Predicting TCR-pMHC interactions using deep learning

Author

Christian Holm Johansen - s202770

Supervisor

Professor Morten Nielsen

Technical University of Denmark

Department of Health Technology

Immunoinformatics and Machine Learning

June 21, 2022

Preface

This is the written piece of work for the thesis *"Predicing TCR pMHC interactions using deep learning"*. The code used in this thesis is available on github at the repository: `https://github.com/chrhjoh/speciale`.

The work has been done at the Immunoinformatics and Machine Learning group at DTU Health Tech from the 31th of January 2022 to the 24th of June 2022 and corresponds to 30 ECTS points. I would like to thank Prof. Morten Nielsen for the invaluable guidance and supervision throughout this project and Alessandro Montemurro for great discussions surrounding the project and the data.

Abstract

The interaction between T-cell receptors (TCRs) and peptide major histocompatibility complexes (pMHC) is essential for the function of the adaptive immune system. Predictions of TCR-pMHC interactions have proven to be a challenge, but accurate predictions have the potential to further vaccine development and cancer treatments.

Here, we will model TCR-pMHC interactions using deep neural networks using a previously created benchmark dataset containing both sequences and predicted energy features. The best performance was obtained using only the peptide and paired complementarity-determining regions (CDR)3 sequences. The remaining sequences and all energy features were unable to provide additional information for models trained on the full dataset.

Furthermore, we will show that attention mechanisms can capture sequence variations between binding and non-binding TCR sequences. Long short-term memory (LSTM) coupled with attention mechanisms also improved performance over convolutional neural networks (CNNs) and LSTMs without attention.

Pan-specific and peptide-specific models showed comparable performances for predicting TCR-pMHC interactions, with pan-specific models performing slightly better when less data was used for training. Pan-specific models were also better at predicting peptide specificity and can be used to predict multiple peptide with the same model. TCR-pMHC binding predictions are feasible on peptides with a substantial number of observations. However, the models' ability to generalize to other peptides is most likely limited by a lack of data and additional TCRs for current and new peptides are needed for a truly pan-specific model.

List of Abbreviations

AUC Area under the curve

AUROC Area under the receiver operator curve

BiLSTM Bidirectional long short-term memory

BLOSUM Block Substitution Matrix

CDR Complementarity-determining region

CNN Convolutional neural network

FFN Feedforward neural network

FN False Negative

FP False Positive

FPR False Positive Rate

GIL GILGFVFTL

GLC GLCTLVAML

HLA human leukocyte antigen

LSTM Long short-term memory

MHC Major histocompatibility complex

NLP Natural Language Processing

NLV NLVPMVATV

RNN Recurrent neural network

ROC Receiver operator curve

Tanh Hyperbolic tangent

TCR T-cell receptor

TAP Transporter Associated With Antigen Processing

TN True Negative

TP True Positive

TPR True Positive Rate

UMI Unique Molecular Identifier

Table of Contents

1 Introduction

T cells are a crucial component of a well-regulated immune system. The large variability in the T-cell receptor (TCR) sequence enables TCRs to recognize a diverse selection of pathogens. This thesis will describe approaches to improve predictions and the understanding of TCR specificity. This will allow for a better understanding of the immunogenicity of antigens and further the research within vaccine development and cancer.

1.1 The Immune System

Here, an introduction to the core aspects of the immune system is given, which serves as this project's foundation. The primary source for this section is *Janeways Immunobiology* [1].

The immune system consists of various mechanisms to recognize and combat pathogens, cancer cells or parasites. The immune system generally consists of the fast and unspecific innate immune system, which recognizes a variety of conserved pathogen-associated molecular patterns [2]. The innate immune system is present from birth and acts as our first line of defence against pathogens that have penetrated the skin or other external barriers. The adaptive immune system generates a slow and specific response to pathogens, which can recognize virtually any antigen a pathogen expresses. The immunity gained from this system is learned through the organism's lifetime, as it accumulates memory of previous infections. The primary cell types involved in the adaptive immune response are lymphocytes. Lymphocytes primarily consist of T-cells and B-cells, expressing immunoglobulin receptors to recognize foreign pathogens. A vital component of the adaptive immune system is the ability to remember signals from past infections and initiate a swift and robust immune response in case of subsequent infections. This ability is gained by having long-lived memory T and B-cells, which can be activated and undergo clonal expansion to quickly generate enough cells to combat the infection [1].

During development, T-cells either become CD4+ helper T-cells or CD8+ effector cells. This project will focus on the CD8+ effector T-cells and their recognition of peptides displayed in Major histocompatibility complex (MHC) class I molecules. These T-cells are responsible for killing infected cells by initiating apoptosis to stop a current infection. While CD8+ T-cells are essential for combating intracellular pathogens, many additional steps are required for a successful immune response, such as the recruitment of CD4+ helper T-cells [3].

1.1.1 MHC class I Displays Antigens to CD8+ T cells

Almost all cells in the human organism contain MHC-I. The MHC-I sit embedded in the cell membrane, allowing cells to show their intracellular contents to CD8+ T-cells.

Proteins inside a cell are continuously monitored for foreign proteins (Figure 1). A protein is fragmented into smaller peptide fragments. These fragments are then moved into the endoplasmic reticulum through the transporter associated with antigen processing (TAP), where they are incorporated into an assembled MHC-I [4]. After the pMHC complex is

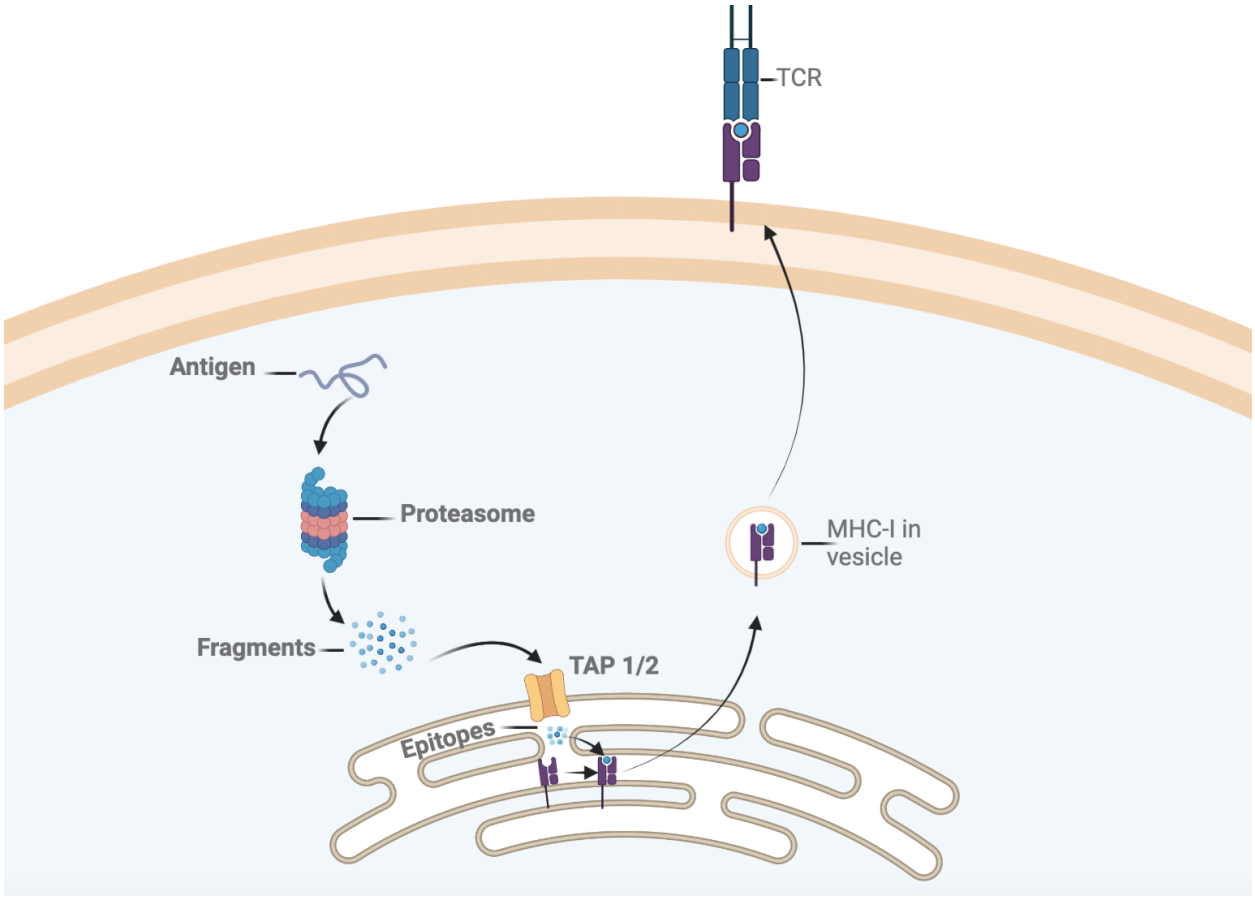


Figure 1: Infected cell displays epitopes to T-cells by incorporating the epitopes into an MHC class I molecule. The newly created pMHC molecule is transported to the surface and displayed to roaming CD8+ TCRs.

formed, the pMHC is transported to the cell surface through a vesicle. From here, the T-cells can assess the presence of non-self peptides bound on the cell surface in complex with MHC-I.

1.1.2 TCRs interacts with Antigen using hypervariable regions

The adaptive immune system needs to recognize a large number of pathogens to protect the body. The molecule responsible for recognizing diverse foreign peptides from pathogens are TCRs. TCRs need to recognize peptides from virtually any pathogen, which poses the issue of generating a molecule with enough diversity to recognize any peptide.

T-cells uses gene rearrangement to generate diverse TCRs [5, 6]. A TCR is a complex of two subunits (TCRα and TCRβ) connected by a disulfide bond. Each subunit consists of a variable and constant domain. The variable domain is responsible for interactions with epitopes displayed on pMHC molecules and is created through gene rearrangement (Figure 2). The TCRα is generated by randomly joining a V and J segment, while the

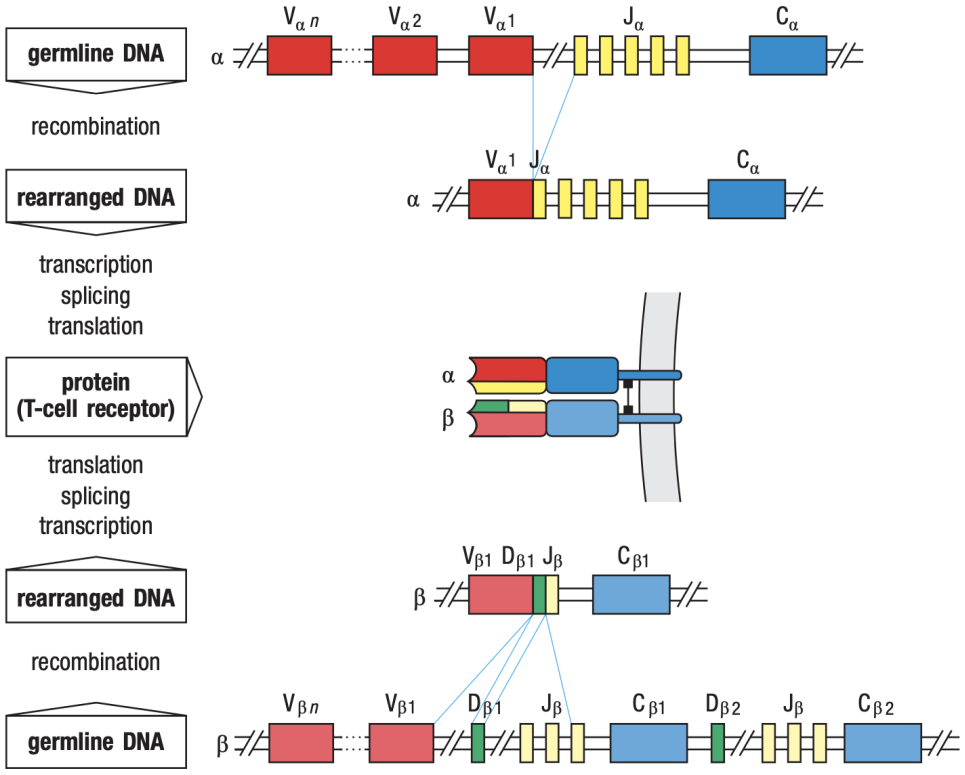


Figure 2: The DNA sequences for TCRs are generated through gene rearrangements of both the α and β locus. A V, (D for the β) and J segment are randomly selected and generate the variable domain of the TCR. Figure adapted from [1].

