed for symbolic encodings, none of them reached the levels of accuracy or AUC achieved by the subsymbolic approaches of TCR-BERT and SCEPTR. These results strongly suggest that learned embeddings from pretrained language models offer a significant advantage in capturing biologically meaningful structure within TCR sequences.

TCR-BERT yielded particularly high training performance, with mean train AUCs of 0.972 for α chains and 0.997 for β chains, and test AUCs of 0.940 and 0.833 respectively. However, closer inspection reveals clear signs of overfitting. As shown in Figure B.35, the testing loss remains relatively flat throughout training, while the training accuracy climbs toward 1.000 and the test accuracy remains between 0.700 and 0.800. The difference between those training and testing metrics is an indication of overfitting.

As noted by Yuen, this is likely due to the disproportion between model complexity and dataset size: *"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."* [1]. This study, using even fewer training files (133), further amplifies this imbalance and reconfirms the observed overfitting tendency.

SCEPTR demonstrated more stable and balanced behaviour. Model trained on both α and β chains achieved a mean train AUC of 0.876, with test AUCs of 0.830 and 0.689 respectively. Notably, SCEPTR achieved a maximum test AUC of 1.000 for α chains and 0.893 for β chains, highlighting its potential when the model generalises well. The training loss curves in Figure B.43 show a steady decline, although not as steep as might be desired, and training accuracy plateaus around 0.85. Testing accuracy remains slightly lower, and while there is an initial spike in accuracy, the trend gradually stabilises. On the evaluation set, mean AUCs of 0.814 for α chains and 0.698 for β chains were observed. These results are slightly lower than those reported in Yuen's original work [1].

In summary, SCEPTR proves to be a well-balanced approach, less prone to overfitting than TCR-BERT and capable of delivering robust and meaningful results that outperform the symbolic encoding baselines. The findings reinforce the conclusion that subsymbolic encodings, especially those derived from pre-trained models, are better suited to capturing intricate dependencies and spatial relationships within TCR sequences, enabling more powerful and generalisable models for TCR repertoire classification tasks.

6.1.2 Comparison to Prior Work

Tables 5.4 and 5.5 also list results from Yuen's original dissertation [1] for each encoding method. In nearly all cases, the reproduced scores reported in this study are somewhat lower than those originally achieved by Yuen. One possible reason is a difference in training data size: while Yuen trained on 190 patient repertoires, this study used only 133, limiting the amount of available

repertoires to train on. Furthermore, while both studies relied on the TRACERx dataset for cancer repertoires, the composition of the files may have differed. In this study, the files were manually provided by the Supervisor (sourced from the Chain Lab RDS). It is highly possible that Yuen's version of the dataset included additional or even completely different patient cohorts, potentially contributing to different performance.

Another methodological difference lies in how the TCR α and β chains are treated. Yuen concatenated the two chains per patient into a single file, effectively increasing the number of TCR sequences per bag. This study trains models on α and β chain files completely separately. This approach provides a more controlled comparison but halves the average number of sequences per bag, which might have negatively impacted the model's capacity to generalise.

Together, these factors explain the differences in model performance to some extent. While this might pose some questions towards reproducibility, the overall trend remains consistent across the two studies: subsymbolic embeddings significantly outperform symbolic encodings, supporting the main conclusion that learned, context-aware representations are more effective for TCR-based cancer prediction.

6.1.3 Alpha vs Beta Chains

An additional point of interest lies in the comparison between α and β chains. The dominant view in the literature suggests that β chains offer greater junctional diversity [38], and are therefore often considered more informative for immunological classification tasks. Many existing studies rely only on β chain data during training, operating under the assumption that it contains the most relevant signal. It was therefore expected that β chains would yield superior performance in the current experiments as well.

However, the results presented reveal an opposite trend. For every tested encoding, the models trained on α chains consistently outperformed those trained on β chains. This is particularly notable considering that the training data included approximately half as many α chains as β chains (509,023 vs 992,826 for cancer samples; 2,115,248 vs 3,558,204 for control samples). Despite this difference in data volume, α chains provided better generalisation performance across all encodings.

Figure 6.1 presents a comparison of the mean test AUC scores for each TCR encoding type, evaluated separately on α and β chains (error bars represent the minimum and maximum AUC observed across repeated runs), clearly illustrating the superior performance of the α chains.

The pattern also holds for the evaluation set (results from SCEPTR). Here, α chains achieved a mean AUC of 0.814 (range: 0.657–1.000), while β chains reached a lower mean AUC of 0.698 (range: 0.570–0.860). Remarkably, one of the α chain models achieved a perfect AUC of 1.000 on the evaluation set, even further highlighting the discriminative power of TCR alpha chains.

Although unexpected and seemingly "contradictory" to prior research, these results provide evidence that α chains hold meaningful immunological information relevant to disease classification and should not be overlooked in TCR-based repertoire analysis.

6.2 Model Interpretability

6.2.1 Identifying Cancer-Associated TCRs

The objective of this experiment is to determine which TCR sequences most strongly influence the classification of cancerous vs healthy repertoires. Since the MIL model relies on the sparse attention

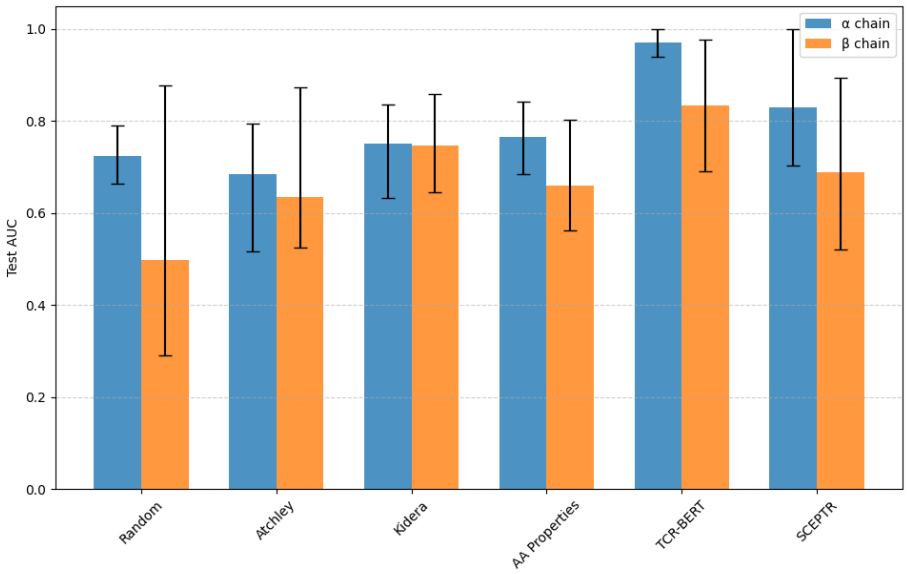


Figure 6.1: Mean test AUC with min/max ranges (α vs β), for each encoding

mechanism, it assigns attention weights to individual TCRs within a repertoire. Sequences that consistently achieve high attention weights in multiple patients' bags may correspond to cancer-associated immune activity.

The analysis focuses on the SCEPTR model, with attention weights extracted from the evaluation set for both the α and β chain models. For each case, the best-performing epoch is selected based on the highest test AUC achieved across the ten training repeats (specifically, run 0 at epoch 49 for the α chains, and run 0 at epoch 33 for the β chains).

For every patient repertoire in the evaluation set (11 cancer and 22 control patient files), the model computes embeddings for all TCR sequences and passes them through the attention-based classifier. Each TCR is assigned a scalar attention score, and the sequences are then ranked by descending attention weight. Top K (here, $K = 100$) TCRs per repertoire are retained for further analysis.

All patients' top-weight TCRs are collated into one table, indicating each TCR sequence, the associated patient, and the patient's label (cancer vs control). The next step is grouping them by occurrence counts in control vs cancer and creating a pivot table of the form

$$\texttt{(TCR, label)} \mapsto \texttt{count}$$

which provides the counts for both control and cancer groups, effectively indicating how many times each TCR appears across all patient repertoires in the evaluation set for each respective class.

Table 6.1 and Table 6.2 present the TCR CDR3 regions from α and β evaluation set, that were assigned high attention scores by the SCEPTR model and were observed in more than one cancer patient. For each TCR CDR3 sequence, the tables report how many times it has appeared in cancer and control samples, along with the range and mean of attention scores it received across those occurrences. Scores are reported in the format: minimum–maximum (μ: mean). Additionally, the percentile rank of that mean among all non-zero attention weights in the evaluation set is given.

The results offer several valuable insights. First, all listed TCRs appeared exclusively in cancer

TCR CDR3A	Count Cancer	Count Control	Weights	%ile
CAVNGGAGGTSYGKLTF	2	0	0.0433 - 0.1077 (μ: 0.0755)	84.80
CAVNAFRRAGGGADGLTF	**4**	0	0.0480 - 0.0480 (μ: 0.0480)	62.60
CAVNIGLTGGGNKLTF	**3**	0	0.0454 - 0.0454 (μ: 0.0454)	60.00
CAVNMYSGGGADGLTF	2	0	0.0398 - 0.0484 (μ: 0.0441)	57.60
CAVLNAGGTSYGKLTF	**3**	0	0.0182 - 0.0566 (μ: 0.0404)	53.60
CAVNIGSGGGADGLTF	2	0	0.0123 - 0.0487 (μ: 0.0305)	44.40
CAVNKGRNRGSTLGRLYF	2	0	0.0279 - 0.0279 (μ: 0.0279)	43.00
CAVNISPRAGSYQLTF	**9**	0	0.0267 - 0.0267 (μ: 0.0267)	39.20
CAVNPDRGSTLGRLYF	2	0	0.0104 - 0.0416 (μ: 0.0260)	36.00
CAVNSGGYSGAGSYQLTF	2	0	0.0225 - 0.0225 (μ: 0.0225)	31.40
CAVKRAGGTSYGKLTF	2	0	0.0095 - 0.0264 (μ: 0.0179)	25.20
CAVNSLHGSSNTGKLIF	2	0	0.0052 - 0.0265 (μ: 0.0159)	22.00

Table 6.1: High-attention α-chain TCRs shared by multiple cancer patients in the evaluation set.

TCR CDR3B	Count Cancer	Count Control	Weights	%ile
CASSYPPATGNSHLGEQYF	2	0	0.0975 - 0.1027 (μ: 0.1001)	99.60
CASSYSVVEGRAPHEQYF	2	0	0.0769 - 0.0829 (μ: 0.0799)	98.19
CASSEVRHRGPPNNEQFF	2	0	0.0547 - 0.0616 (μ: 0.0582)	93.16
CASSLGMARGPRQDTQYF	2	0	0.0499 - 0.0568 (μ: 0.0534)	90.34
CASSESSREGRLDTQYF	2	0	0.0368 - 0.0437 (μ: 0.0402)	85.11
CASLLGQGPYEQYF	2	0	0.0368 - 0.0386 (μ: 0.0377)	82.49
CASSHSGGLYEQYF	**3**	0	0.0271 - 0.0387 (μ: 0.0333)	77.87
CASMSRGGTTPFSGNTIYF	2	0	0.0228 - 0.0312 (μ: 0.0270)	69.82
CASSPMSGSHEQFF	2	0	0.0220 - 0.0308 (μ: 0.0264)	68.61
CASSYSGTGLPLYEQYF	2	0	0.0212 - 0.0280 (μ: 0.0246)	66.00
CASSPQAGGPPSKETQYF	2	0	0.0238 - 0.0238 (μ: 0.0238)	63.88
CASSLMQGGSHEQYF	2	0	0.0210 - 0.0210 (μ: 0.0210)	57.65
CASSFGTRSYEQYF	2	0	0.0121 - 0.0267 (μ: 0.0194)	55.73
CASSWPRTGASYEQYF	2	0	0.0145 - 0.0214 (μ: 0.0179)	53.72
CASSFALAGGPGANVLTF	2	0	0.0160 - 0.0182 (μ: 0.0171)	51.91
CASSYSGRRKPYSYEQYF	2	0	0.0125 - 0.0194 (μ: 0.0160)	47.89
CASSESGDRGKLYQPQHF	2	0	0.0150 - 0.0150 (μ: 0.0150)	46.98
CASSYRLAGGPPILTQYF	**3**	0	0.0082 - 0.0151 (μ: 0.0128)	41.05
CASSYFWGDQRDQYF	2	0	0.0068 - 0.0171 (μ: 0.0119)	38.83
CASSGTALVREGSSSYNEQFF	2	0	0.0073 - 0.0142 (μ: 0.0108)	36.62
CASSSPPPGGGLGETQYF	2	0	0.0100 - 0.0100 (μ: 0.0100)	34.10

Table 6.2: High-attention β-chain TCRs shared by multiple cancer patients in the evaluation set

repertoires and were not present among control patients, suggesting potential disease specificity. Several β-chain TCRs—such as `CASSYPPATGNSHLGEQYF` or `CASSYSVVEGRAPHEQYF`, were assigned exceptionally high attention weights, positioning them in the 90th+ percentile of all non-zero-weighted TCRs in the evaluation set. This indicates that the model consistently prioritised these sequences as informative across different patients. In contrast, α chains generally received slightly lower attention scores; however, they were observed more frequently across patient repertoires. For instance, `CAVNISPRAGSYQLTF` occurred in 9 out of 11 cancer patient files. This may imply a distinction in how the model processes the α and β chains. β chain sequences may act as highly specific but sparse markers, whereas α chain sequences may capture broader, more consistently recurring motifs.

Overall, these findings reaffirm the sparse attention mechanism's ability to highlight a small

subset of TCRs that collectively contribute to the classification outcome. While high-weight TCRs do not definitively prove tumour specificity, their consistent presence among clonotypes in multiple cancer patients points to potential immunological relevance. Further laboratory validation would be needed to confirm their role, but these tables provide concrete leads for potential follow-up studies examining whether the presence of these sequences might serve as an indication of lung cancer.

6.2.2 Alignment Between Scoring and Classifying Layers

A key design feature of the MIL architecture is the use of two separate linear layers (see Figure 5.2.3):

- **Scoring Layer**: assigns each TCR embedding a score, subsequently normalised via sparse attention.
- **Classifying Layer**: takes the resulting bag-level embedding and uses the sigmoid function produce a final prediction.

Each layer has its own learnable weight vector in $\mathbb{R}^d$. These vectors may converge to entirely different directions or align closely. However, the intuition is that both layers should ideally learn to map the embeddings into a shared, discriminative space that reflects cancer-relevant signal. Moreover, one might expect that across repeated training runs, the model would consistently learn similar directional structures, such that the scoring vectors from different runs align with one another (and likewise for the classifying vectors). This would indicate that the model is learning stable, biologically meaningful TCR repertoire representations.

Within-Run Alignment

The first experiment measures how the scoring layer vector aligns with the classifying layer vector in the same trained model. Let $\mathbf{w}_s$ denote the scoring layer weight and $\mathbf{w}_c$ denote the classifying layer weight. Both are extracted from the best checkpoint of a given run. Define:

$$cos\_sim(\mathbf{w}_s, \mathbf{w}_c) = \frac{\mathbf{w}_s \cdot \mathbf{w}_c}{\|\mathbf{w}_s\|\|\mathbf{w}_c\|}, \quad angle(\mathbf{w}_s, \mathbf{w}_c) = \arccos\big(cos\_sim(\mathbf{w}_s, \mathbf{w}_c)\big).$$

Cosine similarities near $+1$ imply near-parallel directions, whereas angles exceeding $90°$ indicate opposing directions. The experiment loads the best checkpoint for each run, normalises the two weight vectors, and computes both the cosine similarity and angle in degrees.

The results of the experiment are presented in Table 6.3. They, however, show weak alignment between the scoring and classifying layers in both α and β chain models. Cosine similarities are generally close to zero, and in several runs even negative, indicating that the two layers often project the embeddings in divergent directions. This is further supported by the consistently high angles between the vectors, typically above $80°$ and reaching over $100°$ in some cases. This suggests little to no directional agreement within individual models. Overall, the findings suggest that the scoring and classifying layers may be learning complementary, rather than redundant, mappings of the embedding space.

Run	Cosine Sim.	Angle (°)
0	0.070	85.97
1	0.007	89.57
2	-0.060	93.42
3	-0.181	100.45
4	0.134	82.31
5	-0.043	92.46
6	0.011	89.35
7	0.096	84.52
8	0.189	79.12
9	0.165	80.49

(a) Alignment in α-chain models

Run	Cosine Sim.	Angle (°)
0	0.022	88.71
1	-0.093	95.33
2	-0.048	92.77
3	-0.168	99.66
4	0.011	89.37
5	0.136	82.20
6	-0.188	100.82
7	-0.029	91.65
8	-0.136	97.79
9	0.096	84.50

(b) Alignment in β-chain models

Table 6.3: Cosine similarity and angle (in degrees) between the scoring and classifying layer weight vectors for each run

Across-Run Alignment

In addition, pairwise comparison was performed across multiple runs. For example, consider the scoring layer weights $\mathbf{w}_s^{(i)}$ from run i and $\mathbf{w}_s^{(j)}$ from run j. By plotting a cosine similarity matrix:

$$\left[cos\_sim\big(\mathbf{w}_s^{(i)}, \mathbf{w}_s^{(j)}\big) \right]_{i,j}$$

it becomes possible to see whether different runs converge to similar scoring directions in $\mathbb{R}^d$. A separate matrix is computed for the classifying layer.

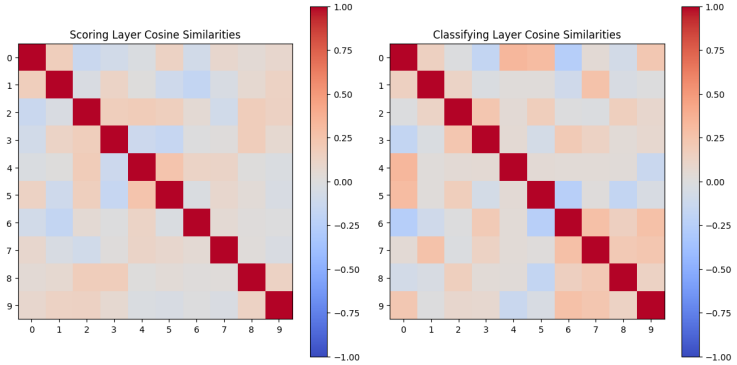
Figure 6.2 shows these matrices for SCEPTR models trained on α and β chain files. Similarly to the "within-run alignment" analysis, there is limited consistency in the directions learned by both the scoring and classifying layer vectors across repeated training runs. The cosine similarity matrices exhibit only modest off-diagonal alignment, suggesting that no single stable direction emerges in $\mathbb{R}^d$. This reaffirms previous statement that the optimisation landscape allows for multiple, similarly effective solutions, rather than enforcing a unique embedding direction. It also shows that the interpretability of learned vector directions is constrained and sensitive to random model initialisation.

6.2.3 2D Visualisations of Bag-Level Embeddings

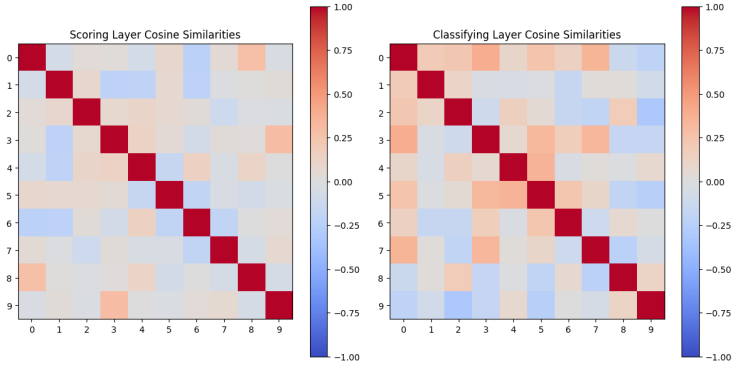
The final step in assessing learned TCR representations involves transforming the bag-level embeddings into a two-dimensional space and visualising them. A straightforward approach is to apply a dimensionality reduction technique such as UMAP or t-SNE on the bag-level vectors generated by the classifier (see *vector representation of the patient's repertoire* in Figure 5.1). If the classifier has learned a meaningful separation between cancer and control patients, their respective 2D embeddings should appear as distinct clusters.

To perform the visualisations, this study applies UMAP (Uniform Manifold Approximation and Projection) [57], a non-linear dimensionality reduction technique for projecting high-dimensional data into two dimensions. The procedure is conducted separately for models trained on α and β chain data, using the best-performing SCEPTR checkpoint across ten runs and fifty epochs, based on the highest test-set AUC achieved.

For each patient in the evaluation set, regardless of label, their TCR sequences are first encoded using SCEPTR, and the resulting embeddings are passed through the trained classifier's



(a) Cosine similarity matrix of learned directions across training runs (α chains)



(b) Cosine similarity matrix of learned directions across training runs (β chains)

Figure 6.2: Across-run cosine similarities of scoring and classifying layer vectors, plotted as symmetric heatmaps for (a) α-chain and (b) β-chain models. Each matrix captures pairwise similarities across the 10 repeated training runs.

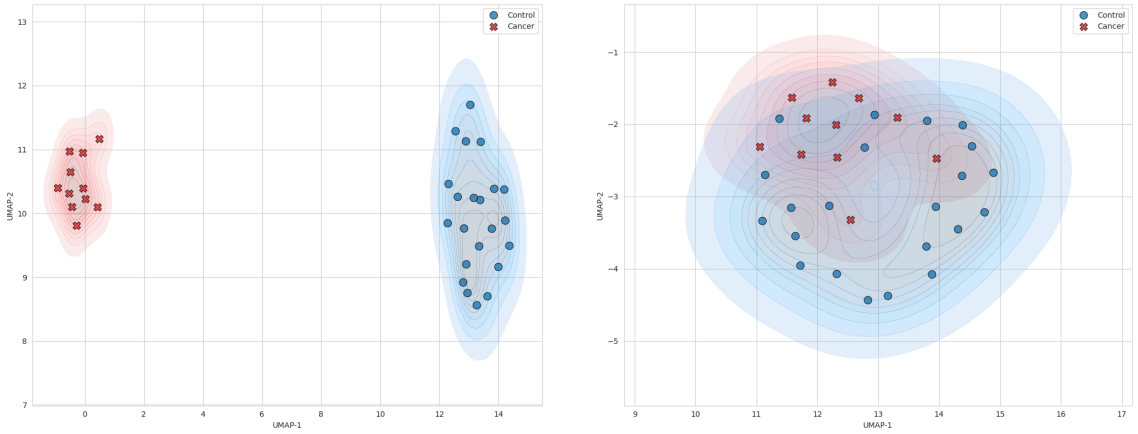
attention layer. The resulting attention-weighted embeddings are then summed to yield a single 64-dimensional vector representing the entire repertoire. These vectors are aggregated into a matrix $\mathbf{X} \in \mathbb{R}^{N \times 64}$, where N denotes the number of evaluation patients. This matrix is then passed through UMAP, producing a 2D representation $\mathbf{Z} \in \mathbb{R}^{N \times 2}$. Finally, each 2D point is plotted with a colour indicating the corresponding label (cancer or control).

Figure 6.3 illustrates the representative visualisations. In case of the α-chain model (Figure 6.3 (a)), the control and cancer patient embeddings form two clearly separable clusters, with no spatial overlap. Cancer repertoires are concentrated tightly on the left side of the plot, while control repertoires appear more dispersed but consistently located on the right. This separation suggests that the SCEPTR model has successfully learned class-discriminative features from the α-chain input alone. The tight grouping of cancer samples may indicate strong consistency in disease-associated signals across patients, while the greater variance in the control group is consistent with the biological heterogeneity expected among healthy individuals [58].

In contrast, the β-chain UMAP projection (Figure 6.3 (b)) reveals significant overlap between cancer and control repertoire embeddings. Although some spatial structure is present, both classes occupy a shared region in the latent space, and their distributions (as visualised by the KDE contours) are largely interleaved. This suggests that the β-chain model was much less effective at learning a robust separation between the two groups. This outcome is particularly notable given the common assumption in the literature that β chains offer greater junctional diversity and

stronger diagnostic signal [38].

These results align with the model performance metrics reported in Chapter 5 which showed that models trained on α-chains consistently outperformed those trained on β-chains in terms of test AUC and accuracy. Together, the visual and quantitative results indicate that α-chain TCR sequences, despite being traditionally underused, do carry important immunological information relevant for cancer classification.



(a) UMAP of α-chain bag embeddings

(b) UMAP of β-chain bag embeddings

Figure 6.3: 2D UMAP visualisations of patient-level TCR repertoire embeddings generated by SCEPTR, shown separately for (a) α-chain and (b) β-chain models. Each point represents a single patient's repertoire, coloured by diagnosis (cancer or control).

Chapter 7

Conclusions

7.1 Achievements and Key Findings

7.2 Limitations

7.3 Future Works

Bibliography

[1] R. C. Yuen, "Multi-instance transfer learning on t cell receptor llms for cancer prediction." April 2024.

[2] D. Crosby, S. Bhatia, K. M. Brindle, L. M. Coussens, C. Dive, M. Emberton, S. Esener, R. C. Fitzgerald, S. S. Gambhir, P. Kuhn, T. R. Rebbeck, and S. Balasubramanian, "Early detection of cancer," *Science*, vol. 375, no. 6586, p. eaay9040, 2022.

[3] R. C. Fitzgerald, A. C. Antoniou, L. Fruk, and N. Rosenfeld, "The future of early cancer detection," *Nature Medicine*, vol. 28, no. 4, pp. 666–677, 2022.

[4] D. D. Chaplin, "Overview of the immune response," *Journal of Allergy and Clinical Immunology*, vol. 125, no. 2, Supplement 2, pp. S3–S23, 2010. 2010 Primer on Allergic and Immunologic Diseases.

[5] 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, p. eaaz3738, 2020.

[6] X. Feng, M. Huo, H. Li, Y. Yang, Y. Jiang, L. He, and S. Cheng Li, "A comprehensive benchmarking for evaluating tcr embeddings in modeling tcr-epitope interactions," *Briefings in Bioinformatics*, vol. 26, p. bbaf030, 01 2025.

[7] J.-W. Sidhom, H. B. Larman, P. Ross-MacDonald, M. Wind-Rotolo, D. M. Pardoll, and A. S. Baras, "Deeptcr: a deep learning framework for understanding t-cell receptor sequence signatures within complex t-cell repertoires," *bioRxiv*, 2019.

[8] 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, p. 469, 2022.

[9] 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, pp. 1671–1680, 04 2019.

[10] J. Jeon, S. Yu, S. Lee, S. C. Kim, H.-Y. Jo, I. Jung, and K. Kim, "Epicpred: predicting phenotypes driven by epitope-binding tcrs using attention-based multiple instance learning," *Bioinformatics*, vol. 41, p. btaf080, 02 2025.

[11] Y. Nagano, A. Pyo, M. Milighetti, J. Henderson, J. Shawe-Taylor, B. Chain, and A. Tiffeau-Mayer, "Contrastive learning of t cell receptor representations," 2024.

[12] T. J. Kindt, R. A. Goldsby, B. A. Osborne, and J. Kuby, *Kuby Immunology*. Macmillan, 2007.

[13] M. G. Rudolph and I. A. Wilson, "The specificity of tcr/pmhc interaction," *Current Opinion in Immunology*, vol. 14, no. 1, pp. 52–65, 2002.

[14] Bio-Rad Laboratories, Inc., "An overview of t cell receptors," 2016.

[15] R. A. Mariuzza, P. Agnihotri, and J. Orban, "The structural basis of t-cell receptor (tcr) activation: An enduring enigma," *Journal of Biological Chemistry*, vol. 295, pp. 914–925, 2019.

[16] A. Lanzavecchia, G. Iezzi, and A. Viola, "From tcr engagement to t cell activation: A kinetic view of t cell behavior," *Cell*, vol. 96, January 1999.

[17] K. Shah, A. Al-Haidari, J. Sun, and J. U. Kazi, "T cell receptor (tcr) signaling in health and disease," *Signal Transduction and Targeted Therapy*, vol. 6, 2021.

[18] X. Hou, M. Wang, C. Lu, Q. Xie, G. Cui, J. Chen, Y. Du, Y. Dai, and H. Diao, "Analysis of the repertoire features of tcr beta chain cdr3 in human by high-throughput sequencing," *Cellular Physiology and Biochemistry*, vol. 39, pp. 651–667, 07 2016.

[19] A. K. Sewell, "Why must t cells be cross-reactive?," *PubMed Central*, vol. 12, August 2012.

[20] M. S. Hirsch and J. Watkins, "A comprehensive review of biomarker use in the gynecologic tract including differential diagnoses and diagnostic pitfalls," *Advances in Anatomic Pathology*, vol. 27, no. 3, pp. 179–197, 2020.

[21] R. R. T., O. N. A. Demerdash, and J. C. Smith, "Tcr-h: explainable machine learning prediction of t-cell receptor epitope binding on unseen datasets," *Frontiers in Immunology*, vol. 15, 2024.

[22] A. Aran, L. Garrigós, G. Curigliano, J. Cortés, and M. Martí, "Evaluation of the tcr repertoire as a predictive and prognostic biomarker in cancer: Diversity or clonality?," *Cancers*, vol. 14, no. 7, 2022.

[23] K. Joshi, M. Milighetti, and B. M. Chain, "Application of t cell receptor (tcr) repertoire analysis for the advancement of cancer immunotherapy," *Current Opinion in Immunology*, vol. 74, pp. 1–8, 2022.

[24] J.-H. Cui, K.-R. Lin, S.-H. Yuan, Y.-B. Jin, X.-P. Chen, X.-K. Su, J. Jiang, Y.-M. Pan, S.-L. Mao, X.-F. Mao, and W. Luo, "Tcr repertoire as a novel indicator for immune monitoring and prognosis assessment of patients with cervical cancer," *Frontiers in Immunology*, vol. 9, p. 2729, 2018.

[25] A. Sesma, J. Pardo, M. Cruellas, E. M. Gálvez, M. Gascón, D. Isla, L. Martínez-Lostao, M. Ocáriz, J. R. Paño, E. Quílez, A. Ramírez, I. Torres-Ramón, A. Yubero, M. Zapata, and R. Lastra, "From tumor mutational burden to blood t cell receptor: Looking for the best predictive biomarker in lung cancer treated with immunotherapy," *Cancers*, vol. 12, no. 10, p. 2974, 2020.

[26] J. Han, J. Duan, H. Bai, Y. Wang, R. Wan, X. Wang, S. Chen, Y. Tian, D. Wang, K. Fei, Z. Yao, S. Wang, Z. Lu, Z. Wang, and J. Wang, "Tcr repertoire diversity of peripheral pd-1+cd8+ t cells predicts clinical outcomes after immunotherapy in patients with non–small cell lung cancer," *Cancer Immunology Research*, vol. 8, pp. 146–154, 01 2020.

[27] S. M. Liu, C. Chen, Y. Zhang, W. Wang, Y. Wang, J. Zhang, F. Teng, J. Li, Q. Wang, and J. Wang, "Specific tcr profiles predict clinical outcome of adjuvant egfr-tkis for resected egfr-mutant non-small cell lung cancer," *Biomarker Research*, vol. 11, no. 1, p. 26, 2023.

[28] J. Han, R. Yu, J. Duan, J. Li, W. Zhao, G. Feng, H. Bai, Y. Wang, X. Zhang, R. Wan, J. Xu, X. Wang, Y. Guan, X. Xia, Z. Yao, K. Fei, D. P. Carbone, Z. Wang, and J. Wang, "Weighting tumor-specific tcr repertoires as a classifier to stratify the immunotherapy delivery in non–small cell lung cancers," *Science Advances*, vol. 7, no. 21, p. eabd6971, 2021.

[29] I. Springer, N. Tickotsky, and Y. Louzoun, "Contribution of t cell receptor alpha and beta cdr3, mhc typing, v and j genes to peptide binding prediction," *Frontiers in Immunology*, vol. 12, p. 664514, 2021. eCollection 2021.

[30] D. S. Fischer, Y. Wu, B. Schubert, and F. J. Theis, "Predicting antigen specificity of single t cells based on tcr cdr3 regions," *Molecular Systems Biology*, vol. 16, no. 8, p. e9416, 2020.

[31] 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, no. 18, pp. 6395–6400, 2005.

[32] 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, pp. 23–55, Feb. 1985.

[33] 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, no. 1676, p. 20140243, 2015.

[34] 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," *bioRxiv*, 2021.

[35] J. A. Pai and A. T. Satpathy, "High-throughput and single-cell t cell receptor sequencing technologies," *Nature Methods*, vol. 18, no. 8, pp. 881–892, 2021.

[36] B. Howie, A. M. Sherwood, A. D. Berkebile, J. Berka, R. O. Emerson, D. W. Williamson, I. Kirsch, M. Vignali, M. J. Rieder, C. S. Carlson, and H. S. Robins, "High-throughput pairing of t cell receptor and sequences," *Science Translational Medicine*, vol. 7, no. 301, pp. 301ra131–301ra131, 2015.

[37] 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, L. Tian, C. M. Weyand, G. Georgiou, and J. J. Goronzy, "Determinants governing t cell receptor /-chain pairing in repertoire formation of identical twins," *Proceedings of the National Academy of Sciences*, vol. 117, no. 1, pp. 532–540, 2020.

[38] P. Dash, A. J. Fiore-Gartland, T. Hertz, G. C. Wang, S. Sharma, A. Souquette, J. C. Crawford, E. B. Clemens, T. H. O. Nguyen, K. Kedzierska, N. L. La Gruta, P. Bradley, and P. G. Thomas, "Quantifiable predictive features define epitope-specific t cell receptor repertoires," *Nature*, vol. 547, no. 7661, pp. 89–93, 2017.

[39] O. Maron and T. Lozano-Pérez, "A framework for multiple-instance learning," in *Advances in Neural Information Processing Systems* (M. Jordan, M. Kearns, and S. Solla, eds.), vol. 10, MIT Press, 1997.

[40] D. Xiong, Z. Zhang, T. Wang, and X. Wang, "A comparative study of multiple instance learning methods for cancer detection using t-cell receptor sequences," *Computational and Structural Biotechnology Journal*, vol. 19, pp. 3255–3268, 2021.

[41] Y. Xu, X. Qian, X. Zhang, X. Lai, Y. Liu, and J. Wang, "Deeplion: Deep multi-instance learning improves the prediction of cancer-associated t cell receptors for accurate cancer detection," *Frontiers in Genetics*, vol. 13, p. 860510, 2022.

[42] X. Qian, G. Yang, F. Li, X. Zhang, X. Zhu, X. Lai, X. Xiao, T. Wang, and J. Wang, "Deeplion2: deep multi-instance contrastive learning framework enhancing the prediction of cancer-associated t cell receptors by attention strategy on motifs," *Frontiers in Immunology*, vol. 15, p. 1345586, 2024.

[43] Z. Song, X. Chen, Y. Shi, R. Huang, W. Wang, K. Zhu, S. Lin, M. Wang, G. Tian, J. Yang, and G. Chen, "Evaluating the potential of t cell receptor repertoires in predicting the prognosis of resectable non-small cell lung cancers," *Molecular Therapy Methods & Clinical Development*, vol. 18, pp. 73–83, 2020.

[44] M. Jamal-Hanjani, A. Hackshaw, Y. Ngai, J. Shaw, C. Dive, S. Quezada, G. Middleton, E. de Bruin, J. Le Quesne, S. Shafi, M. Falzon, S. Horswell, F. Blackhall, I. Khan, S. Janes, M. Nicolson, D. Lawrence, M. Forster, D. Fennell, S.-M. Lee, J. Lester, K. Kerr, S. Muller, N. Iles, S. Smith, N. Murugaesu, R. Mitter, M. Salm, A. Stuart, N. Matthews, H. Adams, T. Ahmad, R. Attanoos, J. Bennett, N. J. Birkbak, R. Booton, G. Brady, K. Buchan, A. Capitano, M. Chetty, M. Cobbold, P. Crosbie, H. Davies, A. Denison, M. Djearman, J. Goldman, T. Haswell, L. Joseph, M. Kornaszewska, M. Krebs, G. Langman, M. MacKenzie, J. Millar, B. Morgan, B. Naidu, D. Nonaka, K. Peggs, C. Pritchard, H. Remmen, A. Rowan, R. Shah, E. Smith, Y. Summers, M. Taylor, S. Veeriah, D. Waller, B. Wilcox, M. Wilcox, I. Woolhouse, N. McGranahan, and C. Swanton, "Tracking genomic cancer evolution for precision medicine: The lung tracerx study," *PLOS Biology*, vol. 12, pp. 1–7, 07 2014.

[45] R. C. Yuen, "Github: Tcr-cancer-prediction." `https://github.com/RcwYuen/TCR-Cancer-Prediction`, 2024. Accessed: 2025-03-31.