TCRβ is created by joining a V, D and J segment. Many V, (D), and J gene segments are available at the TCRα and TCRβ loci, and the possible combinations of V, (D) and J gene segments generates a large portion of the T-cells' diversity. Additional diversity is introduced by adding random nucleotides in the junctions between the V, (D) and J segments. These nucleotides result in random amino acids at the junctions between the V, (D) and J gene segments. The fact that TCRαs and TCRβs are generated separately gives additional diversity, as any two TCR subunits could have been generated and combined to a full TCR. The introduced diversity means that it is possible to generate upwards of 10^{18} possible unique TCRs. However, while the number of possible TCRs is quite large, the number of possible epitopes that T-cells need to recognize is much larger than the number of T cells present at any one time. Therefore, while T-cells are rather specific towards certain peptides, they need to be able to recognize multiple peptides [7, 8].

The variable domain of the TCR is made up of an immunoglobulin fold. This fold contains two beta-sheets folded onto each other in a β-sandwich structure [9]. The β-sheets consist of conserved amino acids, while the loops connecting the sheets are less conserved and contain most of the diversity in the TCR. Each of the two TCR subunits contains three of these loops, which are collectively referred to as Complementarity-Determining regions (CDRs).

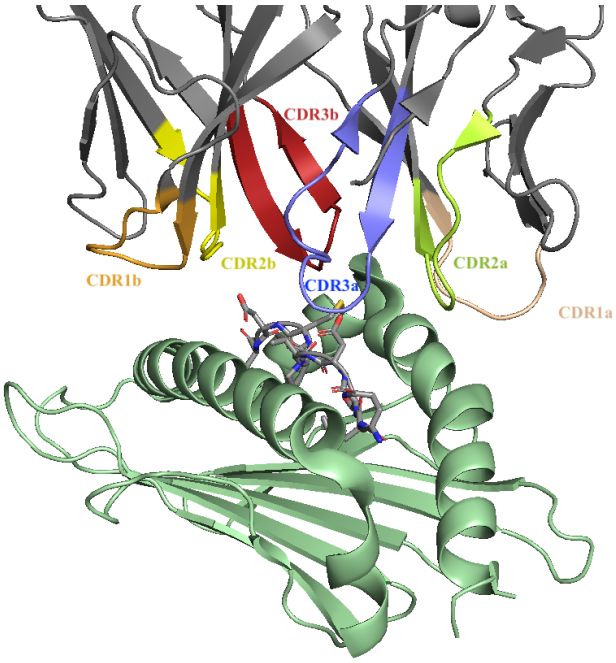


Figure 3: The TCR (top) interact with the pMHC (bottom) complex predominately through the CDR loops at the end of the TCR. Visualization created in PyMOL [12] using PDB structure 3TFK [13]

These CDRs are essential for epitope binding and are the structures closest to the actual peptides (Figure 3). CDR1 and CDR2 are encoded directly in the V gene, while CDR3 consists of both the V, (D) and J segments and nucleotides introduced at the junctions. The CDR3s, therefore, have a higher level of diversity than the other CDRs and are also thought to be the most important for epitope binding [10]. Especially the CDR3b has been widely investigated, as the D gene segment allows for much diversity at the CDR3b [11]. Furthermore, the CDR3s are generally in closer contact with the peptide than CDR1 and CDR2, which also interacts with the MHC molecule (Figure 3).

1.2 Machine Learning Methods

Predicting TCR-pMHC interaction is a supervised classification problem. In these types of problems, a model is fitted to predict the most likely class for a given observation.

$$y = f_\theta(\mathbf{X})$$

The model f_θ is parameterized by weights θ and takes the input $\mathbf{X}$ and attempts to predict the output class y. Predicting TCR-pMHC binding is a binary classification meaning $y = 1$ refers to binding while $y = 0$ refers to non-binding.

Deep learning is a sub-field of machine learning utilizing artificial neural networks for statistical modelling. These types of networks mimic the brain in the way neurons signal and

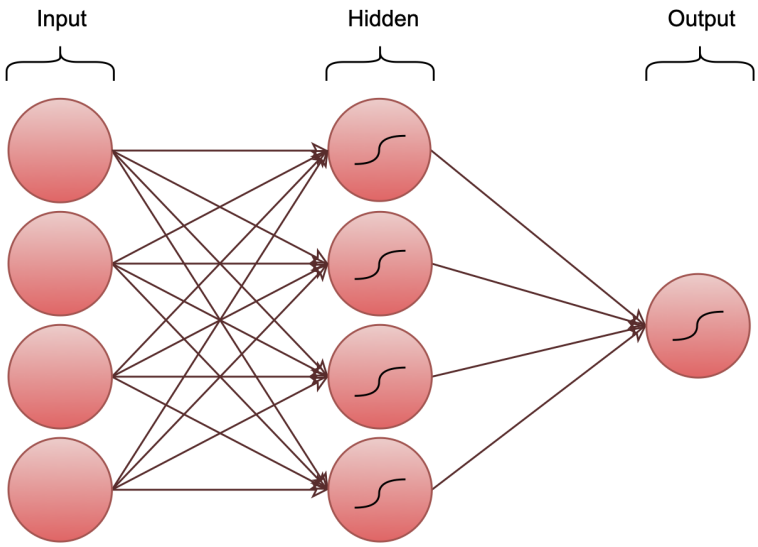


Figure 4: A feedforward network consisting of an input, hidden and output layer. Each circle represents a neuron, and the arrows represent the weights applied when propagating the signal from the previous layer. A non-linear activation function is applied on hidden and output neurons represented by the sigmoid curve.

activate each other in specific patterns to generate a specific output. The simplest type of artificial neural network is the feedforward neural network (FFN). Figure 4 depicts a FFN with one input, hidden and output layer, each containing several neurons. Creating predictions with a neural network is called a forward pass. Here, the output of a layer is calculated from the previous layer, which is repeated for all layers. The output of a neuron is calculated from the previous layer by taking a weighted average of the outputs of the previous layers. This average is then passed through a non-linearity to generate the output of that given neuron.

The learnable parameters in a FFN are the weights used to generate the weighted averages. When the model has generated predictions through the forward pass, these weights can be updated through a method called backpropagation. A loss function such as cross entropy or mean square error is used to calculate the loss of the model. The loss gradient is calculated for each weight by backpropagating the gradient through all hidden layers. The weights are then updated using an optimization method such as gradient descent, thereby lowering the training loss. The cycle of predicting observations in the forward pass and propagating the loss gradient backwards is repeated, and weights are updated each iteration until a specified number of epochs has been reached. The extensions of the FFN introduced in the next sections are more advanced in architecture, but all rely on backpropagation to update their weights.

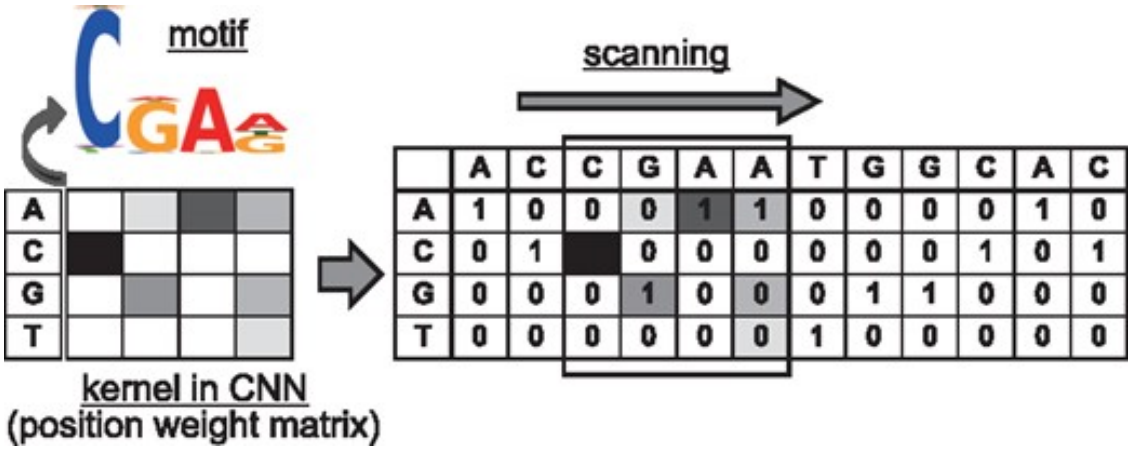


Figure 5: A kernel/filter scans the entire sequence looking for a similar motif. The filter finds a complete match (marked by a box). The subsequent max pooling operation will then select this match. Figure adapted from [16]

1.2.1 Convolutional Neural Networks

FFNs cannot handle sequential data gracefully, as they are highly dependent on data having the same input size. Convolutional Neural Networks (CNNs) handles this by finding local motifs in the input independently of their position. CNNs were initially developed for image analysis to identify certain pixel arrangements, which can then be combined into more complex features resembling e.g. facial features being combined into a face [14]. CNNs convolute an image by scanning with a filter/kernel with specific weights. These filters obtain a high value in areas where the input matches the weights, generating a feature map specific for that filter. After convolution, the output is transformed using a non-linear transformation such as hyperbolic tangent (Tanh) function. The feature map's dimensionality can be reduced by max pooling. This operation pools the image by selecting the largest positive value within an area to reduce the image's dimensions and regularize the network.

CNNs have also proven helpful for finding motifs in biological sequences [15, 16]. Each filter represents a position weight matrix with a specific motif depending on the weights (Figure 5). The filter scans over the sequence and yields a high output if the sequence resembles the motif stored in the filter (Figure 5). Multiple convolutional and max pool layers can be used to generate a more complex convolution of the sequence. However, for this project, we only investigate shallow one-layer convolutions with a subsequent global max pool, as this type of network has shown promise for modelling TCR-pMHC interactions [17]. The resulting feature vector is made by having multiple filters scan the sequence and represents whether the motifs encoded by the filters were present in the sequence. Downstream modelling then decides whether the presence of these motifs result in an interaction or not.