[46] M. Jamal-Hanjani, G. Wilson, S. Horswell, R. Mitter, O. Sakarya, T. Constantin, R. Salari, E. Kirkizlar, S. Sigurjonsson, R. Pelham, S. Kareht, B. Zimmermann, and C. Swanton, "Detection of ubiquitous and heterogeneous mutations in cell-free dna from patients with early-stage non-small-cell lung cancer," *Annals of Oncology*, vol. 27, no. 5, pp. 862–867, 2016.

[47] E. C. de Bruin, N. McGranahan, R. Mitter, M. Salm, D. C. Wedge, L. Yates, M. Jamal-Hanjani, S. Shafi, N. Murugaesu, A. J. Rowan, E. Grönroos, M. A. Muhammad, S. Horswell, M. Gerlinger, I. Varela, D. Jones, J. Marshall, T. Voet, P. V. Loo, D. M. Rassl, R. C. Rintoul, S. M. Janes, S.-M. Lee, M. Forster, T. Ahmad, D. Lawrence, M. Falzon, A. Capitanio, T. T. Harkins, C. C. Lee, W. Tom, E. Teefe, S.-C. Chen, S. Begum, A. Rabinowitz, B. Phillimore, B. Spencer-Dene, G. Stamp, Z. Szallasi, N. Matthews, A. Stewart, P. Campbell, and C. Swanton, "Spatial and temporal diversity in genomic instability processes defines lung cancer evolution," *Science*, vol. 346, no. 6206, pp. 251–256, 2014.

[48] N. McGranahan, F. Favero, E. C. de Bruin, N. J. Birkbak, Z. Szallasi, and C. Swanton, "Clonal status of actionable driver events and the timing of mutational processes in cancer evolution," *Science Translational Medicine*, vol. 7, no. 283, pp. 283ra54–283ra54, 2015.

[49] N. McGranahan, A. J. S. Furness, R. Rosenthal, S. Ramskov, R. Lyngaa, S. K. Saini, M. Jamal-Hanjani, G. A. Wilson, N. J. Birkbak, C. T. Hiley, T. B. K. Watkins, S. Shafi, N. Murugaesu, R. Mitter, A. U. Akarca, J. Linares, T. Marafioti, J. Y. Henry, E. M. V. Allen, D. Miao, B. Schilling, D. Schadendorf, L. A. Garraway, V. Makarov, N. A. Rizvi, A. Snyder, M. D. Hellmann, T. Merghoub, J. D. Wolchok, S. A. Shukla, C. J. Wu, K. S. Peggs, T. A. Chan, S. R. Hadrup, S. A. Quezada, and C. Swanton, "Clonal neoantigens elicit t cell immunoreactivity and sensitivity to immune checkpoint blockade," *Science*, vol. 351, no. 6280, pp. 1463–1469, 2016.

[50] Y. Nagano and B. Chain, "tidytcells: standardizer for tr/mh nomenclature," *Frontiers in Immunology*, vol. 14, p. 1276106, October 25 2023.

[51] A. Martins and R. Astudillo, "From softmax to sparsemax: A sparse model of attention and multi-label classification," in *Proceedings of The 33rd International Conference on Machine Learning* (M. F. Balcan and K. Q. Weinberger, eds.), vol. 48 of *Proceedings of Machine Learning Research*, (New York, New York, USA), pp. 1614–1623, PMLR, 20–22 Jun 2016.

[52] A. N. Holm, D. Wright, and I. Augenstein, "Revisiting softmax for uncertainty approximation in text classification," *Information*, vol. 14, no. 7, 2023.

[53] UCL Computer Science Technical Support Group, "Gpu facilities for cs students." `https://tsg.cs.ucl.ac.uk/gpus/`, 2023. Accessed: 2025-03-31.

[54] PyTorch Contributors, *torch.nn.BCELoss*. The PyTorch Foundation, 2024. Accessed: 2025-04-04.

[55] Google Developers, "Classification: Accuracy, precision, recall." `https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall`, 2025. Accessed: 2025-04-04.

[56] Google Developers, "Classification: Roc and auc." `https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc`, 2024. Accessed: 2025-04-04.

[57] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," 2020.

[58] Y. H. Sun, Y.-L. Wu, and B.-Y. Liao, "Phenotypic heterogeneity in human genetic diseases: ultrasensitivity-mediated threshold effects as a unifying molecular mechanism," *Journal of Biomedical Science*, vol. 30, p. 58, jul 2023.

Appendix A

GitHub Repository

Appendix B

Figures

B.1 Model Training Results

B.1.1 Random Encodings

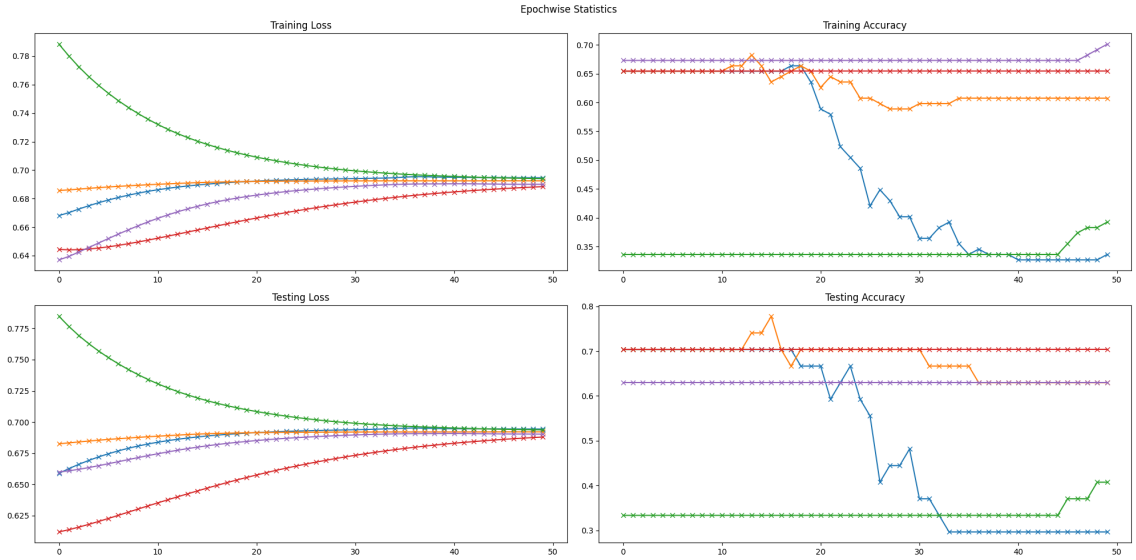


Figure B.1: Loss & Accuracy on Training and Test Sets for each Epoch (α chains)

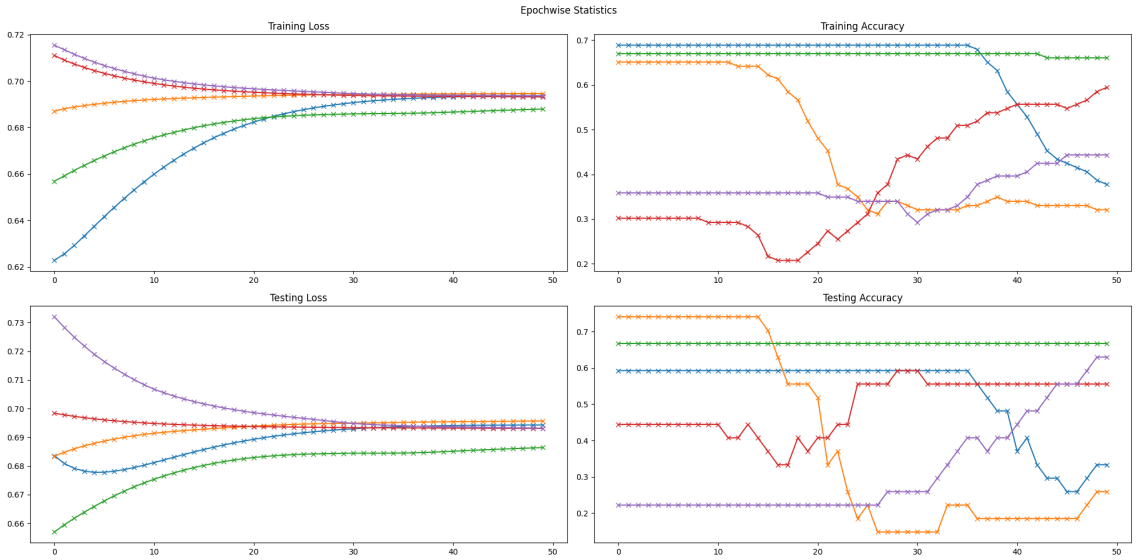


Figure B.2: Loss & Accuracy on Training and Test Sets for each Epoch (β chains)

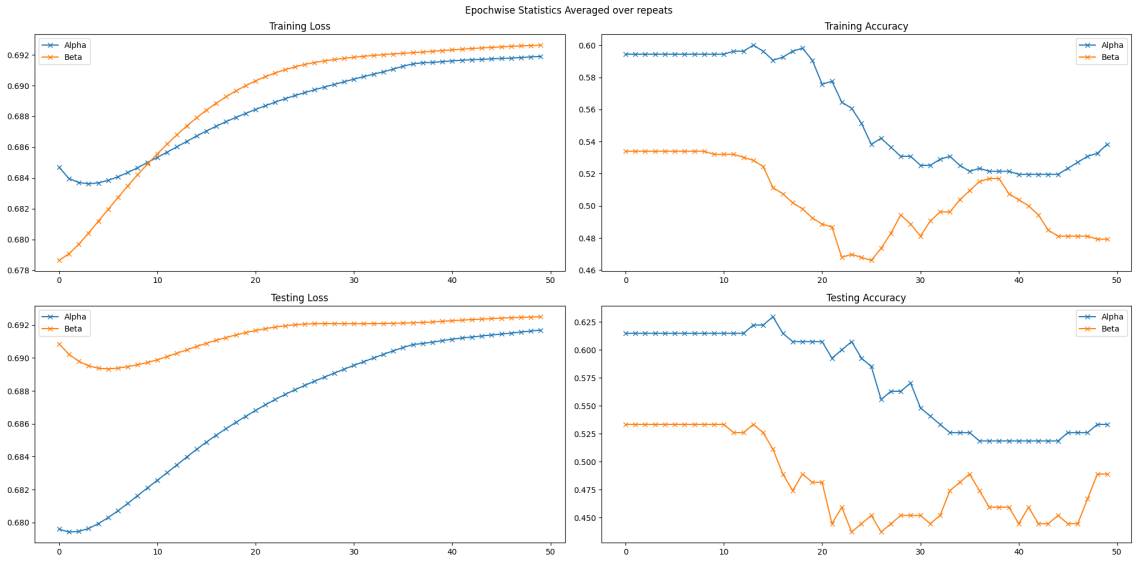


Figure B.3: Loss & Accuracy on Training and Test Sets Averaged on Repeats (α vs β chains)

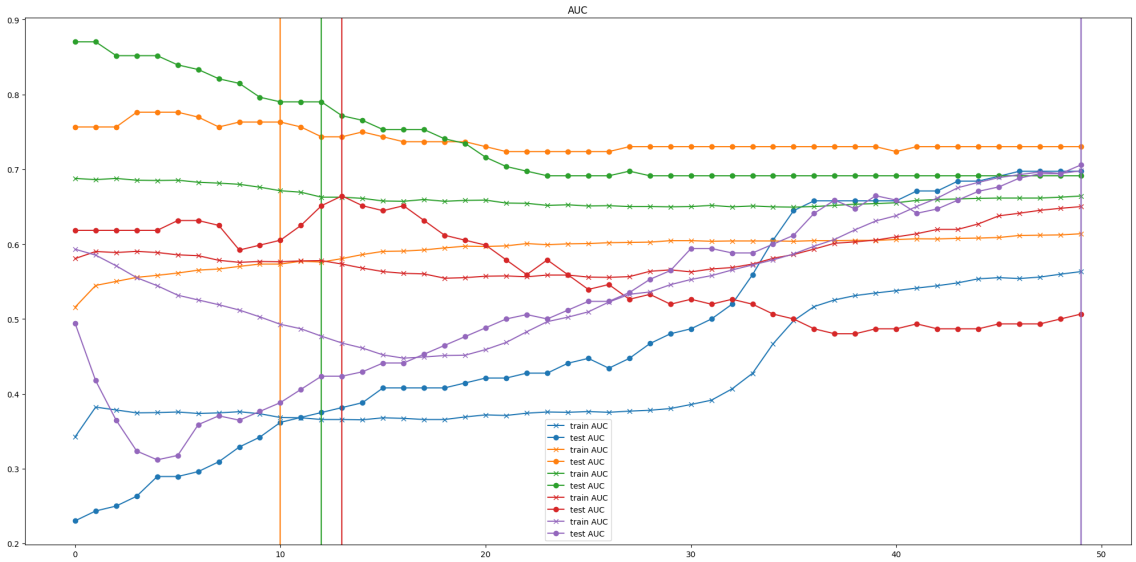


Figure B.4: AUC on Training and Test Sets (α chains)

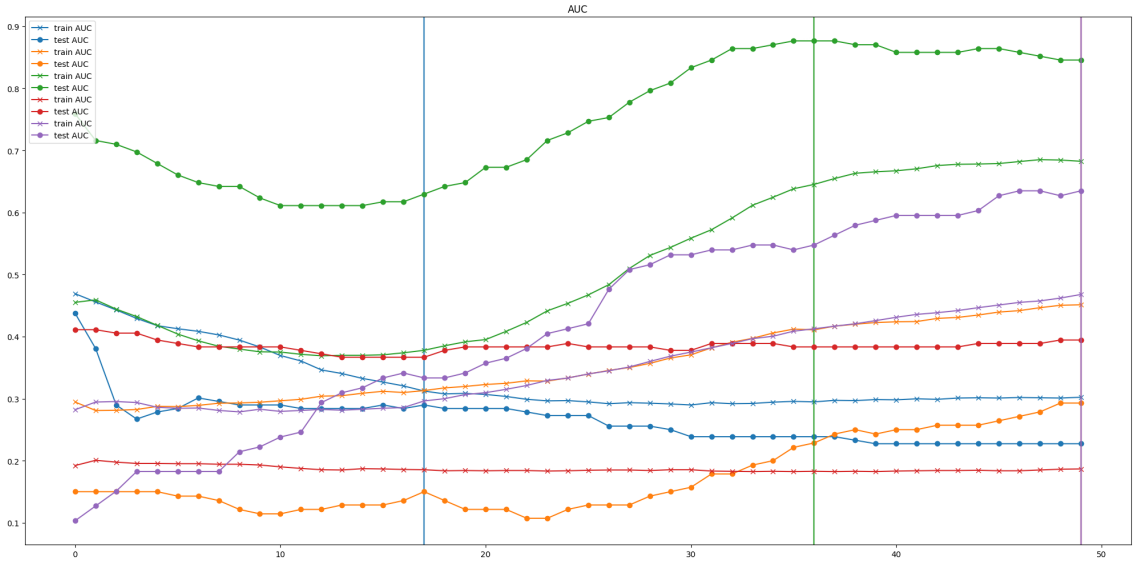


Figure B.5: AUC on Training and Test Sets (β chains)

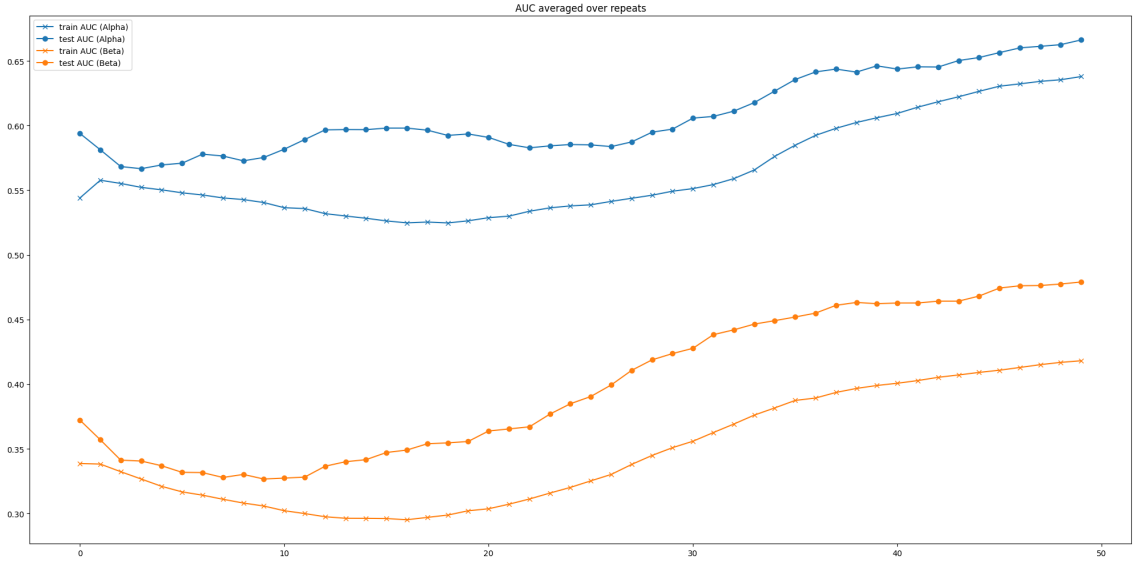


Figure B.6: AUC Averaged on Training and Test Sets (α vs β chains)

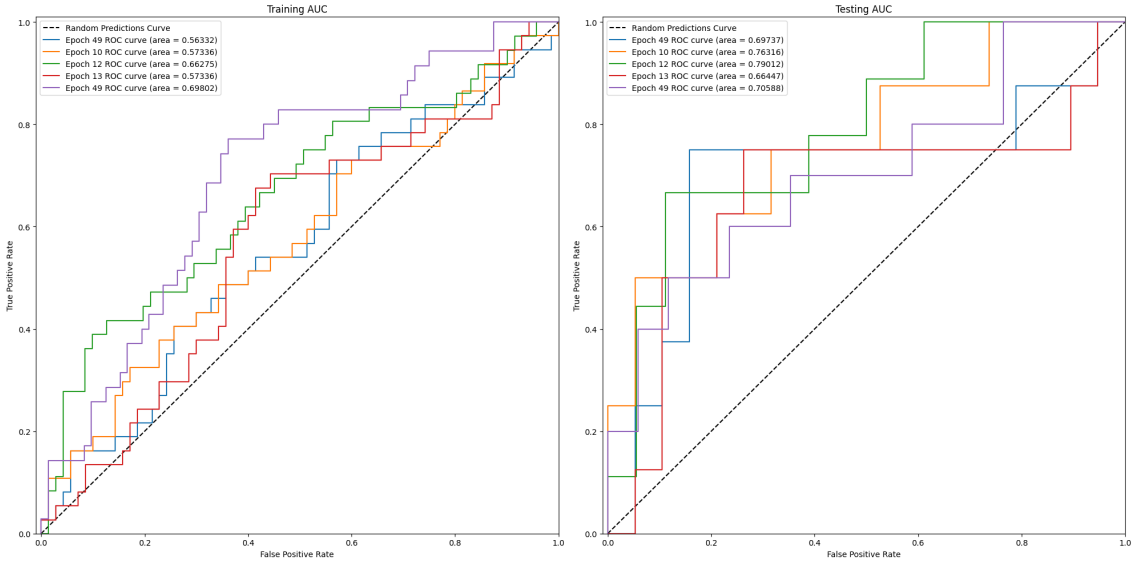


Figure B.7: AUC Curve for each repeated run (α chains)

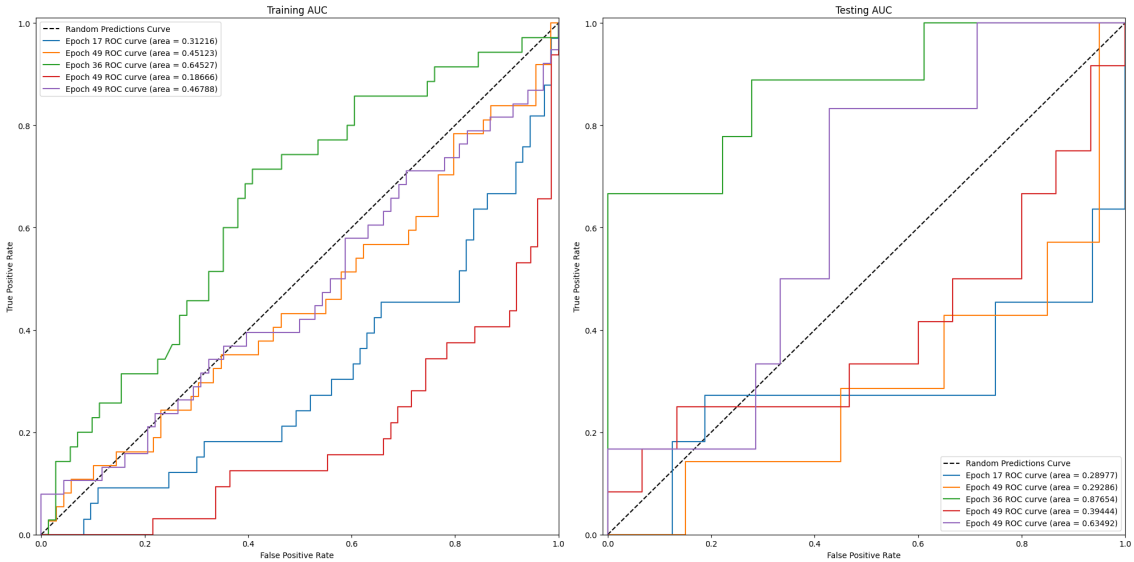


Figure B.8: AUC Curve for each repeated run (β chains)

B.1.2 Atchley Factors

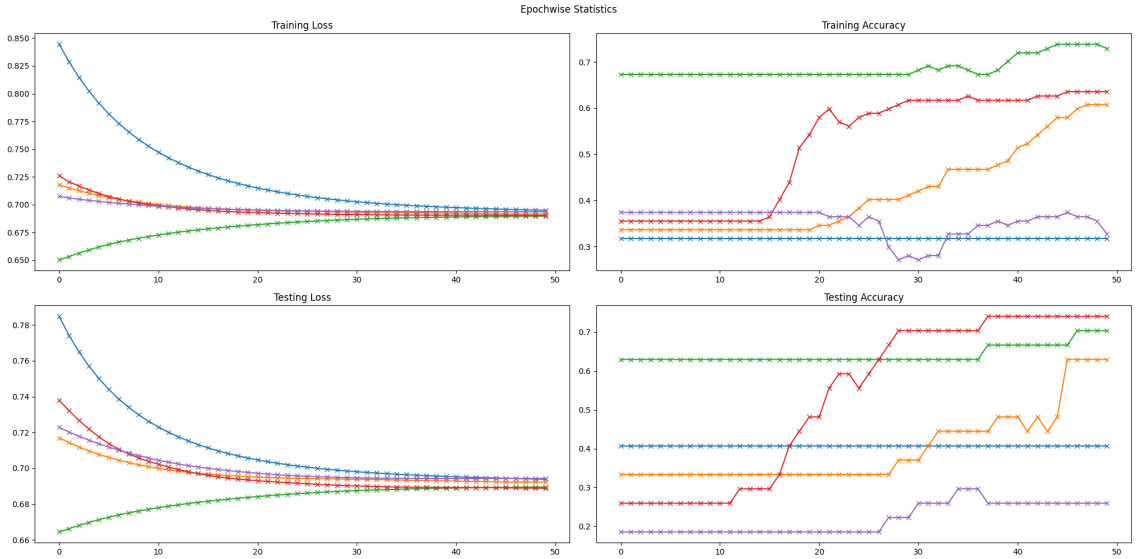


Figure B.9: Loss & Accuracy on Training and Test Sets for each Epoch (α chains)

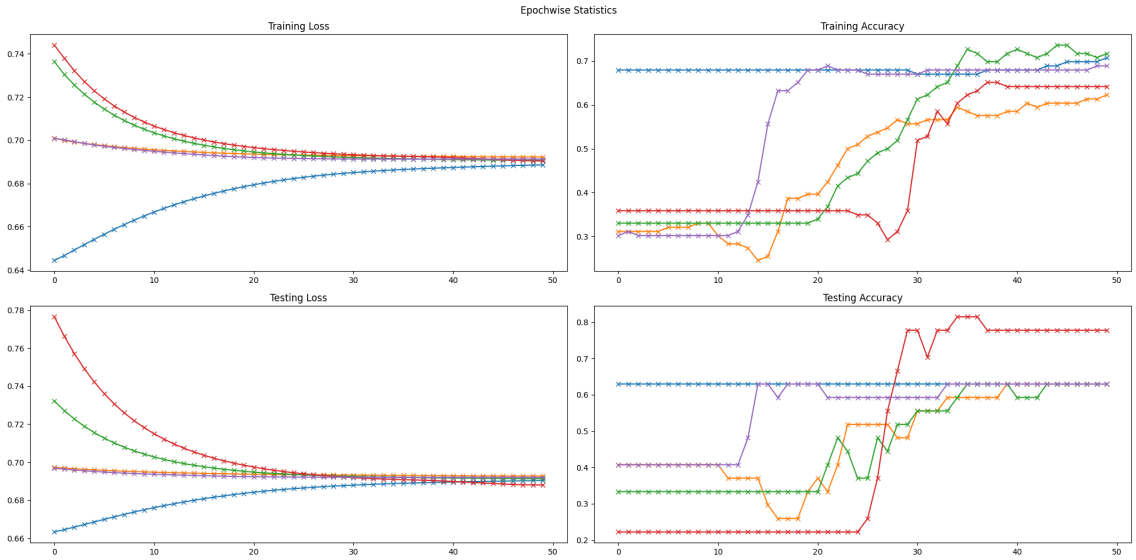


Figure B.10: Loss & Accuracy on Training and Test Sets for each Epoch (β chains)

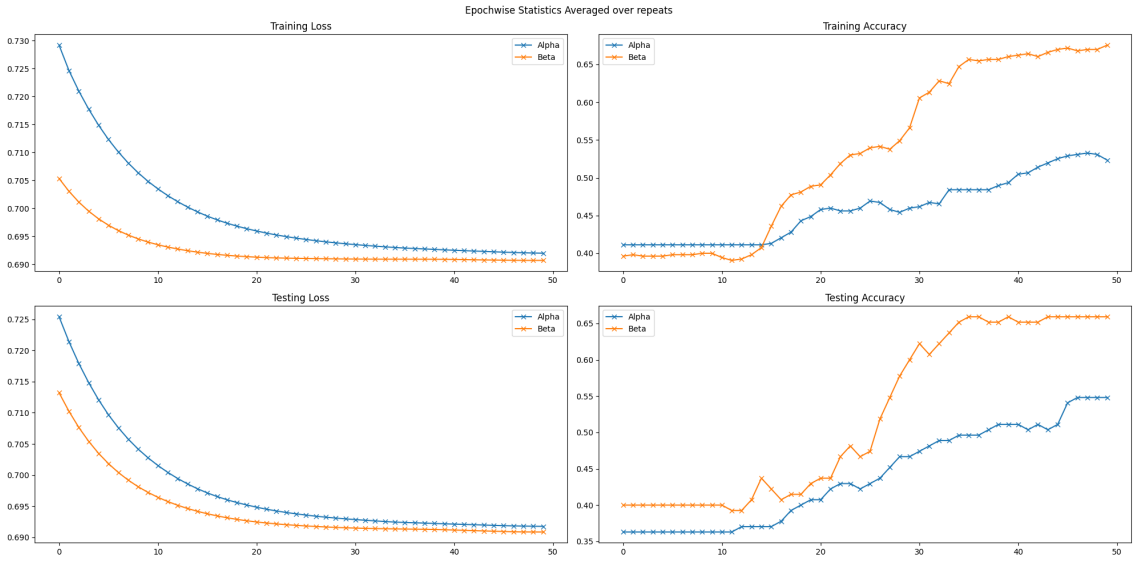


Figure B.11: Loss & Accuracy on Training and Test Sets Averaged on Repeats (α vs β chains)

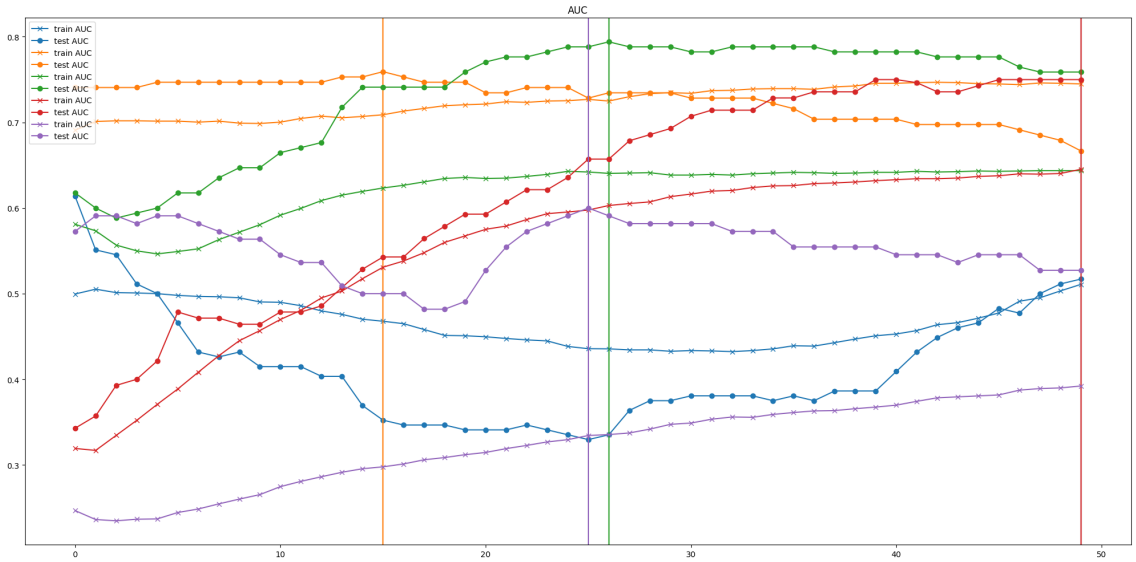


Figure B.12: AUC on Training and Test Sets (α chains)

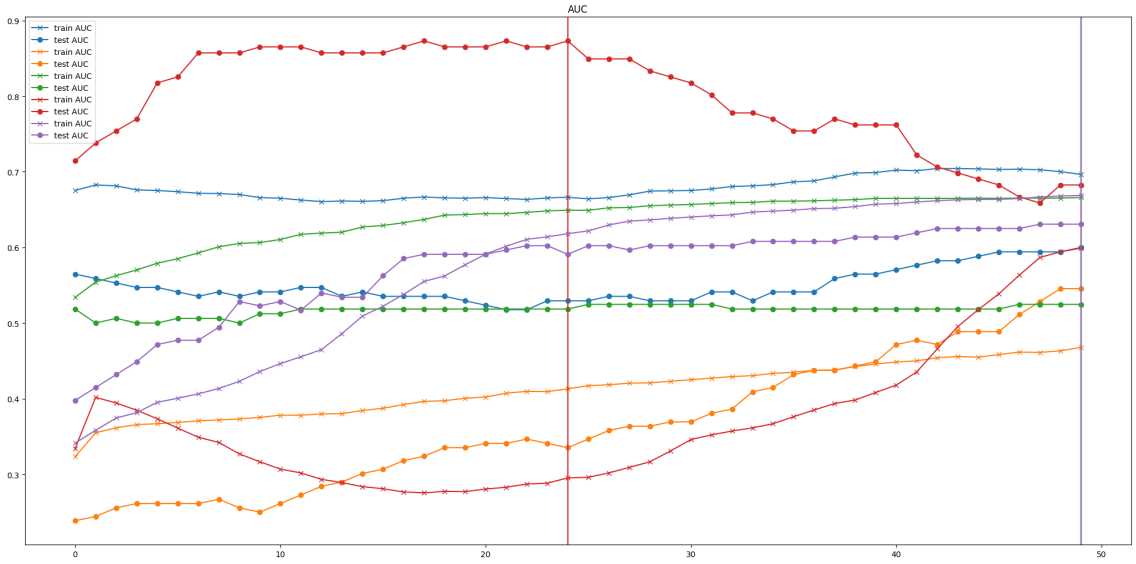


Figure B.13: AUC on Training and Test Sets (β chains)

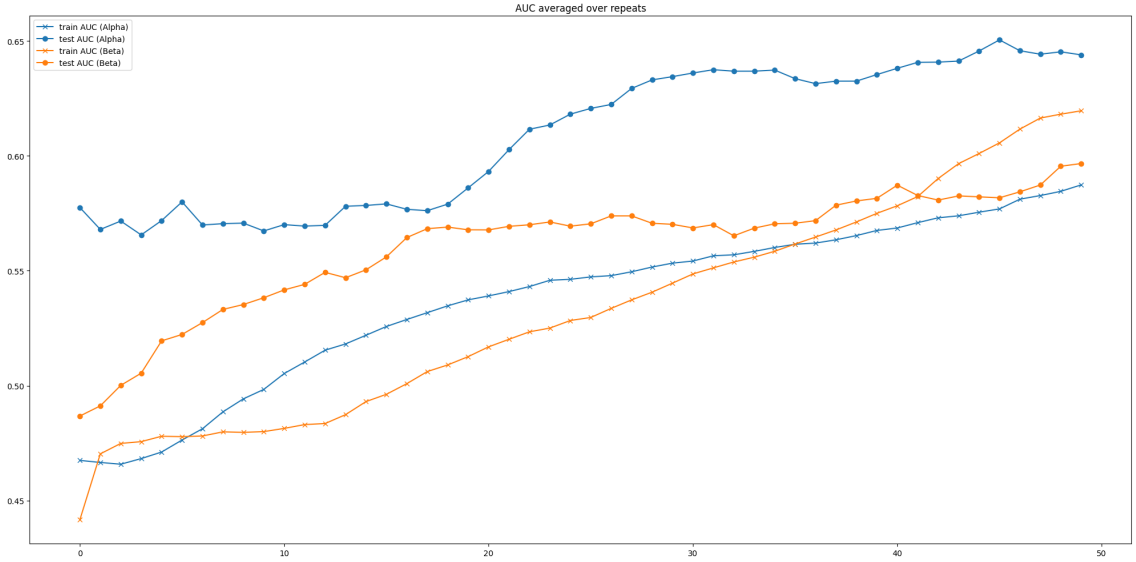


Figure B.14: AUC Averaged on Training and Test Sets (α vs β chains)

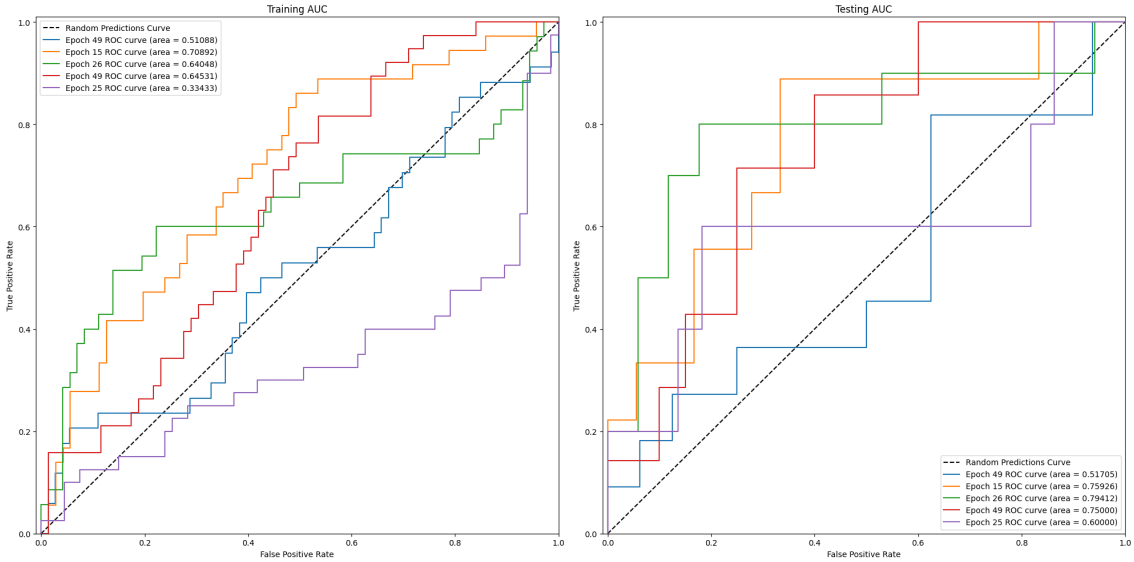


Figure B.15: AUC Curve for each repeated run (α chains)