1.2.2 Long Short Term Memory

Recurrent neural networks (RNN) can also model sequential data. These networks propagate the signal through the sequence by using the previous state in the calculations. The simplest RNN can be challenging to train as the loss gradient starts to vanish when the hidden states are propagated through the sequence. When the gradient vanishes, it becomes hard for the network to update weights, as a gradient is required for updating the weights. Long short-term memory (LSTM) was introduced to solve this issue by allowing the signal to bypass most of the matrix multiplication and thereby alleviate the problem with vanishing gradients [18].

An LSTM consists of an LSTM cell for each input in the sequence (Figure 6). The input for a given cell is the input at that position (x_i), the previous cell state (c_{i-1}), and the previous hidden state (h_{i-1}). The cell state represents the long-term memory of the LSTM used to avoid the vanishing gradient, whereas the hidden state is the working memory and also serves as the model's output. Four gates exist in the LSTM, each consisting of a dense layer and an activation function (yellow blocks in Figure 6). The forget gate (f_t) calculates how much should be forgotten of the current cell state (c) by a sigmoid layer, followed by elementwise multiplication. The input gate (i) determines how much new information should be integrated into the cell state, while the next gate (č) determines what should be integrated. Lastly, the output gate (o) calculates the new hidden state, fed to the next cell along with the updated cell state.

There exists two common methods for using LSTMs depending on the problem at hand. Using the hidden state for each cell can often be beneficial in a many-to-many problem such as amino acid structure prediction, as we want information about each individual residue. At the same time, many-to-one problems, such as predicting TCR-pMHC interactions, often uses the last hidden state in the sequence. At this state, information from all previous positions has been incorporated into the model, allowing the model to condense information about all positions into one hidden state.

1.2.3 Attention Mechanism

While the last hidden state in LSTMs incorporates information from all states in the sequence, information may still be lost, especially if the first positions of the sequence are important. Instead, attention can be used to focus on important areas of the sequence while excluding unimportant areas. By using attention, the hidden state for each position can be incorporated into an overall hidden state, which means all states can be used instead of only using the final hidden state even for a many-to-one problems such as TCR-pMHC binding.

Attention mechanisms were first developed for machine translation, where words placed in certain areas would contribute more to the prediction of words in the translated text [19]. Another widespread use of attention is transformers, which uses self-attention to understand languages [20, 21].

Within biological sequence analysis, attention has been used for predictions of subcellular

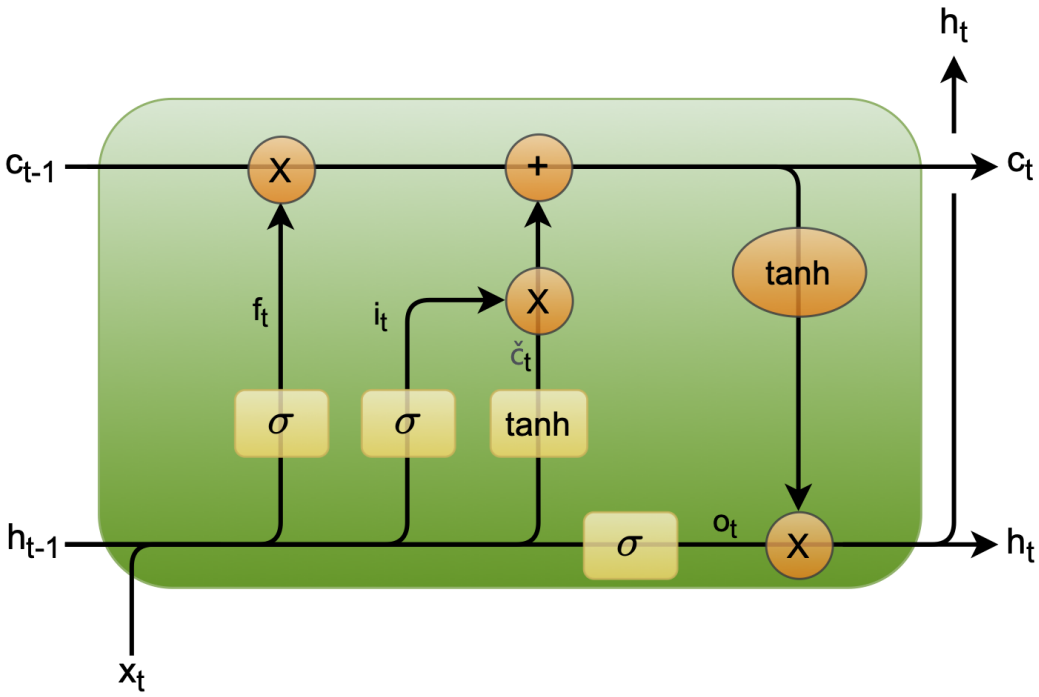


Figure 6: Visualization of operations and layers present in an LSTM cell. The previous hidden state and the new input is concatenated. Each gate determines what should be forgotten (f), how much should be added (i) and what should be added (č) to the cell state. The last gate determines how to combine the new input, hidden state and cell state in the output.

location of proteins [22], where the signal peptide stores hold most of the information about its subcellular location. Similarly, the CDRs contain most of the information from TCRs, which can be selected using attention.

In order to calculate the attention on a position, all hidden states from the LSTM output are used ($h_1, h_2...h_L$), where L represents the sequence length. The attention function (Equation 1) is used to calculate the similarity of each hidden state to a high-level query vector ($\mathbf{q}$) and then scaled using the softmax function (Equation 2). Here the query vector ($\mathbf{q}$) and the weight matrix ($\boldsymbol{W_k}$) are learnable parameters learned during training.

$$e_l = Tanh(\boldsymbol{h_l W_k})\boldsymbol{q}^T \tag{1}$$

$$a_l = \frac{\exp e_l}{\sum_{i=1}^{L} \exp e_i} \tag{2}$$

Lastly, a hidden state can be calculated with the context of all hidden states.

$$\boldsymbol{h_c} = \sum_{l=1}^{L} a_l \boldsymbol{h_l} \tag{3}$$

A schematic version of these calculations, along with the dimensions of the vectors, can be seen in Figure 7. The attentions for each position can be calculated simultaneously using matrix multiplication to speed up the computations.

The query vector represents how an important hidden state would look. If a hidden state has a vector representation similar to the query vector, their dot product will be large, resulting in a considerable attention value for that position. An average of the hidden states weighted by importance can be calculated by first calculating the attention put on each position and then incorporating each hidden state according to this value.

1.3 TCR-pMHC binding studies can improve vaccine technologies

The field of immunoinformatics investigates immunogenicity in silico. Knowledge of peptide immunogenicity is essential for designing better vaccines, as peptide-based vaccines allow for better targeting of immunodominant peptides. Also, peptide vaccines are easier to produce because of the simplified construct and could potentially be tailored for specific populations [23]. In contrast, current vaccines often use entire antigens, which may result in a less effective immune response than a peptide-based vaccine with carefully selected peptides [24].

Peptide-based vaccines both have prospects within cancer therapy and treating various infectious diseases by enhancing the organism's immune system [25]. Knowing the rules for TCR binding will therefore provide a substantial step toward more personalized treatments for vaccinations and cancer treatments.

Predictions tools for immunogenicity by predicting pan-specific pMHC binding, such as NetMHCpan [26, 27], have existed for a while. However, the immunogenicity of a peptide

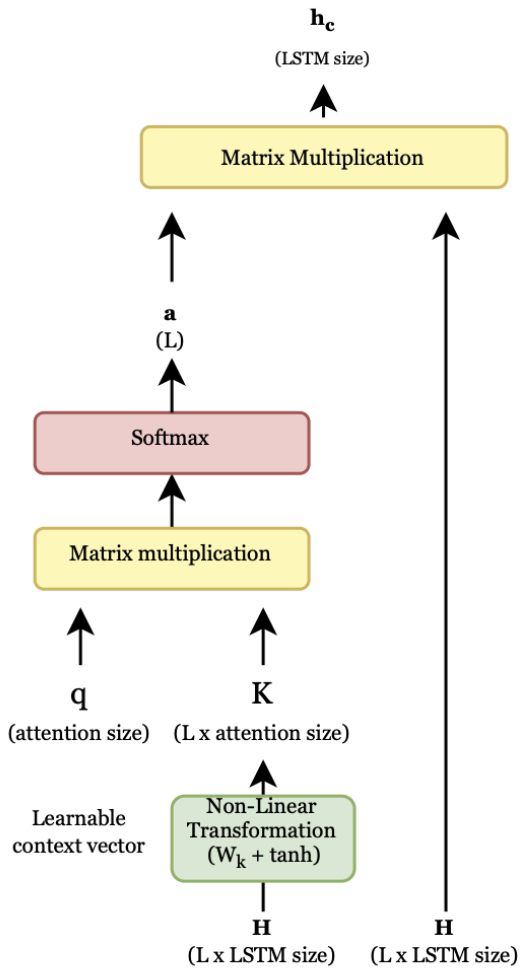


Figure 7: Attention mechanism used for this project. Q (query) is a vector of learnable parameters. K (Key) is created by a linear layer and the Tanh activation function. The matrix-vector product between q and K is scaled using a softmax, and the resulting attention values are then used to create a weighted average using matrix multiplication. Dimensions of matrices and vectors are shown in parenthesis.

also depends on additional factors, including TCR binding and stimulation by cytokines [1]. Multiple tools have also been developed for predicting TCR-pMHC predictions previously [17, 28, 29, 30, 31]. These tools use various methodologies from alignment [28], to CNN based approaches [30, 17] and NLP based models using LSTMs [29] or transformers [31] to predict TCR peptide interactions.

In the thesis by Meitil [32] additional force field energy terms were estimated. These energies provide structural information about the TCR-pMHC complex and could be used to increase model performance. Furthermore, a benchmark dataset was also created, serving as the primary data source for this project.

1.4 Regularization

When training machine learning models, it is crucial to have a model that can generalize to unseen data. Regularization techniques exist to avoid having the model overfit on the training set. As overfitting occurs, the model generalizes less as it learns the noise within the training set. One regularization technique is early stopping, where training is terminated after a specified number of epochs without improvement on a validation set [15]. As the model is training, it moves down the loss gradient, which improves the training loss. If the model learns noise specific to the training data, the model will continue to improve on the training data but not on the validation data. Therefore, stopping the model when this occurs avoids learning excessive amounts of noise from the training data.

Another regularization approach is dropout [33]. Neurons are selected with a random probability and have their value set to 0, effectively removing that neuron from the parameters for one iteration. Therefore, the model cannot rely on a specific set of neurons during training, meaning the model has to find other combinations of inputs to predict binding. By keeping the network in check and forcing it to learn multiple ways of predicting the label, the network is regularized by preventing overfitting to specific inputs.