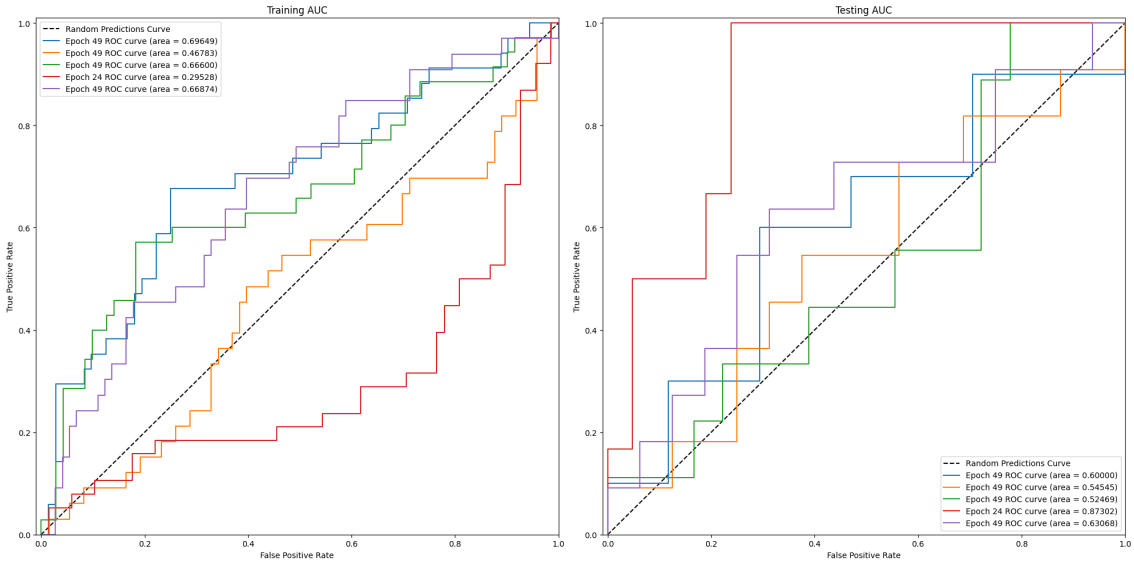


Figure B.16: AUC Curve for each repeated run (β chains)

B.1.3 Kidera Factors

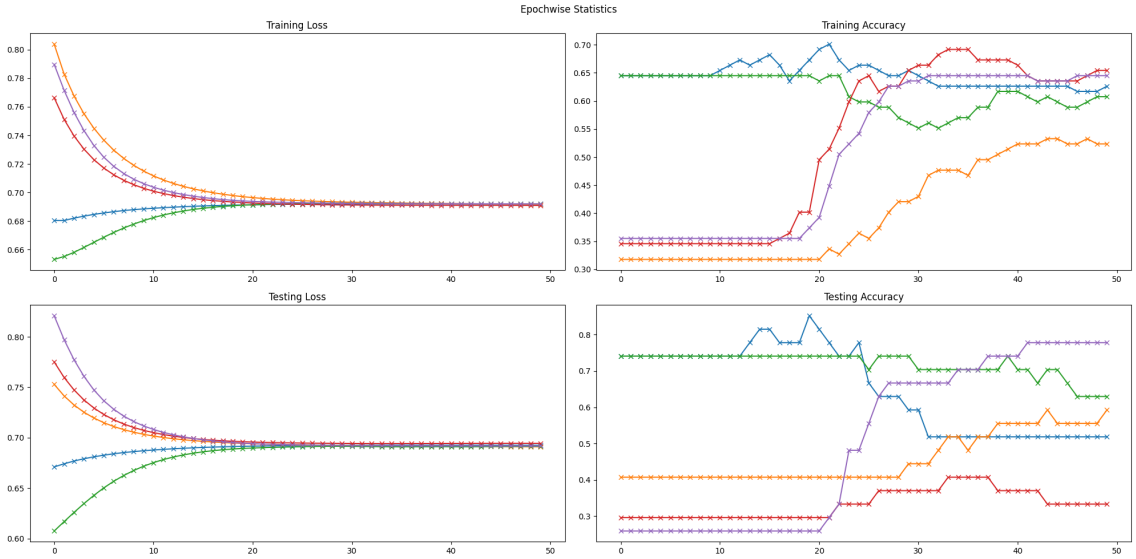


Figure B.17: Loss & Accuracy on Training and Test Sets for each Epoch (α chains)

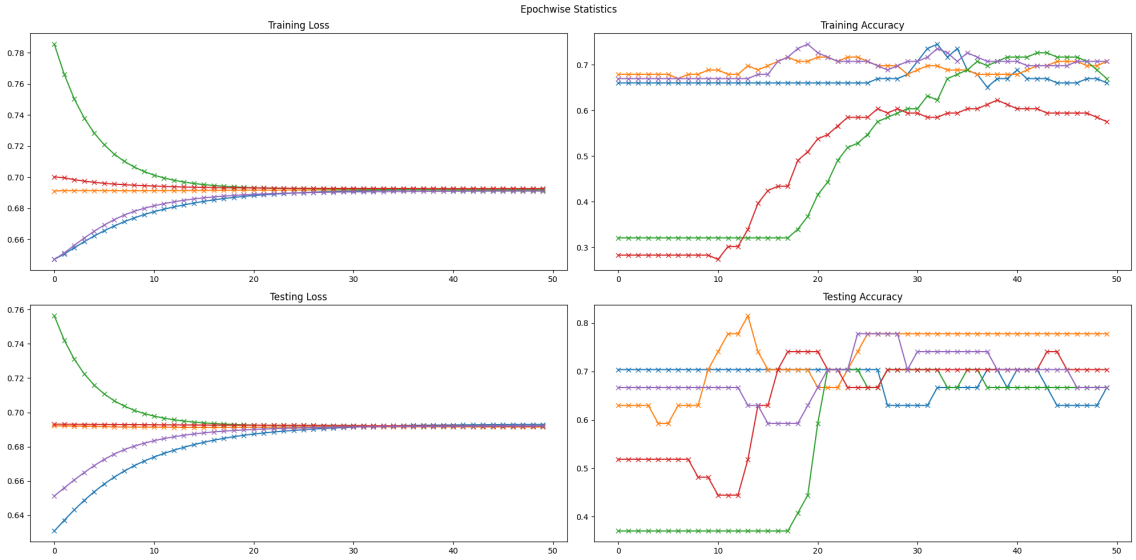


Figure B.18: Loss & Accuracy on Training and Test Sets for each Epoch (β chains)

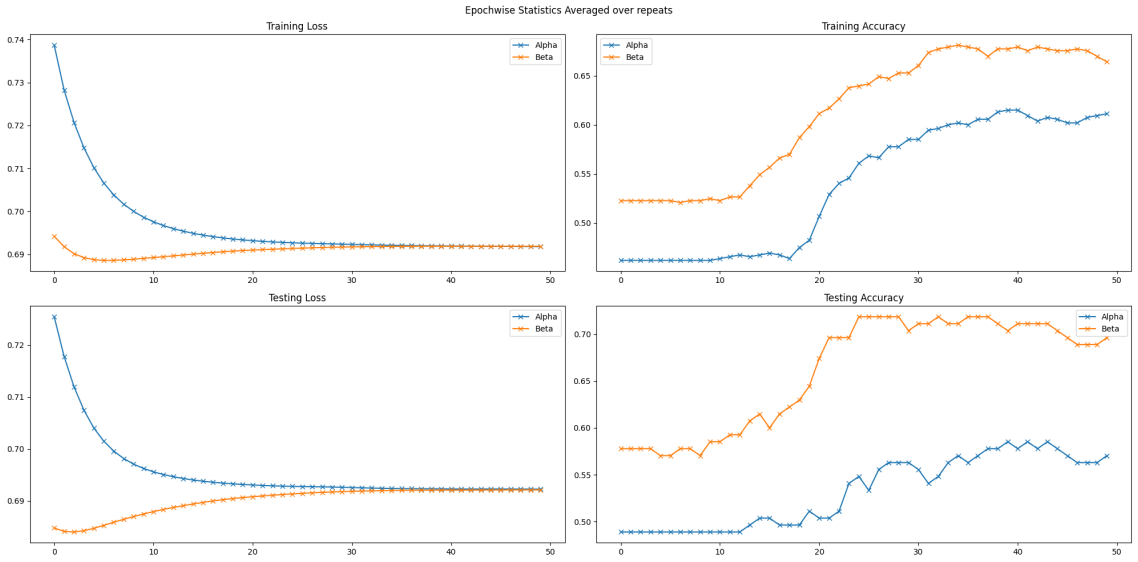


Figure B.19: Loss & Accuracy on Training and Test Sets Averaged on Repeats (α vs β chains)

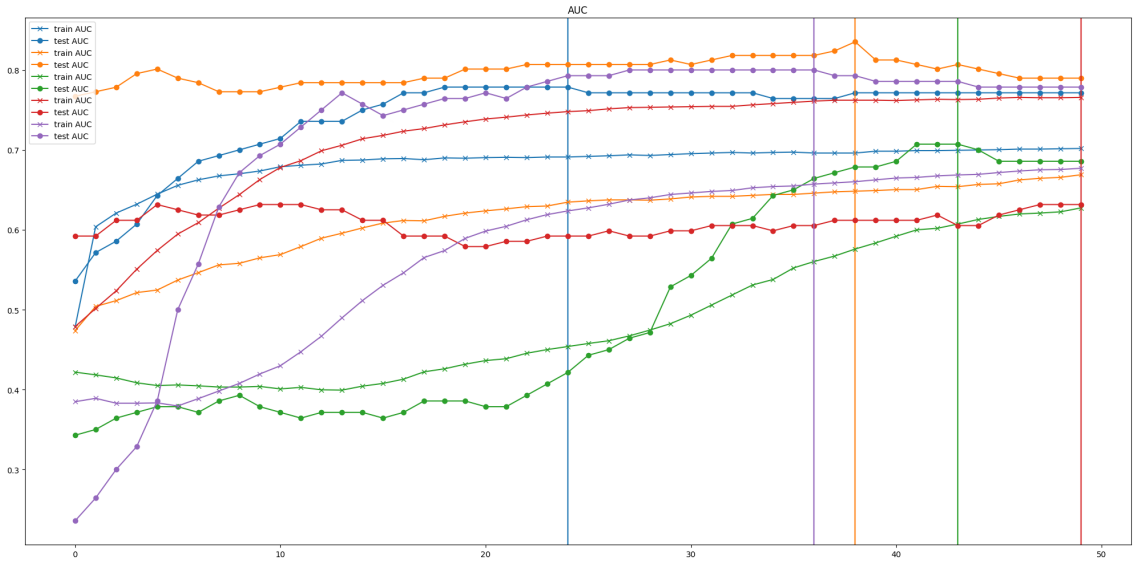


Figure B.20: AUC on Training and Test Sets (α chains)

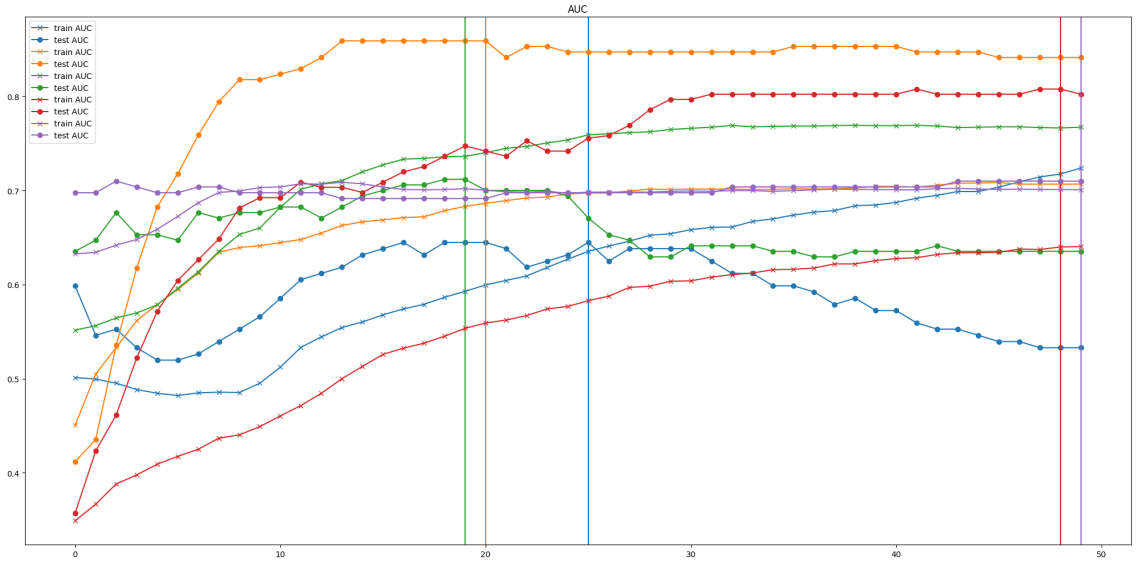


Figure B.21: AUC on Training and Test Sets (β chains)

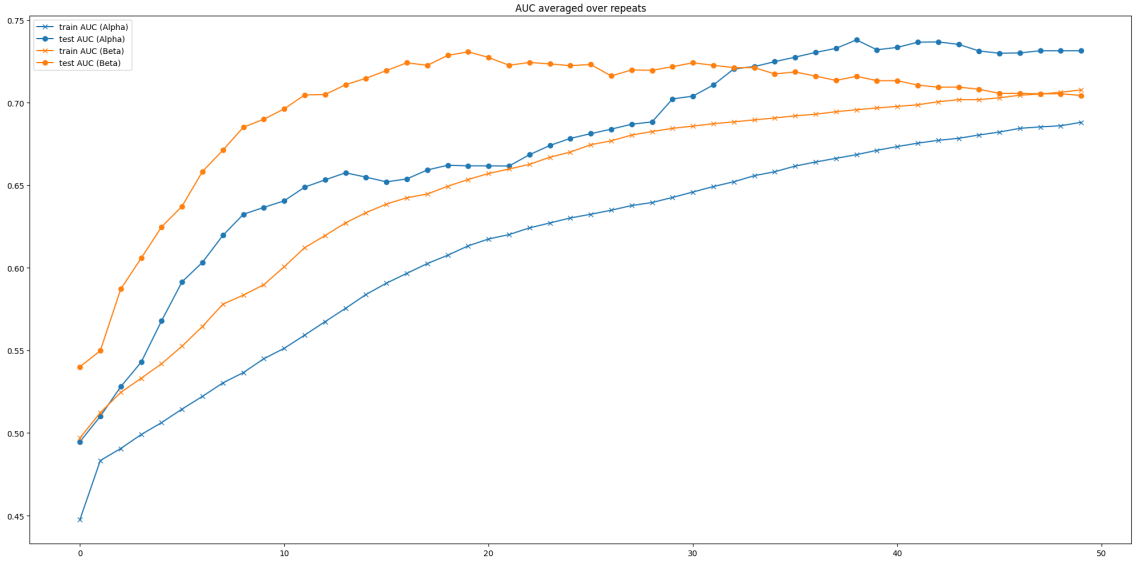


Figure B.22: AUC Averaged on Training and Test Sets (α vs β chains)

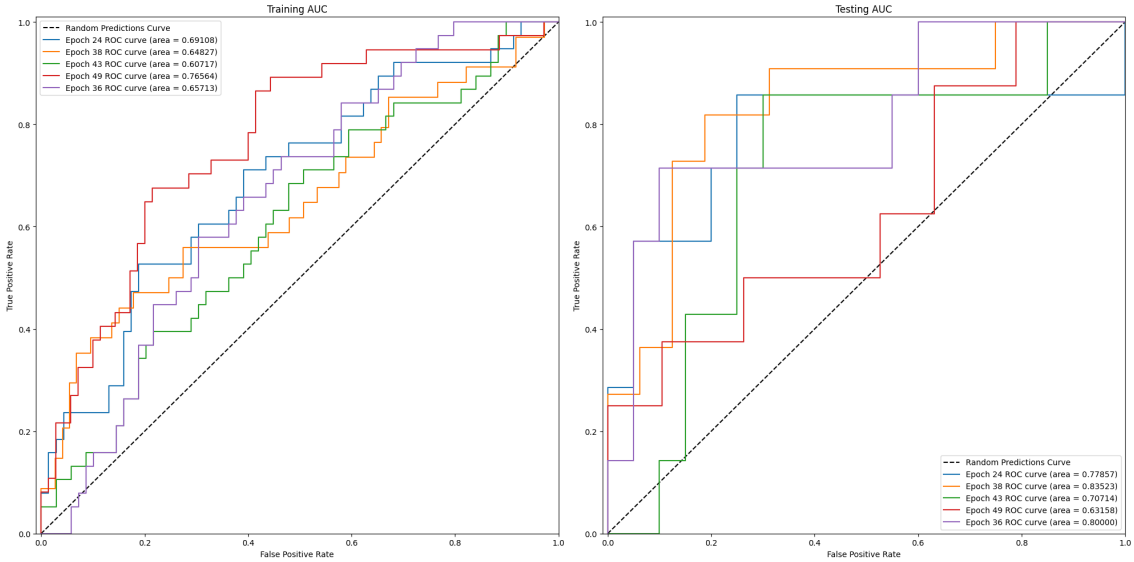


Figure B.23: AUC Curve for each repeated run (α chains)

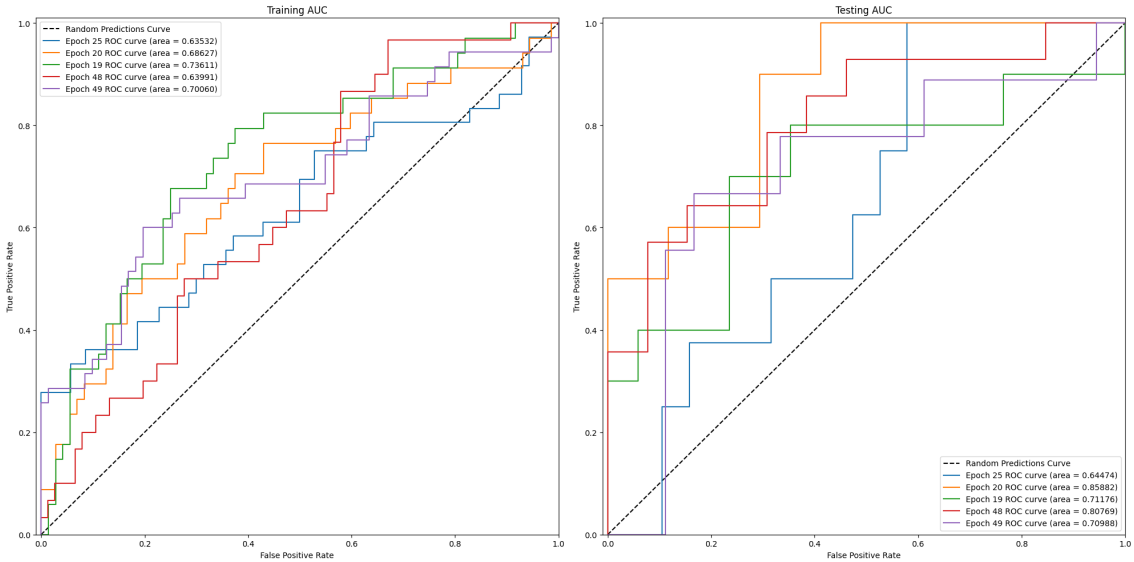


Figure B.24: AUC Curve for each repeated run (β chains)