The easiest way to generate an independent test set is to never use a specific portion for training and only evaluate upon this data. However, if the amount of data is limited, there could be a risk of the test sample being too small and not able to fully represent the variability of the data. Data should not be present in both the training and testing for a proper estimate of the generalization error. Nested cross-validation can use all data for both training and testing (Figure 8) and still manage to generate an unbiased estimate of the generalization error. Nested cross-validation works by first dividing data into several partitions. Then two nested loops go over the partitions, where a test partition is selected in the outer loop, while the inner loop selects a validation set for early stopping. A model is trained on the three remaining partitions and used to predict the test set. A new validation set is then chosen, and a new model is trained on the new training partitions. For 5-fold nested cross-validation, models are trained for all combinations of test and validation sets resulting in 20 models and each observation having four unbiased predictions. The final prediction is calculated as the average of the four predictions. Cross-validation can thereby avoid the issues of evaluating performance on data that have been used for training the model but still use all data for

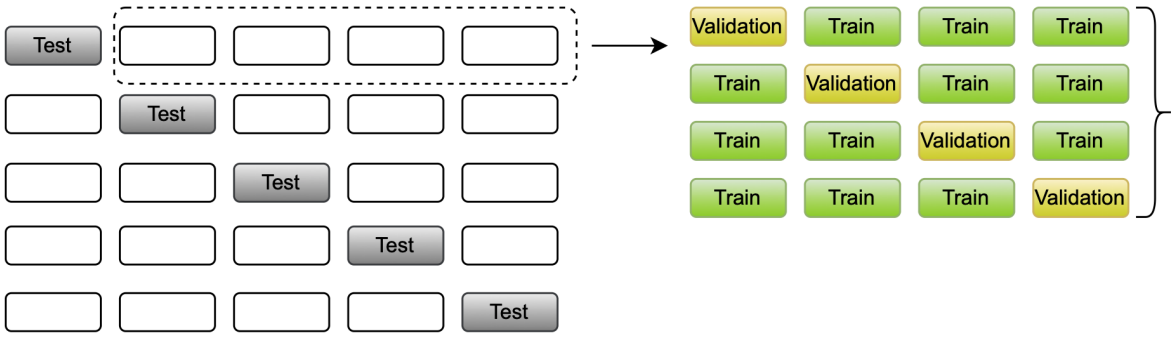


Figure 8: Nested cross-validation to generate test predictions for the entire dataset. The outer cross-validation loop selects a partition for testing, and the inner loop trains for models for each test set, each with a different validation set. After training, 20 models have been trained, and each observation has four predictions. These predictions can then be averaged to generate final predictions.

training models.

1.5 Methods for Residue Specific Embeddings

For machine learning, a protein sequence needs to be represented by numerical values. The simplest way to achieve this is by one-hot encoding the sequence. However, representing amino acids as orthogonal vectors does not fully represent the relationship between amino acids. Instead, the sequence can be represented by a Block Substitution Matrix (BLOSUM) encoding [34]. This encoding captures how some amino acids are closely related and are found more frequently as mutation pairs in nature. When encoding using BLOSUM, amino acids often substituted in nature will be closely related in the encoding space and allow the model to understand how some amino acids have similar functions within proteins.

BLOSUM matrices are generated from homologous proteins and give generic protein-specific encodings representing evolutionary selection. However, CDRs are made by stochastic re-arrangement during T-cell development. The BLOSUM encoding may therefore be inappropriate to represent TCR sequences. Within Natural Language Processing (NLP), word embedding methods such as Word2Vec models [35] are used to learn word embeddings from the context of surrounding words.

These models can be trained using the skip-gram method (Figure 9). Here words are used to predict the surrounding words. A selected word is first mapped to a hidden representation (projection) using a single dense layer. This hidden representation is then used to predict surrounding layers using a softmax layer with the vocabulary size as the output dimension. Each ord projection is unique, but the weights used to generate the outputs are shared. Therefore, if words are often found in similar contexts (e.g. king and queen), then the model will give these words similar projections as they are used to predict the same words during

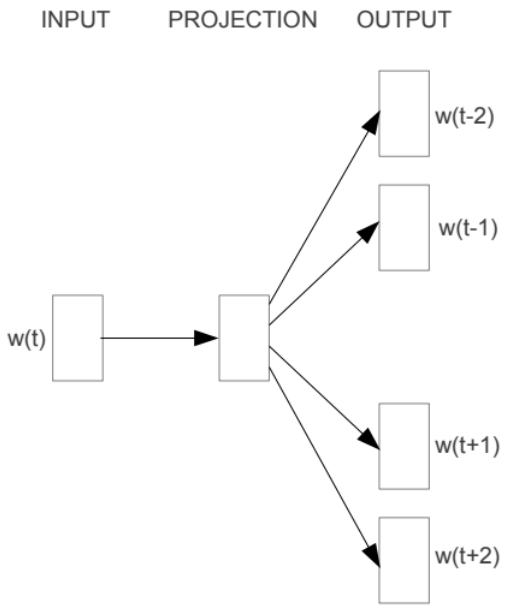


Figure 9: The skip-gram algorithm for training Word2Vec models. A word is selected from the input and projected using a linear layer. This projection is then used to predict output words [35].

training. These projections can then be used in a neural network to embed each word.

When used on proteins, amino acids used in similar contexts would obtain similar embeddings. The Word2Vec model makes it possible to train embeddings specific to CDRs by only including CDR sequences in training. These embeddings may be able to change the input space to be more favourable for predictions of TCR-pMHC interactions.

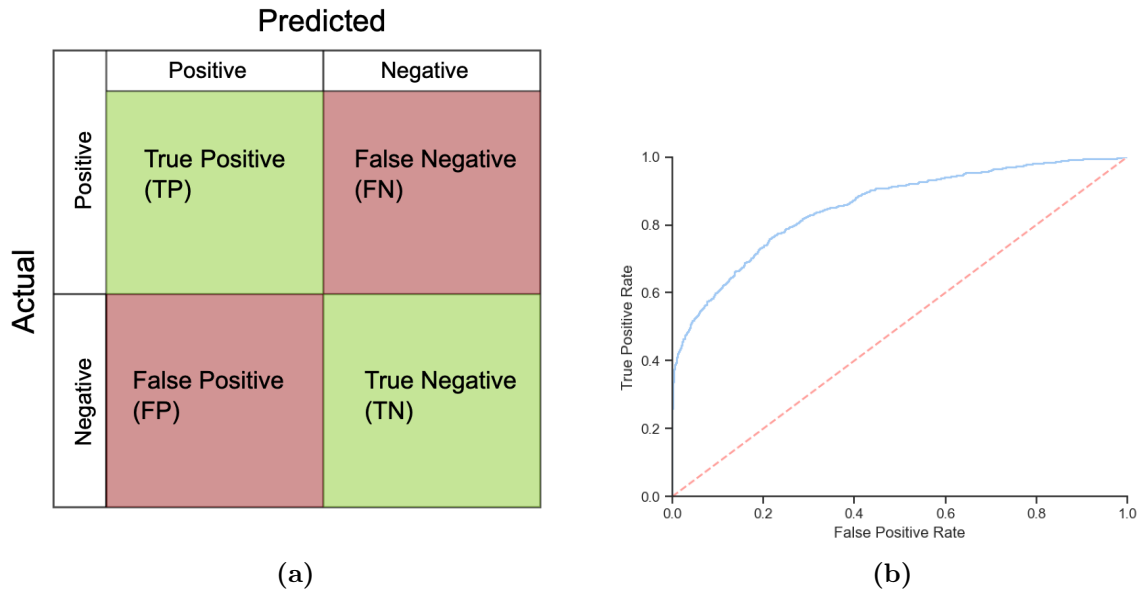
1.6 Performance measures

The performance of a model can be measured by many different measures. In binary classification, a confusion matrix is an excellent way of visualising the errors a model makes (Figure 10A). The confusion matrix is based on the True Positives (TP), False Positives (FP), False Negatives (FN) and True Negatives (TN) from comparing predictions with their actual class. These classifications can be used in various metrics to evaluate performance.

The most straightforward quantitative performance metric is accuracy, which measures the fraction of correctly predicted observations.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{4}$$

However, accuracy is ill-suited for unbalanced data, as an accuracy above random (0.5) is easily achievable by only predicting the dominant class. Instead, the area under the receiver operator characteristic curve (AUROC) or simply the area under the curve (AUC) is used.



(a) **(b)**

Figure 10: **(A)** Confusion matrix for a binary classifier. **(B)** Receiver Operator Characteristic (ROC) curve for a binary classifier. Random performance is shown by the dashed red line (AUC = 0.5).

The receiver operator characteristic (ROC) curve consists of the true positive rate (TPR) and false positive rate (FPR).

$$TPR = \frac{TP}{FP + TP} \tag{5}$$

$$FPR = \frac{FP}{FP + TP} \tag{6}$$

The ROC curve is calculated by varying the threshold required for classifying a positive in the interval $\theta \in (0, 1)$ and for each possible threshold, calculate the TPR and FPR (Figure 10B). Firstly the threshold is so high that all observations are assigned as negatives. As the threshold is lowered, the TPR and FPR increase as more observations are classified as positives until the ROC curve reaches (1, 1), and all observations are classified as positive. The ROC curve will be roughly linear at random performance and give an AUC of 0.5. If the model manages to separate the positive and negatives perfectly, there will be no overlap in the positive and negative score distributions. Therefore, when varying the threshold, the TPR will become one before the FPR changes from 0, yielding an AUC of 1 at perfect performance. For this project, we will primarily use the AUC as it can handle unbalanced data more gracefully than the accuracy metric while still being easy to interpret.

1.7 Project scope

The primary goal of this project is to train a classifier to predict TCR-pMHC binding using deep learning and assess its performance. This thesis springs from work done by Meitil

[32], which had a large focus on creating force field energy terms to assist in predicting TCR-pMHC binding.

In this thesis, the focus will be primarily on using the dataset created in Meitil's thesis. Here we will be evaluating the importance of the sequences and features in the available dataset to see if any sequences or features are irrelevant for TCR-pMHC predictions using a simplified version of the CNN from Montemurro et al. [17]. Furthermore, we will introduce a network architecture comprising of Seq-to-LSTM-to-Attention-to-Dense. The model behaviour will be investigated by visualizing the importance of specific positions in the CNN and attention-based LSTM.

Additionally, the effect of adding attention will be investigated by comparing its performance to models similar to the ones described in Section 1.3. Lastly, the performance gained from having a pan-specific model will be explored by comparing it to two kinds of peptide-specific models.

In this thesis, we want to answer the following questions:

- Which sequences and features from the dataset developed by Meitil carry the most information?
- Can attention mechanisms be used to distinguish positive and negative CDR3 sequences?
- Is the attention mechanism able to preserve performance as data redundancy is reduced?
- Can pan-specific models outperform peptide-specific models?

2 Materials And Methods

2.1 Creation Of Data From Previous Projects

This section describes the creation of the dataset and the generation of the force field energies used as the starting point for this thesis. This was initially done in the NetTCR-2.0 paper and Meitil's master thesis [17, 32].

2.1.1 Initial Creation Of Dataset

The data originates from paired CDR3α and CDR3β datasets used by Montemurro et al. The data was initially created as described below [17].

Three sources of data were used to create this dataset.

Positive data was retrieved from VDJdb [36] on the 5th of August 2020, and IEDB [37] on the 26th of August 2020. The negative data was extracted from the 10x dataset [38]. The IEDB and VDJdb contains curated data obtained from published literature, while the 10x dataset was created from single-cell immune profiling of four different healthy humans. The positive TCRs were restricted to only include TCRs with CDR3 lengths between 8-18 that bind peptides with length 9 displayed on HLA-A02*01.

The negative data was similarly restricted to TCRs with length 8-18 and peptides with length 9. The negatives were identified as TCRs with an Unique Molecular Identifier (UMI) count of 0 for all peptides in the dataset. The UMI count for a given TCR-pMHC data point specifies the number of TCRs with that sequence was bound to the pMHC complex with that UMI.

Redundancy reduction was performed to reduce information leakage between partitions and reduce the amount of redundancy present in the dataset. The positive and negative data were reduced separately using the Hobohm 1 algorithm [39] with 95% similarity cutoff using the Levenshtein similarity metric [17]. The average similarity of the CDR3α and CDR3β was used as the metric for similarity between TCRs. This ensured that no two data points were more than 95% similar within their class. The positives were then randomly partitioned into 5 partitions, and the negatives were added randomly in a 3:1 ratio to the 5 partitions.

2.1.2 Adding Swapped Negatives

Swapped negatives were added to force the model to learn what specific peptides the TCR binds instead of only identifying what makes a positive TCR. The swapped negatives were generated by taking a positive TCR and selecting a different random peptide from the same partition. The swapped negatives were kept in the same partitions as the positive TCR to avoid information leakage.

2.1.3 Retrieval Of V Gene Sequence And Force Field Modelling

Recreating the entire TCR sequence and the force field modelling was done by Meitil in her thesis [32]. The negatives from the 10x dataset already had an annotated V gene, which could be used to recreate the missing sequence. For the positive CDR3s, they were mapped to the VDJdb to retrieve the V gene.

The hidden markov model from LYRA [40] was used to align the V gene sequence with the CDR3 to form the whole TCR sequence.

TCRpMHCmodels was used to estimate a TCR-pMHC structure used for the energy calculations [41]. The resulting PDB files were used to create force field simulations of interaction energies using Rosetta and FoldX [42, 43]. Rosetta generated both interaction energies for each residue in the sequence and global interaction terms between the proteins. FoldX was used to add additional global interaction terms between the proteins. See the thesis by Meitil for specific commands used to generate interaction energies [32].

TCRpMHCmodels succeeded in generating a structure for 82.9% of the positive complexes and 97% of the negative [32]. The total amount of data points ends up at 6897 (1721 positives, 1721 swapped and 3455 negatives).

2.2 Creation Of Comparable CDR Dataset

To investigate the relevance of the energy features on only the CDR3α and CDR3β, the per residue energy features simulated using Rosetta would have to be restricted to only these sequence features. The sequences created with LYRA had not been annotated after construction. Therefore, IgBLAST [44] was used to annotate the full TCRs, and afterwards, the per residue energy features were restricted to the CDR3 positions.

2.3 Constructing Input For Networks

The amino acid sequence had to be encoded or embedded in a representation useable by the network. Encoding of the data was done using a BLOSUM50 matrix scaled by a factor of $\frac{1}{5}$. A vector of length 20 is created for each residue, yielding an l x 20 matrix for a sequence with length l. The other approach creates embeddings using a Word2Vec model for the CDR3 sequences and is explained in Section 2.6. Not all TCR sequences have the same length. Therefore, the sequences were padded on the right with zeros such that all TCRα/CDR3α and TCRβ/CDR3β had the same length. The sequence encoding was combined with the energy features generated by Rosetta and FoldX.

The dimensions of the final tensor used for training can be seen in Figure 11. Each of the 6897 TCRpMHC complexes is a matrix with the padded sequences as the first dimension and the encoding and energy features as the second dimension.

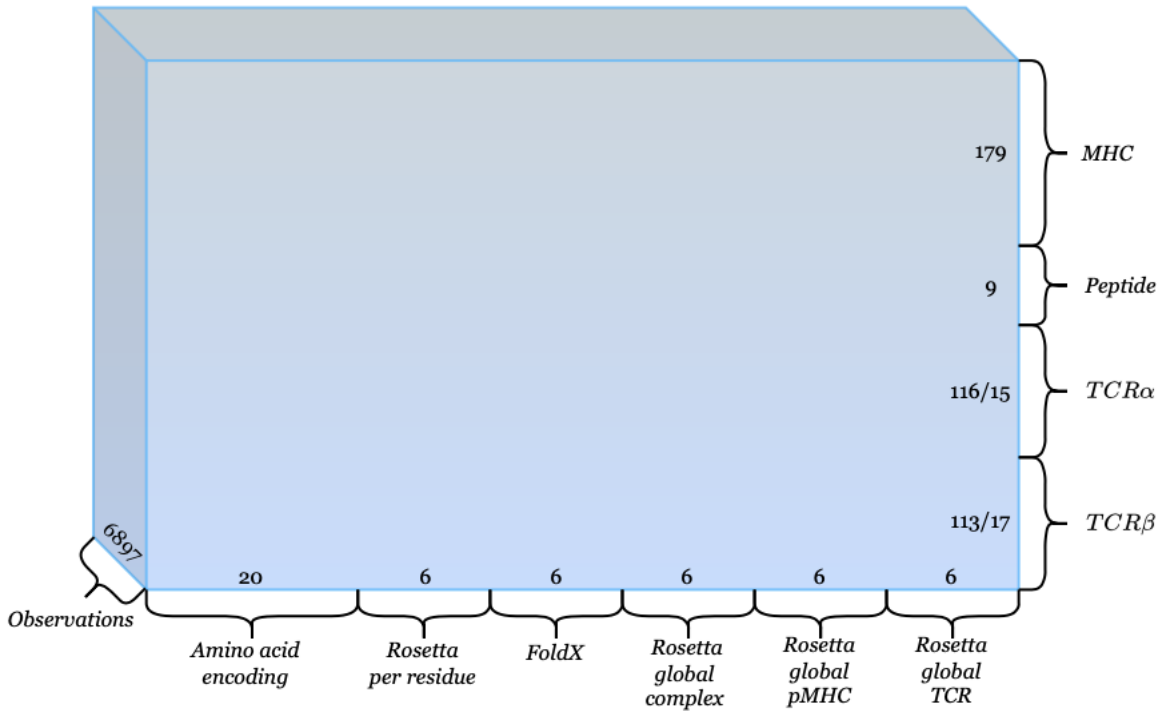


Figure 11: Dimensions of dataset used throughout the project. TCRα and TCRβ shows dimensions for full TCR and only CDR3 respectively.

2.4 Downsampling of Dataset

The full dataset still contains some level of redundancy even with the Hobohm redundancy reduction using Levenshtein similarity. Two different types of partitions were created. One to investigate how the model performs when redundancy is reduced further, and the other to see what happens when the amount of training data becomes limited. The first approach is similar to the approach used for data creation using Hobohm 1. The other uses random sampling to be able to subsample peptides individually.

2.4.1 Kernel similarity

Multiple datasets were created with a decreasing amount of redundancy. The approach used here is modified from the approach described in Section 2.1.1. Here a kernel similarity using k-mers [45] was used instead of the Levenshtein similarity as the similarity measure for the Hobohm 1 cutoff. The kernel calculates the similarity between two sequences without an alignment. The similarity is calculated using the BLOSUM62 matrix to compare all k-mers generated from the two sequences, with k being all integers in $k \in [1, 30]$. The kernel also uses the BLOSUM62 matrix to compare amino acids from the k-mers. The similarity is averaged over the CDR3α and CDR3β to obtain an overall similarity for the TCR.

The kernel similarity gives a value between 0 and 1, where a high value means high similarity,

which is used by the Hobohm algorithm to reduce redundancy. After obtaining a list of less redundant positives and negatives, they were randomly distributed to 5 partitions as before. The swapped negatives were dealt with differently. If a positive TCR was removed during the Hobohm 1 redundancy reduction, the swapped negatives generated from the removed positive TCR were also removed from the dataset. Else the swapped negative was added to the same partition as the corresponding positive TCR.

This was repeated for several similarity cutoffs (1, 0.98, 0.95 and 0.9), which yielded datasets with a decreasing number of observations (6897, 6345, 5698, 4534, respectively).

2.4.2 Random Sampling

A single peptide was randomly downsampled to create smaller datasets and investigate performance when training data for a single peptide is limited. Sampling was done stratified by downsampling each type of sequence (positive, swapped negative or 10x negative) individually from the primary dataset. The sampling was done for a single peptide while keeping all observations from the other peptides. If a positive TCR was removed, the swapped negative TCRs were also removed. This type of sampling simulates a situation where model performance on a single peptide can be seen as a function of the number of TCRs for that specific peptide, as opposed to the kernel similarity method, where all peptides were downsampled.

2.5 Network Architectures

2.5.1 Convolutional network

A 1-layer CNN similar to the approach used in NetTCR-2.0 [17] was implemented to benchmark against a simplified version of the NetTCR-2.0 model. The architecture can be seen in Figure 12 and consists of parallel convolutional and max pool modules followed by a 1-layer FNN using the output from all the parallel max pools as input.

Each CNN contains 20 convolutional filters with size 3. Each filter contributes one value to the dense input by doing a global max pool. Each type of sequence (MHC, peptide, TCRα and TCRβ) has its own set of filters, yielding a total of 120 filters. TCRs were either represented as the whole TCR or restricted to the CDR3. The global max pool means only the three positions in the sequence the filter deems most informative get to contribute to the prediction. The use of multiple filters then allows multiple different types of information to be captured from the sequence and used for prediction. The global max pool is a simple attention mechanism by selecting the highest value and has no learnable weights directly tied to attention. Instead, the convolutional filters are responsible for finding motifs and ensuring the attention is put on the important sections of the sequence. This architecture allows us to model our many-to-one problem by removing the sequence dimension through the global max pool.

The activation function used for the convoluted outputs was the sigmoid function. The dense layer had a size of 64 neurons and used the Tanh function as the activation function for the

hidden layer and the sigmoid function on the final output value to shrink it to the interval $\hat{y} \in (0, 1)$.

This network was also used to investigate the importance of the different features in the data. Here, the model also uses the MHC and the energy features, which are not used in the LSTM-based models. The global energy features were added after the convolutional layers, as these describe the complex as a whole and do not need max pooling to generate global terms. Certain sequences or features such as the sequence encoding was removed as input during training and evaluation to find the model's most important features and sequences. However, only features used in all models (peptide and CDR3s without any energy features) were used to ensure a legitimate comparison between the different architectures.

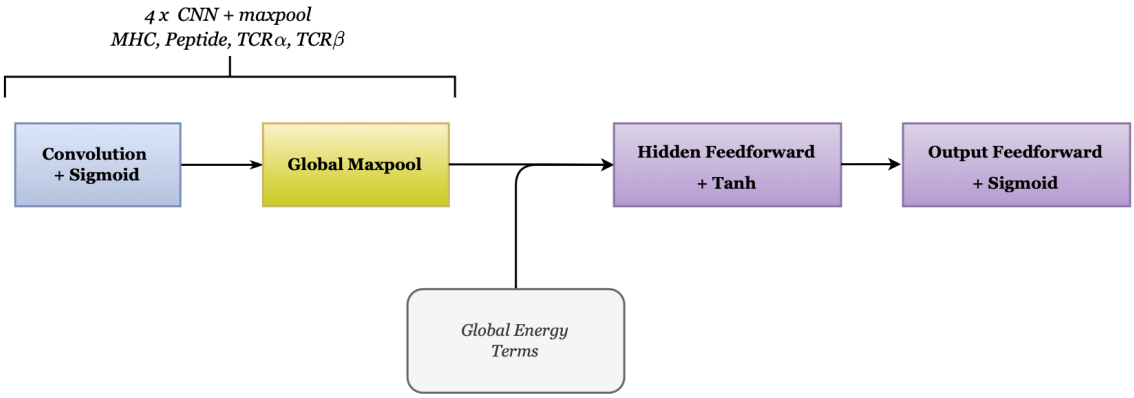


Figure 12: CNN model used in this project. The model consists of 4 parallel CNNs for the encoded sequences. The CNN output is concatenated and used as input in a dense layer to generate output predictions.