B.1.4 Amino Acid Properties

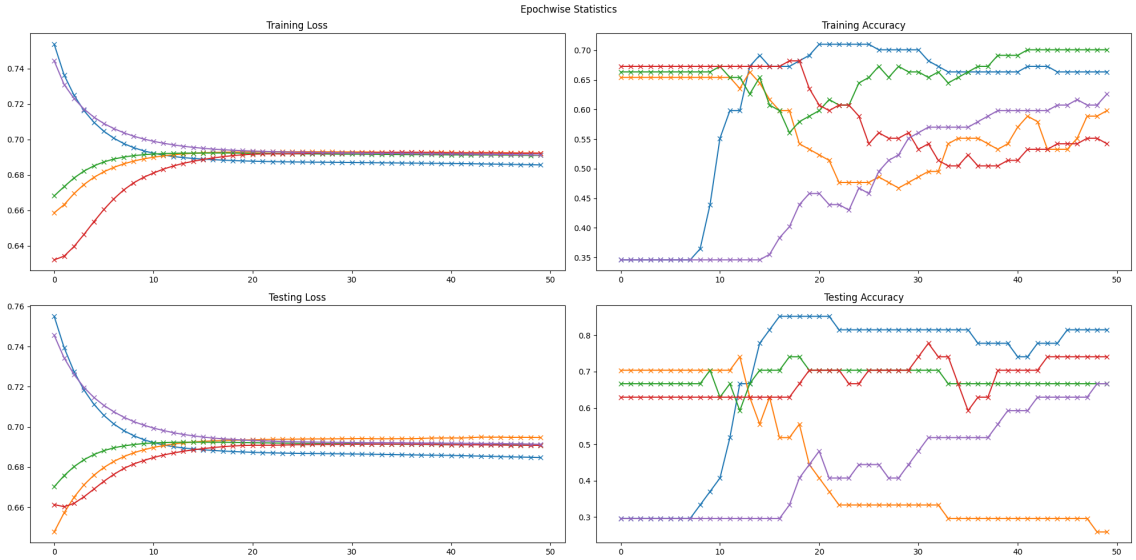


Figure B.25: Loss & Accuracy on Training and Test Sets for each Epoch (α chains)

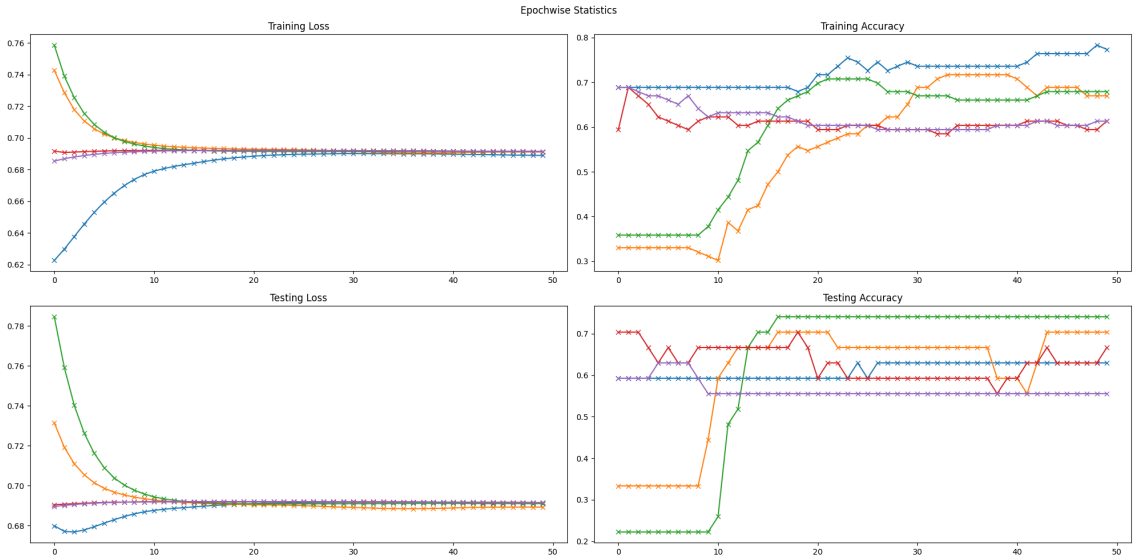


Figure B.26: Loss & Accuracy on Training and Test Sets for each Epoch (β chains)

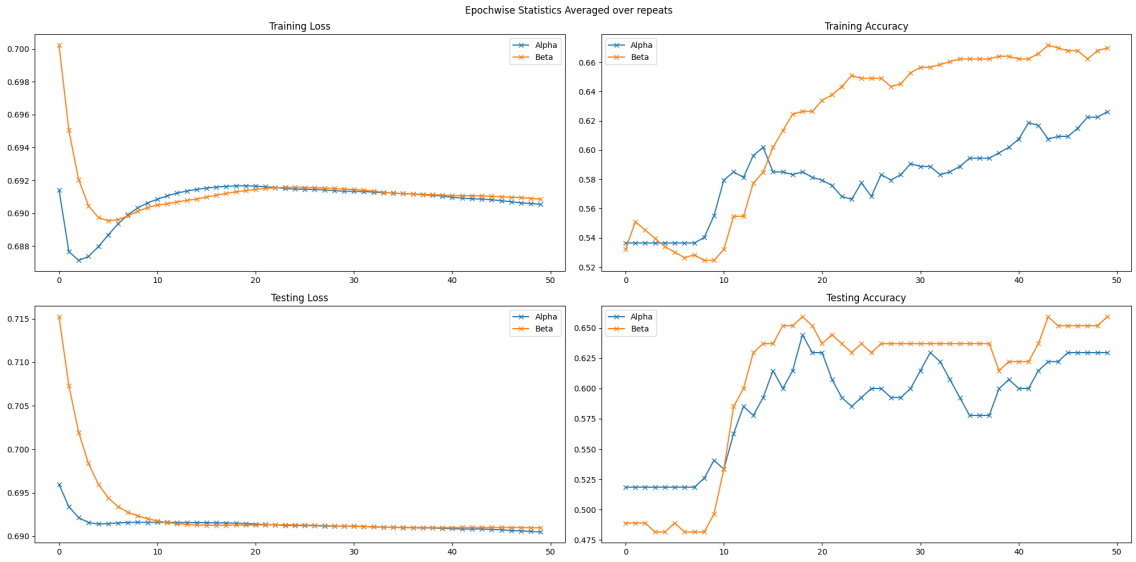


Figure B.27: Loss & Accuracy on Training and Test Sets Averaged on Repeats (α vs β chains)

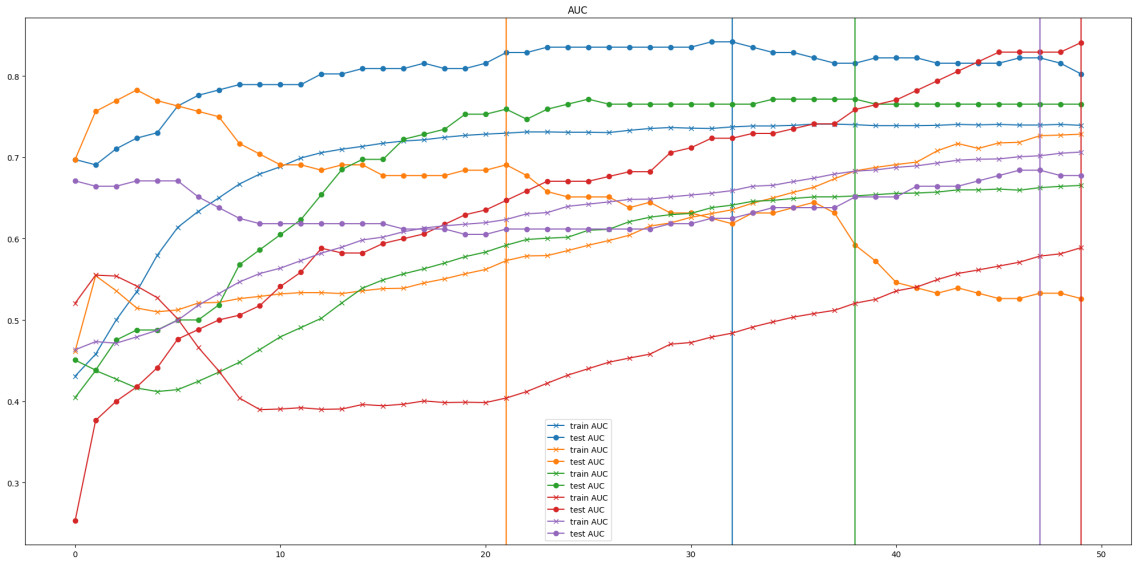


Figure B.28: AUC on Training and Test Sets (α chains)

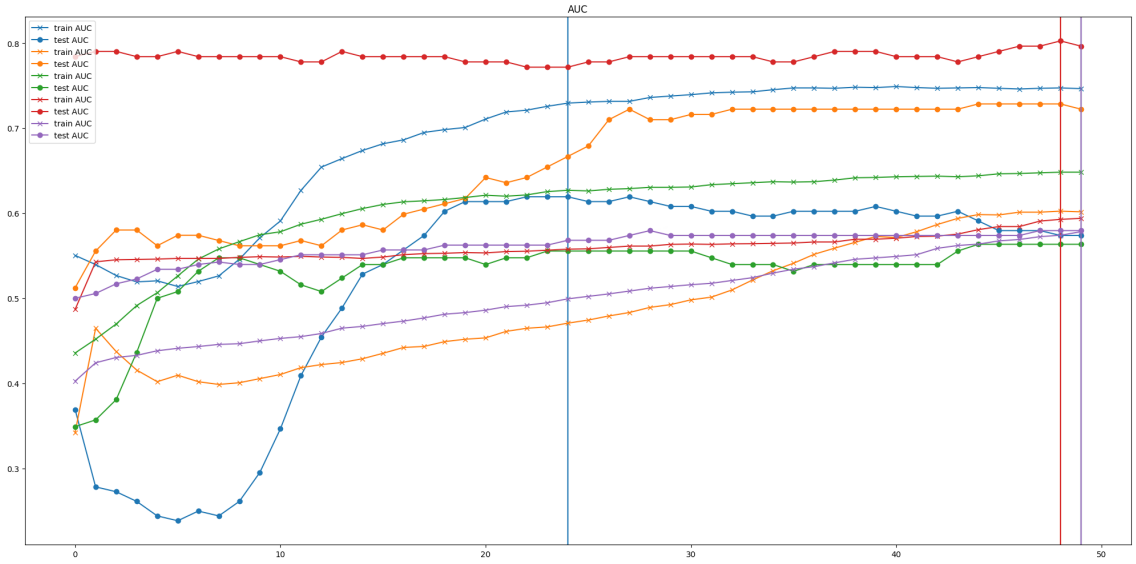


Figure B.29: AUC on Training and Test Sets (β chains)

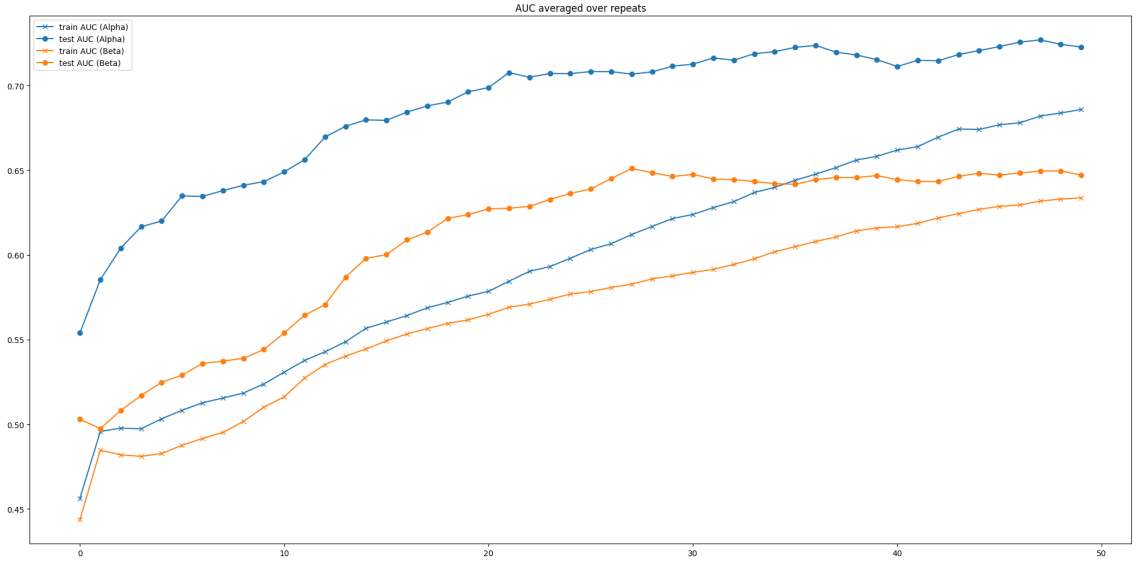


Figure B.30: AUC Averaged on Training and Test Sets (α vs β chains)

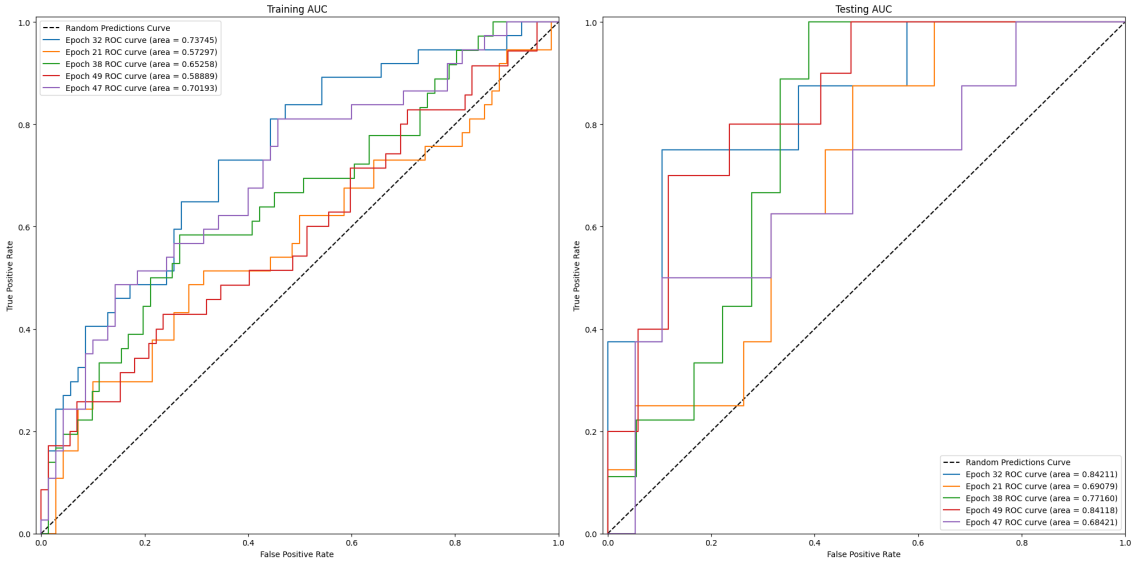


Figure B.31: AUC Curve for each repeated run (α chains)

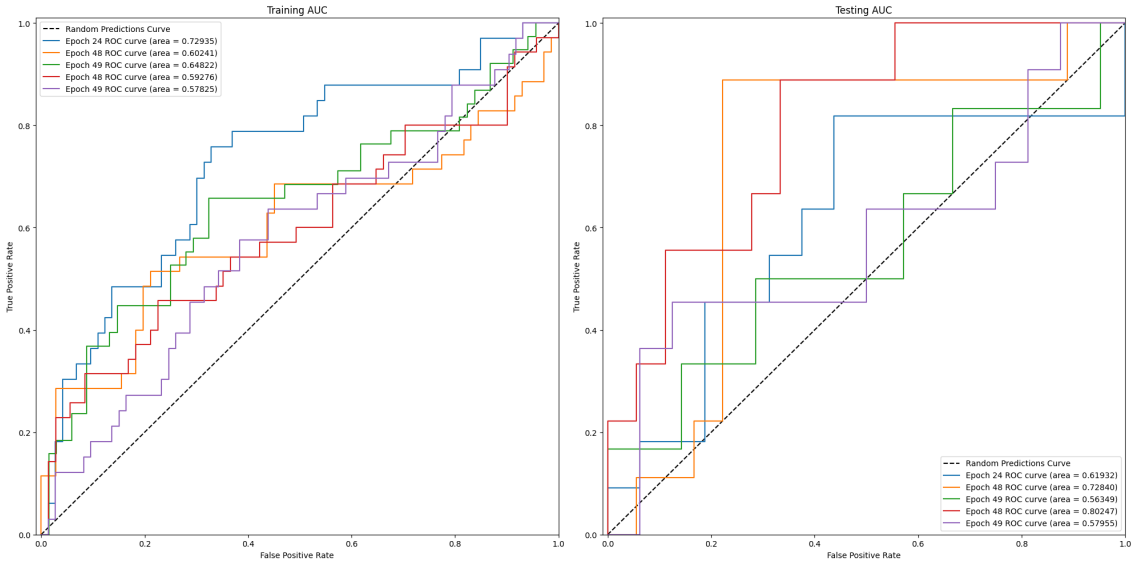


Figure B.32: AUC Curve for each repeated run (β chains)