2.5.2 LSTM network

A simple Bidirectional LSTM (biLSTM) network was implemented as a baseline LSTM based neural network. A schematic view of the network can be seen in Figure 13. Here the different sequences (peptide, CDR3α and CDR3β) are used as input to three parallel biLSTMs. The hidden representation of the biLSTM had a size of 24 for all sequence types. The last hidden state from all three biLSTMs is then concatenated to a vector with 144 ($24 \cdot 3 \cdot 2$) entries and fed into a dense layer using the same activation functions and dimensions as the CNN model.

This architecture should propagate the most important signals through the recurrent network, which is then recovered as the last hidden state. The dense layers are again responsible for generating the prediction. However, the input comes from parallel LSTMs instead of parallel CNNs with global max-pooling.

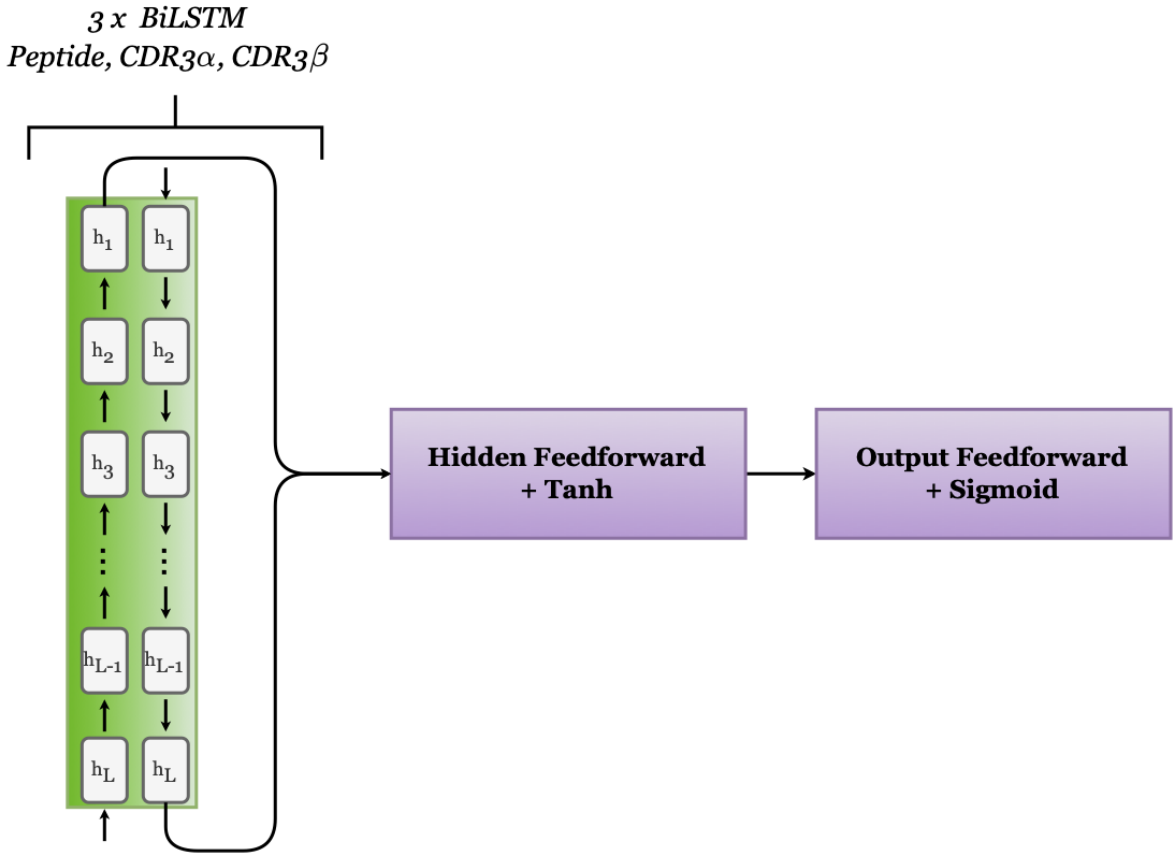


Figure 13: LSTM network used in this thesis. The hidden output from each of the directions in the biLSTM is concatenated with the other sequences and used as input for a 1-layer dense network.

2.5.3 AttLSTM network

Attention can create an importance weighted average of the different positions in the sequence and allow hidden states from all positions to contribute to the dense input. The schematic view of this approach is shown in Figure 14. Similarly to the LSTM network each type of sequence (peptide, CDR3α and CDR3β) is passed through a biLSTM. However, instead of extracting the last hidden state, a weighted average of the hidden states for each position is used. The attention layer introduced in Section 1.2.3 calculates the weight for each position's hidden state, which is then used to generate a weighted hidden state.

The hidden dimension in the biLSTM was set to 24 as in the LSTM network. The hidden dimension in the attention layer (number of features in the context vector and hidden representation of the LSTM output, see Figure 7) was set to 24. The weighted average of the output states is calculated using matrix multiplication for both directions of the biLSTM and concatenated with the different sequence types, yielding a vector of length 144 as input

for the dense layer, which is identical to the previous explained dense layers.

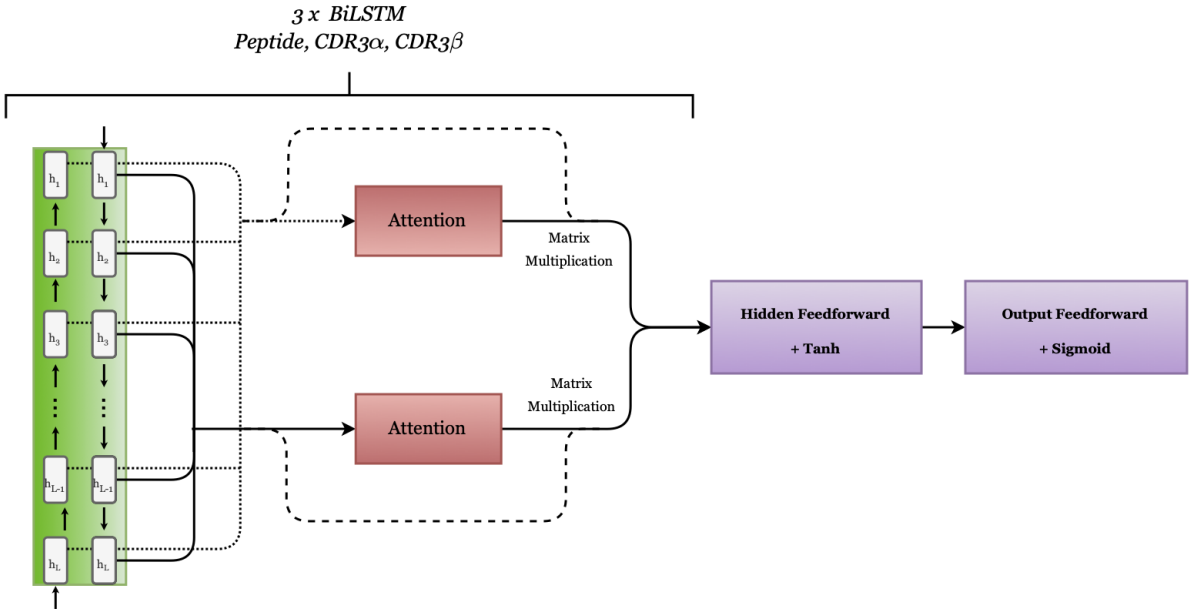


Figure 14: attLSTM architecture. A biLSTM generates an output for each position. Each output is then combined using a weighted average, by using attention to determine the importance of specific positions. The weighted average is concatenated with the output from the other sequences and used as input in a single dense layer.

2.6 Amino Acid Embeddings

Embeddings were generated using the Word2Vec model [35, 46]. Due to the low number of unique peptides in our dataset, generating a peptide-specific embedding using Word2Vec is not feasible. However, CDR specific embeddings were created by taking positive non-crossreactive TCRs with specificity towards peptides bound to Human leukocyte antigen (HLA)-A*02:01 from the 10x dataset (n = 9606). The positives were supplemented with 190000 randomly sampled TCRs negative towards peptides from HLA-A*02:01 from the 10x dataset. Additionally, to investigate the ability of Word2Vec to extract information on biophysical properties of amino acids, 20000 sequences were randomly sampled from the Swiss-Prot database [47] and used to train a general protein embedding used only for visualization.

CDR3α and CDR3β embeddings were trained separately using the skip-gram algorithm. After training, the embeddings for each residue consist of a vector with a size of 20. For the peptide sequence, the embedding weights were initialized randomly from $\mathcal{N}(0, 1)$. These embeddings are initially used to represent the sequence similar to the approach with BLOSUM encoding. However, the embeddings were updated during model training, thereby fine-tuning them to improve the embeddings and be specific towards predicting TCR-pMHC binding.

2.7 Training Neural Networks

All networks were implemented in Pytorch 1.10.2 [48] and Python 3.8.3. All networks were trained using a weighted binary cross entropy.

$$-\frac{1}{N}\sum_{i}^{N} w_i \cdot y_i \cdot \log(\hat{y}_i) + (1 - w_i) \cdot (1 - y_i) \cdot \log(1 - \hat{y}_i) \tag{7}$$

Here N is the number of observations, w_i is the weight on the positive class calculated as the fraction of negatives in the training data, y_i is the class, either 0 or 1, and $\hat{y}_i$ is the prediction for the observation.

The weighted version was introduced to increase the importance of the positive observations in the loss function. Using an unweighted version of cross entropy resulted in the model only predicting the negative class. The Adam optimizer was used to update the weights using a learning rate of 0.005 [49] and the batch size used was 64. For initialization of the weights, the uniform kaiming was used [50].

All networks contained dropout layers with a probability of 0.3 before and after the convolutional or LSTM layers and at the hidden dense layer. The dropout was enabled during training and subsequently disabled for evaluation. Early stopping was used with patience set to 10 epochs to avoid overfitting further. If there was no improvement in the loss after the 10 epochs, training was stopped.

Performance evaluation was done using a nested cross-validation approach. One partition was kept as a test set, one was chosen as a validation set for early stopping and three partitions were used for training the model. After training, the test set was predicted using the trained model, and the partitions were reassigned to training, validation and test. After the nested cross-validation, each observation had four predictions. The average of the four predictions was taken to generate a single prediction value.

AUC was calculated using scikit-learn and was calculated primarily only using the positive and true negatives. The swapped negatives were not included unless explicitly stated in the AUC calculations.

2.8 Baseline model

A sequence similarity-based baseline was constructed to compare the performances of the various networks. The model does inference based on sequence similarity to positive TCRs binding the same peptide. The underlying reasoning is that TCRs with high similarity to a positive TCR are more likely to be positive than a TCR with low similarity. The similarity metric is calculated as an average of the similarity of both CDR3 chains using the kernel approach described in Section 2.4.1.

Using 5-fold cross-validation, one partition is used as a test set, while the positives from the remaining 4 partitions are used as a database. Each TCR in the test set is then scored to

all positive TCRs with the same peptide specificity in the database, and the highest score is reported as the final prediction score for that TCR. These prediction scores ($\hat{y} \in [0, 1]$) are used to calculate the final AUC score.

2.9 Bootstrapping

Bootstrapping was used to compare two models. The null hypothesis for this test is that the two models have the same performance. One model was selected as the null model and tested against the other model. The models were trained, and predictions were made using nested cross-validation. After having achieved predictions for all observations, these predictions were sampled with replacement to create a new set of predictions with the same number of observations. The AUC could then be calculated for both models on the newly sampled predictions. The sampling and calculation were repeated 10000 times to give a distribution of AUC values used to calculate a p-value. The p-value for the null hypothesis is calculated as the fraction of times the tested model obtains a higher AUC than the null model.

3 Results

3.1 Characterization Of Dataset

The complete dataset consists of 6897 pairs of TCR-pMHC sequences with both TCRα and TCRβ sequences. The data consists of both positive TCR-pMHC pairs from VDJdb and IEDB. The negatives were taken from the 10x dataset and added in a 3:1 ratio to the positives. Adding additional negatives in the form of swapped negatives increases the ratio to 4:1. The difference in the number of positives and negatives means the dataset has a large class imbalance toward negatives. This is a more accurate representation of the natural setting for the immune system, where it would be expected that a TCR would bind only a few of the peptides it was presented to. However, the model needs to focus on predicting the positives in the dataset, which is achieved using a weighted loss.

In Figure 15 the length of all CDR3s are shown. The CDR3α has lengths between 8-15 residues, while the CDR3β has lengths between 8-17 residues. The most striking observation here is the number of positive CDR3β with a length of 12. The large bias can also be seen in the germline genes for the TCRs (Figure 16-17). Besides the large bias in CDRβ lengths, the sequences are primarily distributed with lengths between 10-14 for CDR3α and 10-16 for CDR3β with a few outliers. The bias in the CDR3β can pose a real issue when training the model, as the model could only learn the germline bias in the CDR3β and ignore other more interesting signals. The bias should be kept in mind when interpreting the results in this thesis, as the results might represent only a subset of the CDRs in the dataset.

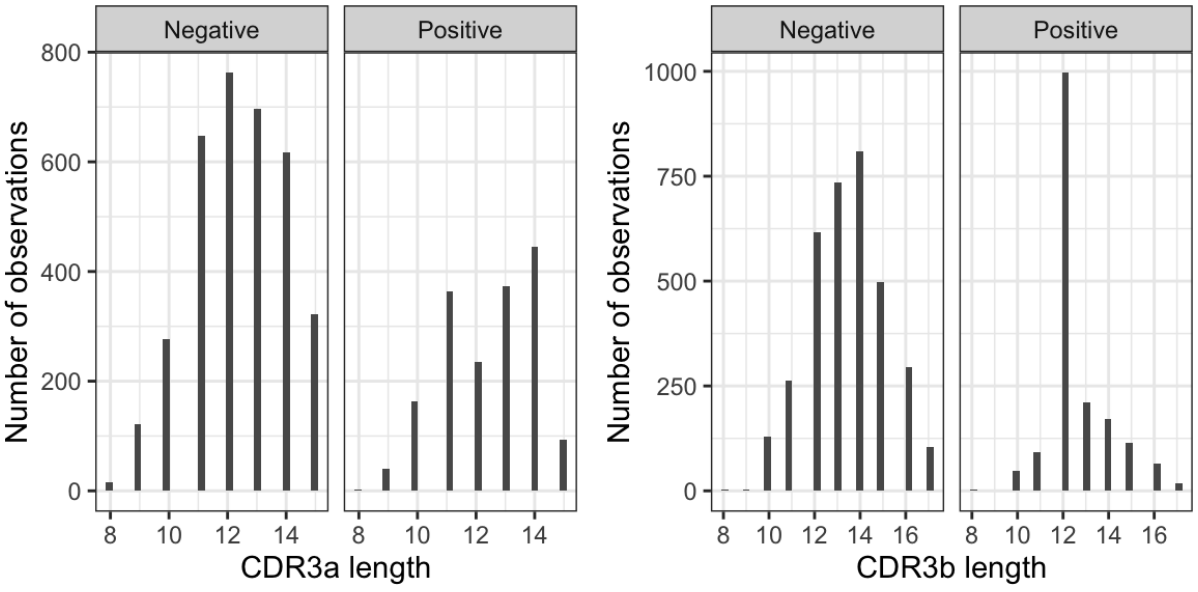


Figure 15: CDR3 lengths for 10x negative and positive TCRs.

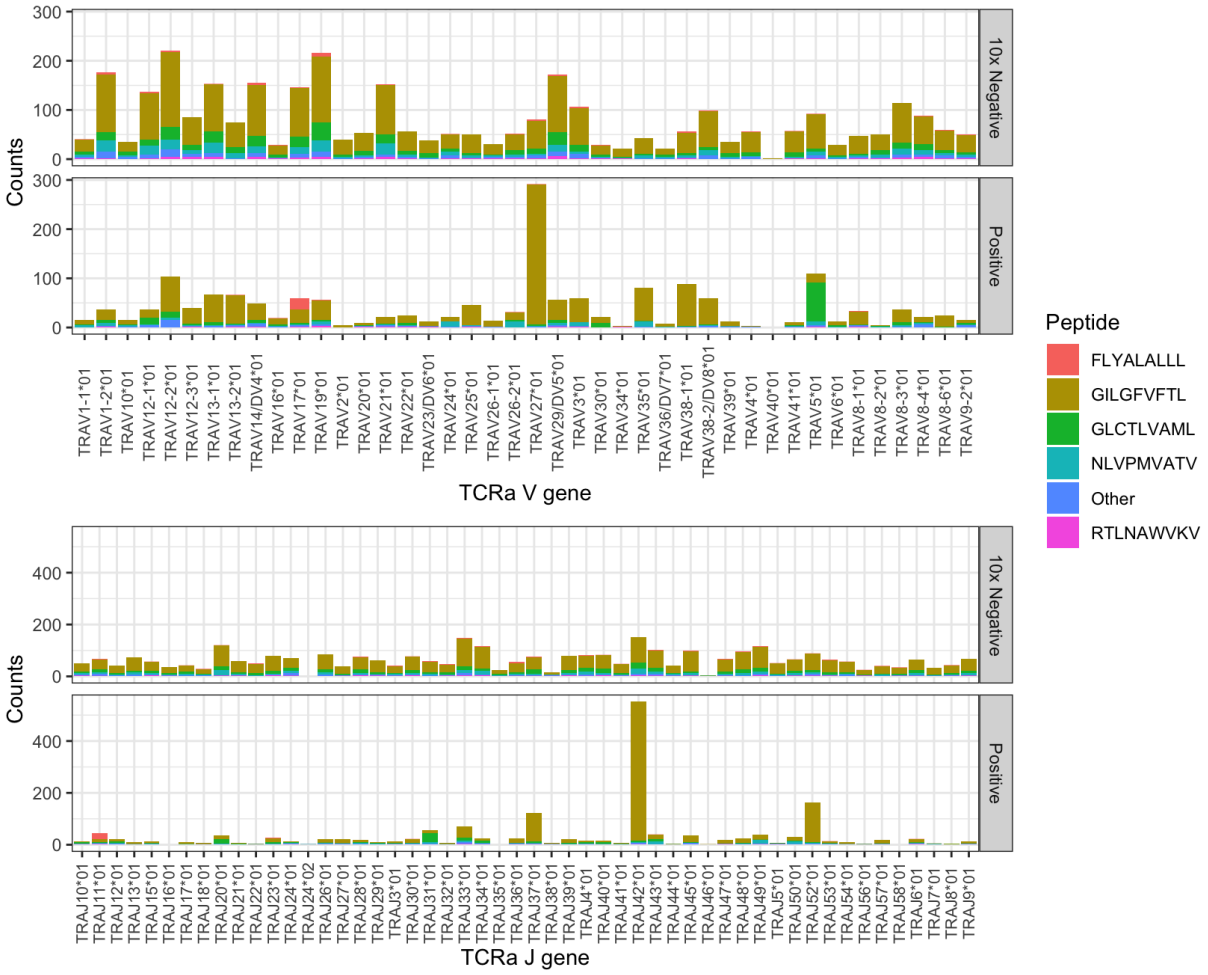


Figure 16: Germline genes used in the TCRα sequences in the data.

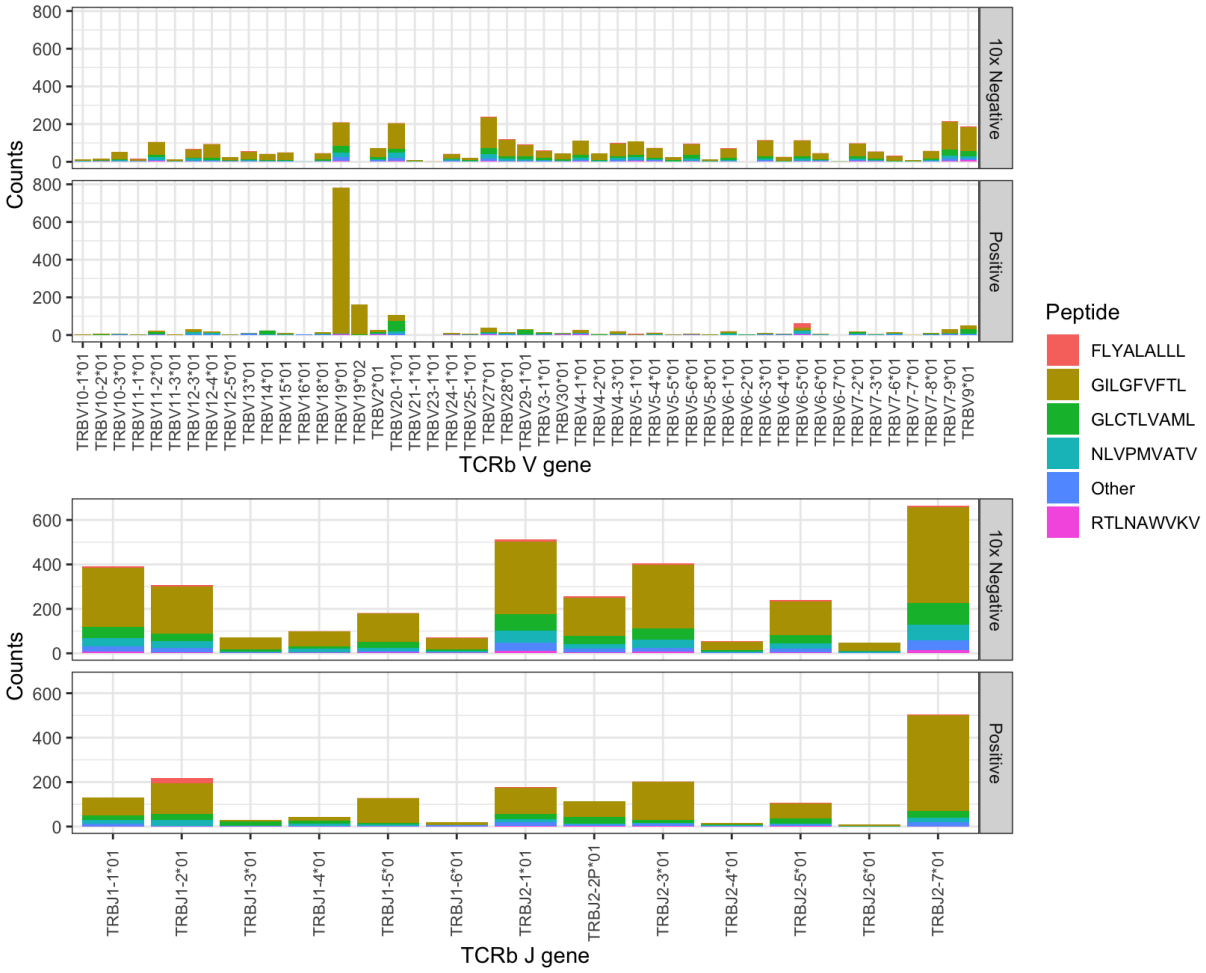


Figure 17: Germline genes used in the TCRβ sequences in the data.