B.1.5 TCR-BERT

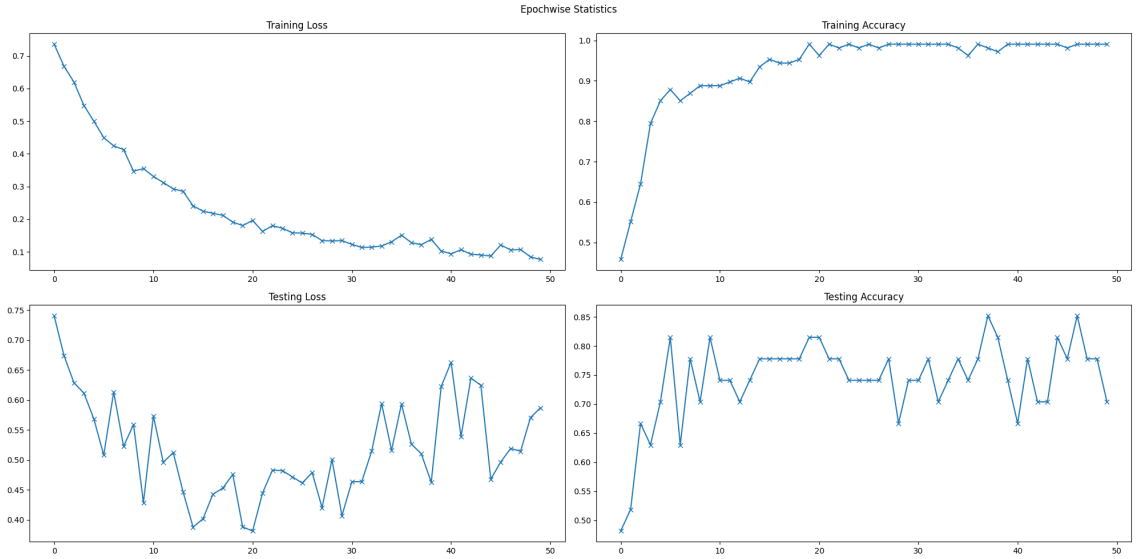


Figure B.33: Loss & Accuracy on Training and Test Sets for each Epoch (α chains)

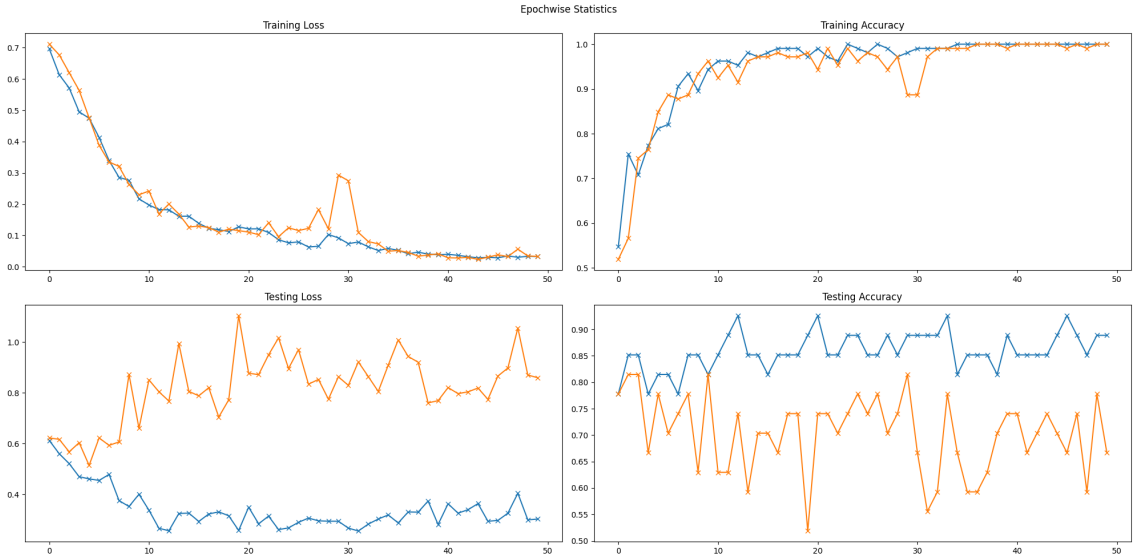


Figure B.34: Loss & Accuracy on Training and Test Sets for each Epoch (β chains)

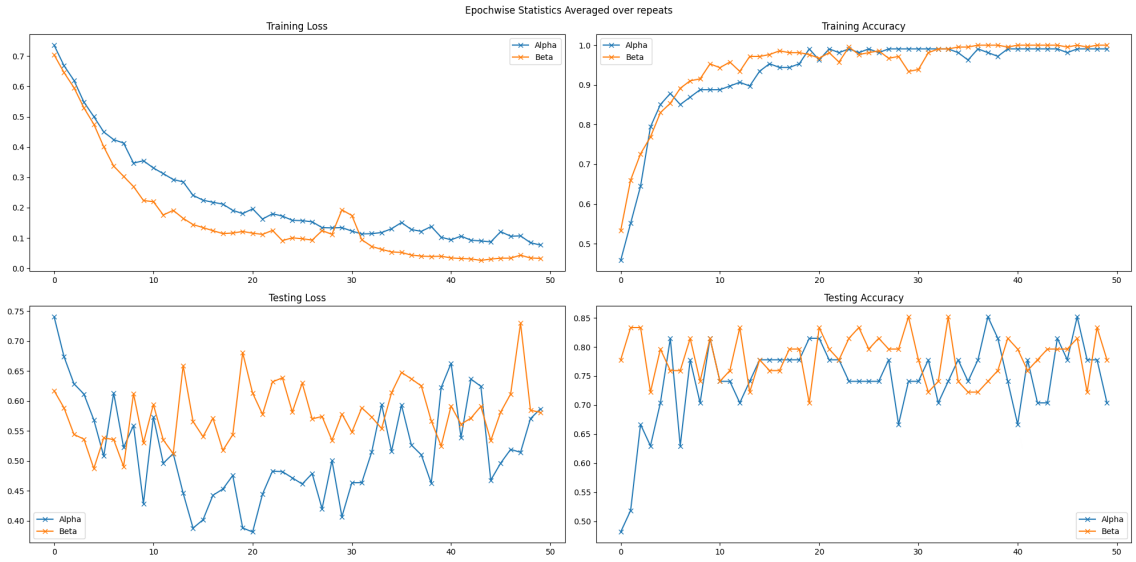


Figure B.35: Loss & Accuracy on Training and Test Sets Averaged on Repeats (α vs β chains)

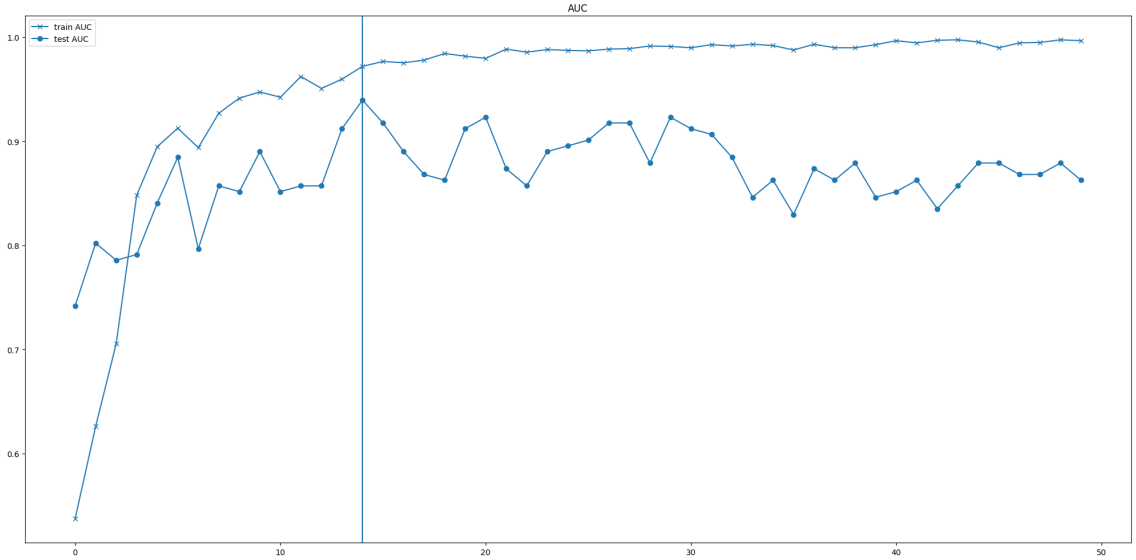


Figure B.36: AUC on Training and Test Sets (α chains)

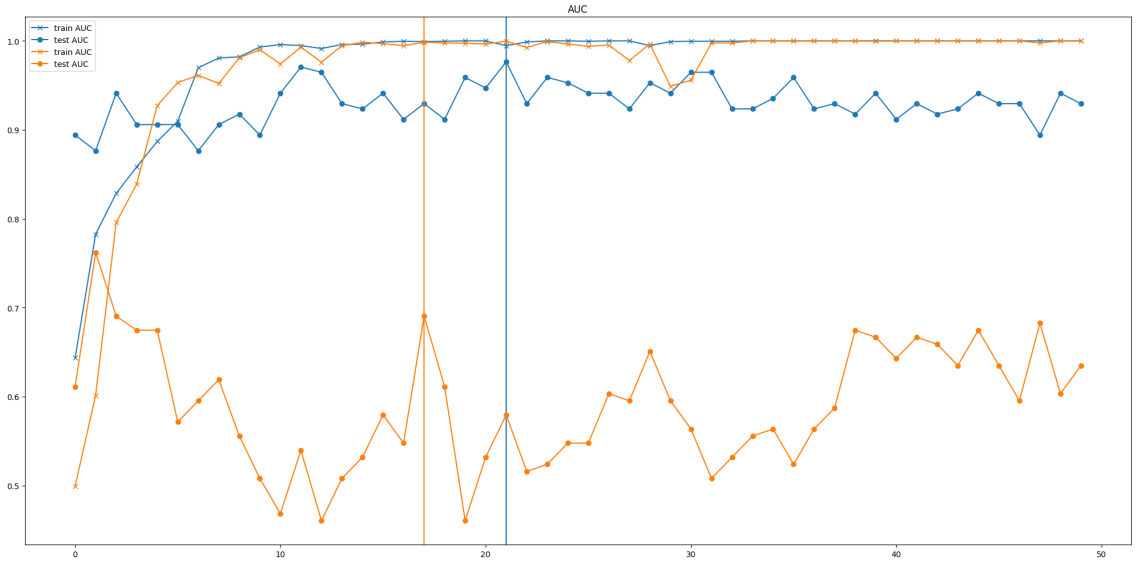


Figure B.37: AUC on Training and Test Sets (β chains)

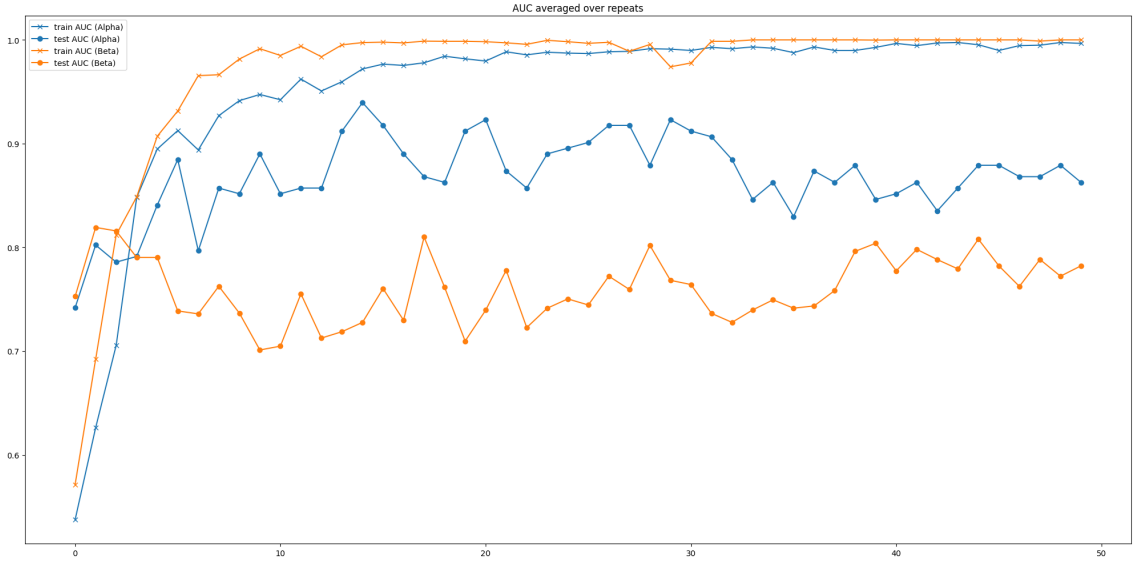


Figure B.38: AUC Averaged on Training and Test Sets (α vs β chains)

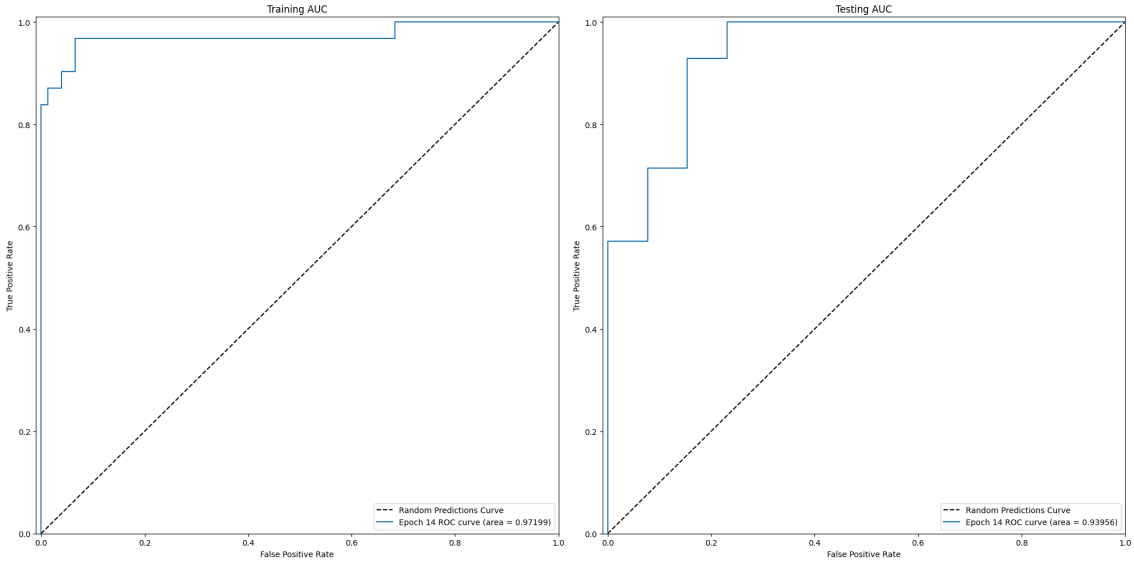


Figure B.39: AUC Curve for each repeated run (α chains)

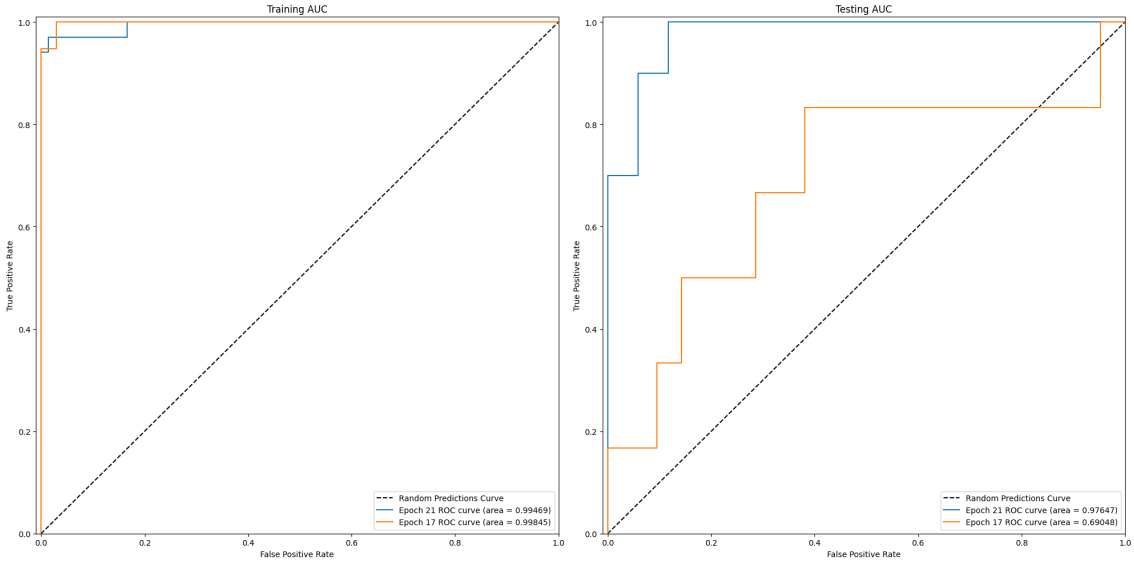


Figure B.40: AUC Curve for each repeated run (β chains)

B.1.6 SCEPTR

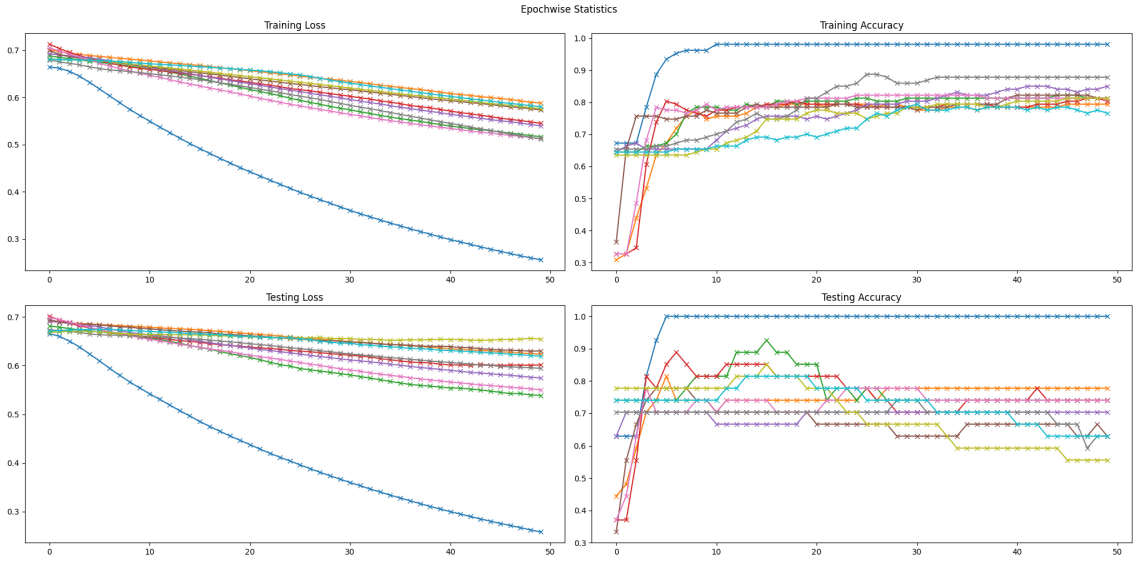


Figure B.41: Loss & Accuracy on Training and Test Sets for each Epoch (α chains)

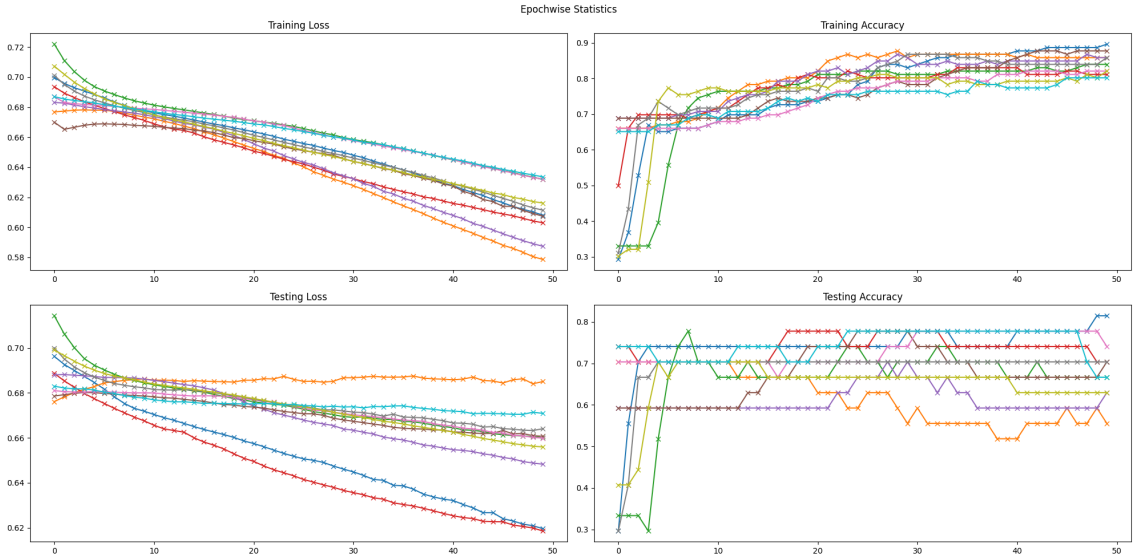


Figure B.42: Loss & Accuracy on Training and Test Sets for each Epoch (β chains)

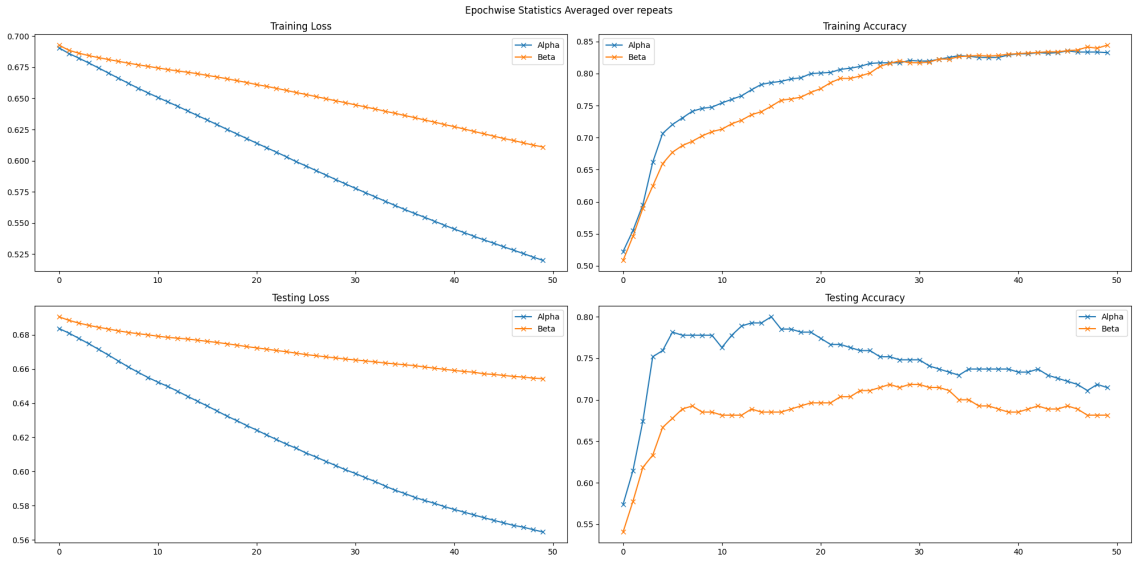


Figure B.43: Loss & Accuracy on Training and Test Sets Averaged on Repeats (α vs β chains)

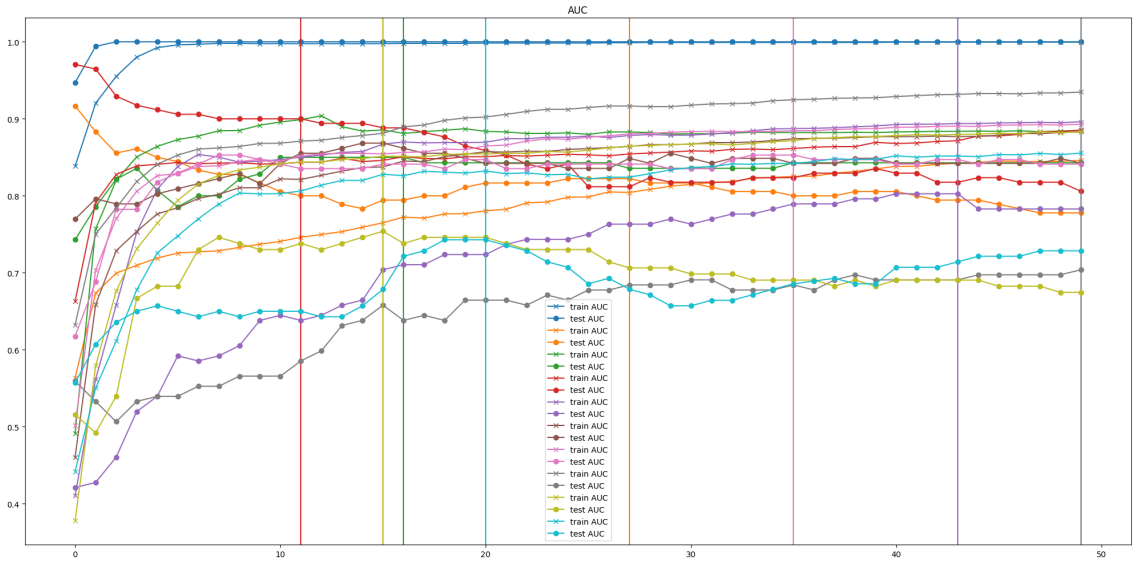


Figure B.44: AUC on Training and Test Sets (α chains)

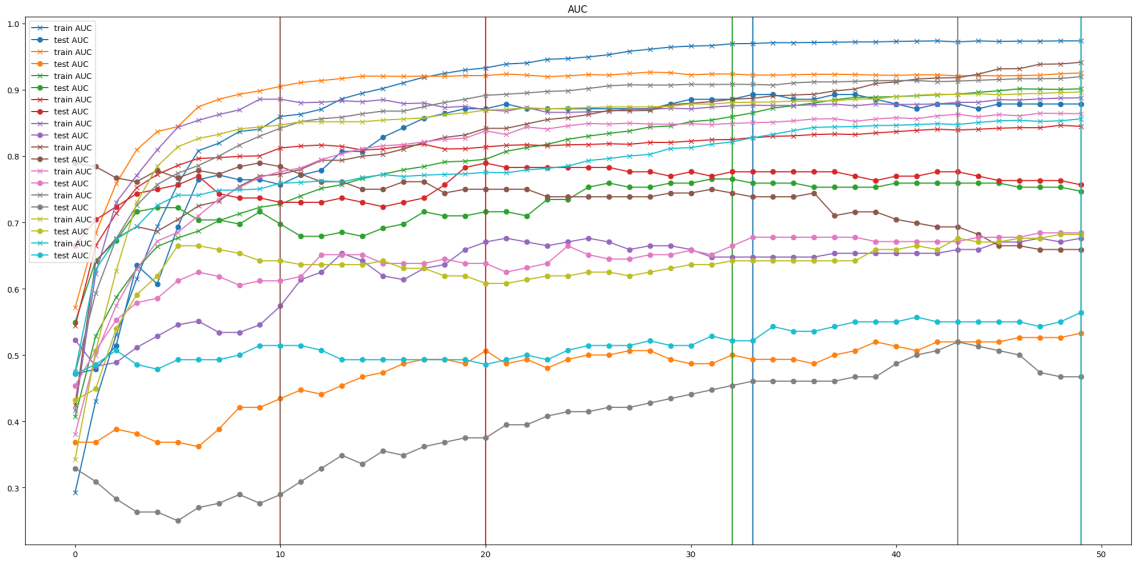


Figure B.45: AUC on Training and Test Sets (β chains)

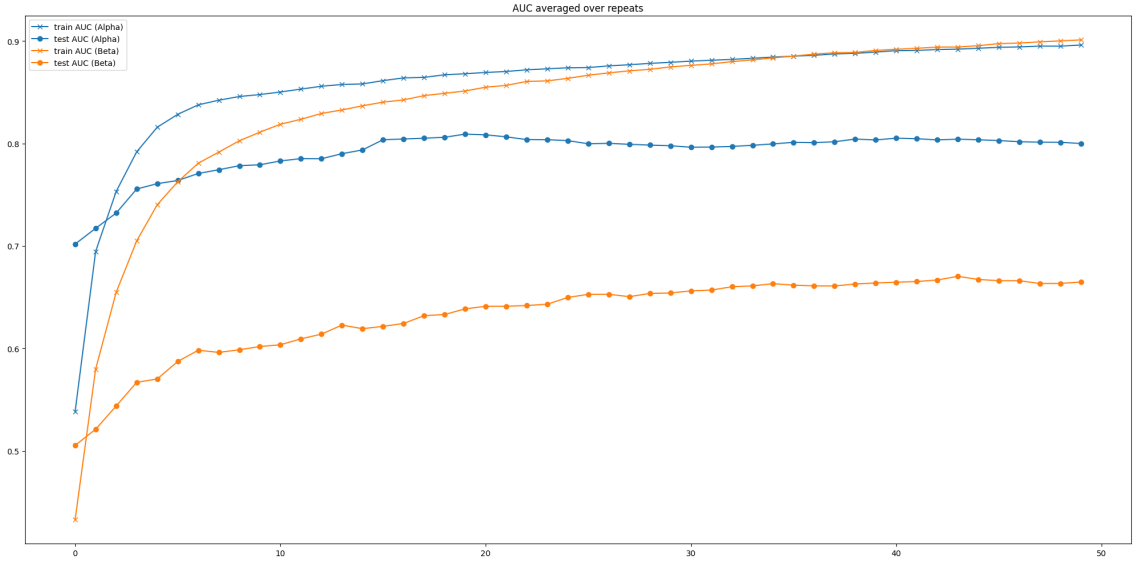


Figure B.46: AUC Averaged on Training and Test Sets (α vs β chains)

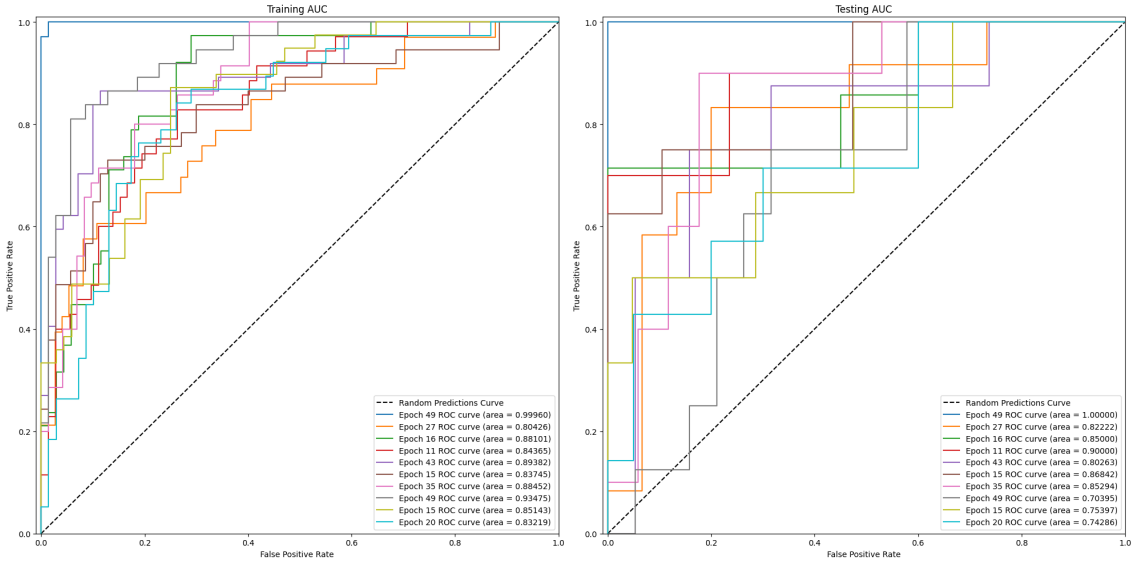


Figure B.47: AUC Curve for each repeated run (α chains)

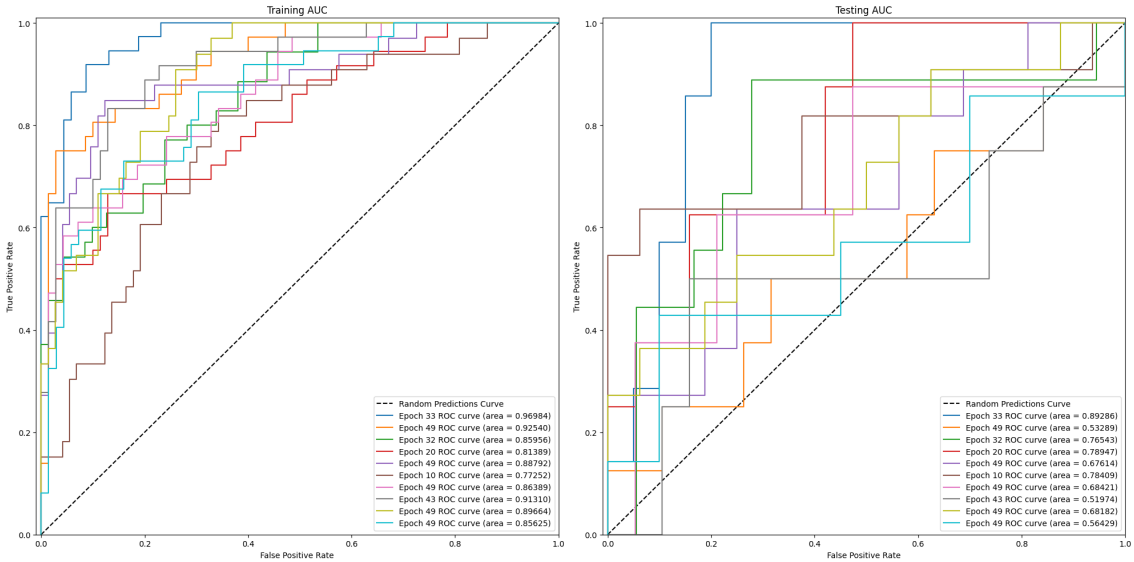


Figure B.48: AUC Curve for each repeated run (β chains)

Appendix C

Project Plan

Project Title

Exploring Large Language Models for T Cell Receptor Sequence Classification for Cancer Prediction

Supervisor

Professor John Shawe-Taylor

Aims and Objectives

The aim of the project is to investigate the use of LLMs as a pretraining tool to create representations of TCR amino sequences that enable the analysis of pathogen specificity and overall immune response, with the ultimate goal of distinguishing between cancerous and non cancerous repertoires. The project will test and compare the effectiveness of different models including off the shelf LLMs such as BERT, and bespoke systems such as SCEPTR, developed at UCL specifically for TCR analysis. It will focus on comparing different models and exploring various types of post processing tasks and algorithms.

Expected Outcomes/Deliverables

The project will deliver a documented codebase including reproduced experiments from the "Multi-Instance Transfer Learning on T cell" thesis, focusing on TCR-BERT and SCEPTR models. The codebase will also include implementations of the chosen approach for improving or extending the thesis work (see **Work Plan – End of January**). The findings will provide clear conclusions and recommendations for advancing TCR sequence analysis and cancer prediction. These deliverables will be supported by the Interim Report in January and Final Report in April.

Work Plan

End of November Complete preliminary research and literature review. Identify the key papers on TCR sequence analysis, machine learning methods, and LLM applications. Get foundational understanding of the TRACERx dataset and preprocessing requirements.

End of December Reproduce the experiments from the "Multi-Instance Transfer Learning on T cell" paper. Confirm SCEPTR's performance metrics. Get a solid understanding of the underlying machine learning algorithms.

End of January Decide on the further approaches for extending or improving the thesis work. These could include e.g. exploring alternative machine learning algorithms for representation or classification, or exploring different preprocessing or post processing methods. Design experiments to test the selected approaches. Submit the Interim Report by the deadline.

End of February Implement, test and compare new approaches against the currently used. Obtain supervisor's approval of the section outline for the report.

End of March Refine and improve the experiments and methodology based on findings. Submit the first draft of the project to supervisor for feedback and revision.

End of April Finalise the project report, incorporating the supervisor's feedback. Submit the completed project by the deadline. Discuss the possibility of publishing the work if it demonstrates significant findings.

Appendix D

Interim Report

Project Title

Exploring Large Language Models for T Cell Receptor Sequence Classification for Cancer Prediction

Supervisor

Professor John Shawe-Taylor

Progress Made to Date

Significant progress has been made in understanding the foundational work upon which this project builds. The "Multi-Instance Transfer Learning on T cell receptor LLMs for Cancer Prediction" master thesis has been thoroughly analysed to understand the problem, methodologies, and areas for further work. Detailed notes have been taken on key concepts and findings, which have been complemented by a literature review. Efforts have also been directed toward reproducing the experiments outlined in the thesis. The original codebase has been successfully set up and executed, providing a solid starting point for the project. The process of creating own codebase with reproduced experiments is currently underway.

Remaining Work

The next steps involve writing custom code and completing the reproduction of the experiments from the master thesis, with a focus on validating the performance of TCR-BERT and SCEPTR. This includes confirming the reported results and documenting any deviations or challenges encountered. Access to the Chain Lab RDS will need to be requested to needs be requested to allow this work to be completed.

Following this, the project will transition to exploring alternative approaches such as different machine learning algorithms for representation and classification, or different preprocessing and post-processing methods. A literature review will be conducted to identify the most promising approaches to implement and test. The findings from these will be analysed and described.

The final stage will involve obtaining feedback from the Supervisor on the experiments and methodology and incorporating this feedback to produce the final version of the report.