There are 18 unique peptides in the dataset, which have a large span in terms of the number of observations. The number of observations per peptide is shown in Figure 18. The 10 most frequent peptides are displayed stratified on their origin. There is a clear bias for the GILGFVFTL (GIL) peptide in the positives, with 1233 out of 1718 positives. Because we preserve the ratio between positives and negatives across peptides, there is also a large bias in the negatives towards GIL.

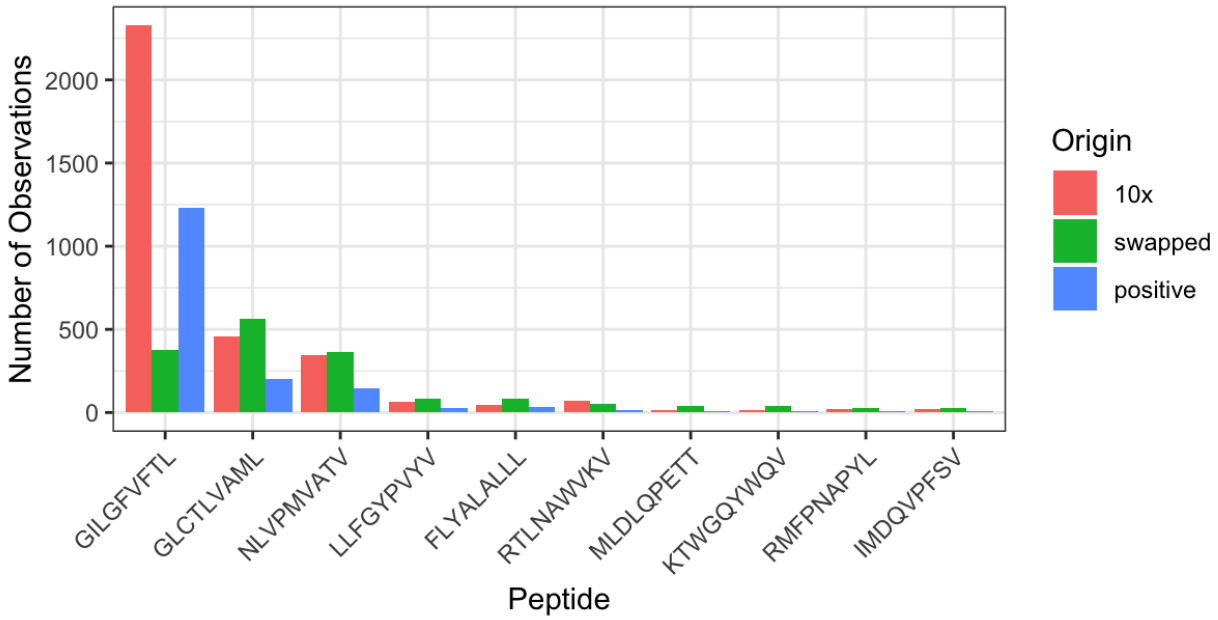


Figure 18: Number of observations per peptide and origin for the 10 most frequent peptides.

The swapped peptides have a different distribution than the other types of observations. Here a random peptide was chosen from positive observations with other peptides. Since most positive observations are positive for GIL, a TCR positive for the GLCTLVAML (GLC) peptide will often be observed as a swapped negative with the GIL peptide. If the swapped peptide was chosen from a unique list of peptides, there would rarely be any swapped observations with a GIL peptide since most positive observations are from GIL. By choosing a random peptide from the dataset, the swapped peptides have a distribution more like the rest of the data compared to creating swapped from a unique list.

The bias in the peptides can affect the pan-specific models' ability to predict infrequent peptides in the dataset since the signal from these observations is overshadowed by much more frequent peptides such as the GIL peptide.

3.2 Locating important features for model performance

It is unreasonable to assume all sections of the TCR are equally important for recognition of the pMHC complex, as the CDRs are much closer to the displayed peptide than the framework regions on the TCR (Figure 3). Therefore, we investigated which areas of the

sequence were required for performance and which features were the most substantial for TCR-pMHC binding predictions. The CNN model was trained on varying amounts of input by leaving out either different sequences and features.

Figure 19 shows the AUCs for these experiments. The TCR and peptide are needed in some capacity to achieve meaningful performance in most cases. Removing the peptide restricts the model to distinguish between positive and negative TCRs in general instead of distinguishing at a peptide-specific level, which results in a performance drop. Removing the MHC has no real effect on performance and, in most cases, achieves the same or better performance. This is unsurprising, as the data used here only contains HLA-A02*01. The input does not add any information and the higher number of parameters most likely leads to overfitting for models containing the MHC. Restricting the TCR to only the CDR3s gives the same or a slightly increased performance in all cases except for only energy features. This supports the idea that the CDR3s are the most important parts of the sequence for determining TCR specificity [8, 51].

Interestingly, using only the energy features as input allows models trained even without the TCR to recover some performance. The most likely explanation to this is that large interaction energies increase binding probability. It thereby gets some information about the TCR through the interaction energies between the TCR and peptide.

Often the most interesting model is also the simplest model. Feature selection was done by first evaluating the significance of the top-performing model for each type of feature input and selecting the simplest model of the significantly best performing models. Afterwards, all models from that feature type were tested against each other, and the simplest model of the significantly best performing models was chosen.

When selecting the most important features, the model with only energy features was significantly worse than all other models ($P < 0.05$ for all tests using bootstrapping with 10000 replications), while all other models were equivalent. For selecting the most important sequence, the model with only peptide and CDR3 is significantly better than all other models ($P < 0.05$ for all tests using bootstrapping with 10000 replications). Therefore, we focus on training models using only the sequence features for the peptide and CDR3s for the remainder of this thesis.

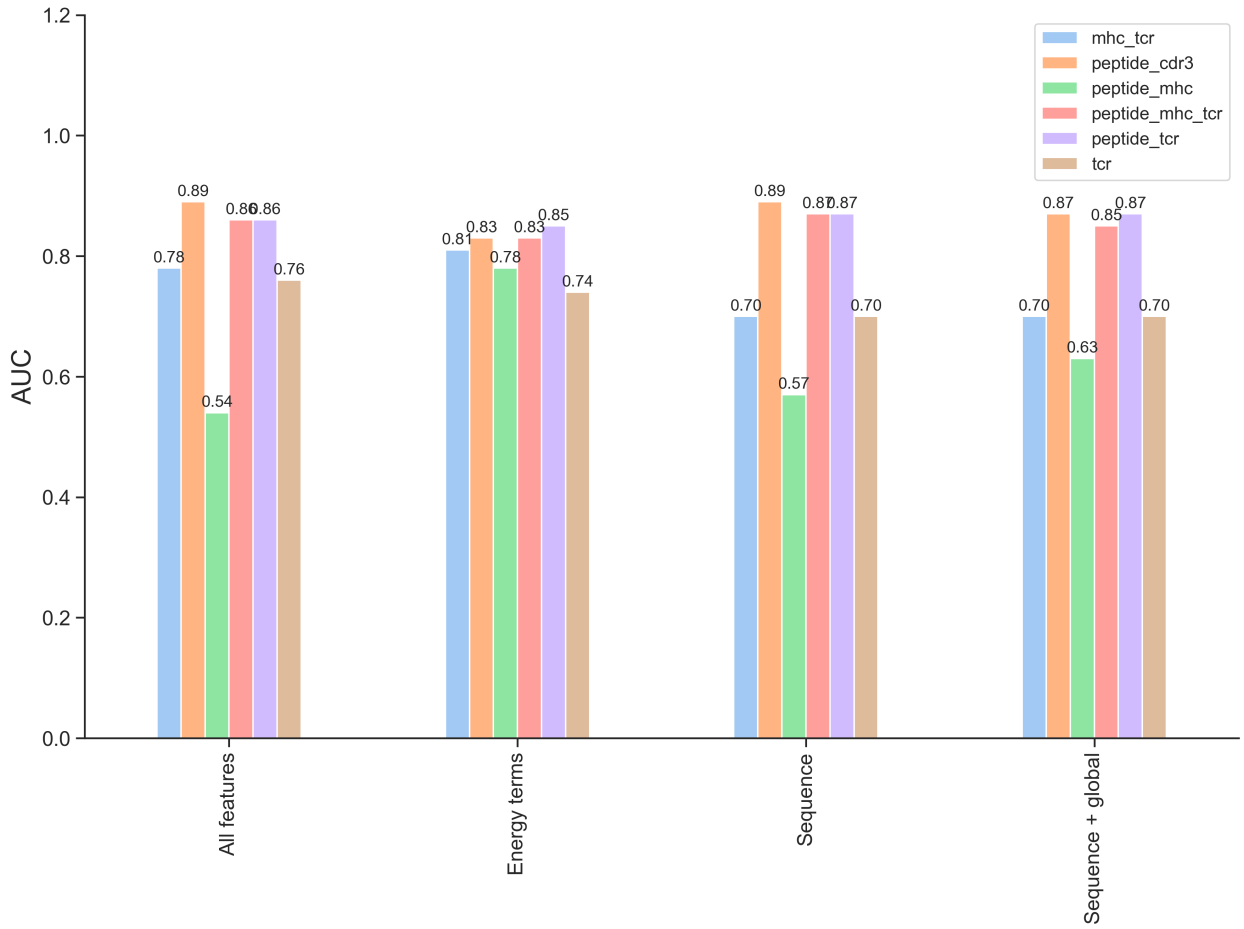


Figure 19: AUC for the CNN model trained on different amounts of input. AUC was evaluated using nested cross-validation. All features refers to using sequence, per-residue energy features and global energy features. The energy terms are only trained on per-residue and global energy features. Sequence is trained on only the sequence encoding. Sequence + global leaves out the per-residue energy features. These features were then trained with different number of sequences. The legend describes the types of sequences used in that model.

Paired CDR3α and CDR3β data is more expensive than bulk sequencing of only the CDR3β. We investigated whether a paired model performs better than models trained only on CDR3α or CDR3β. CNN models were trained on the same data using the peptide and either both or only one of the CDR3 sequences. As seen in Figure 20, the paired CDR3 approach obtains a score of 0.88, compared to 0.81 and 0.83 for the CDR3α and CDR3β respectively. The difference between the paired model and the individual sequence is significant ($P < 0.05$ using bootstrapping with 10000 replications). In contrast, the difference between the two individual models is not significant ($P = 0.42$ using bootstrapping with 10000 replications).

Both chains seem to be important for generating accurate TCR-pMHC binding predictions. While the model using only the CDR3β does slightly outperform the CDR3α, there was no significant difference between the two. Thereby we have reduced the input space for

the model from having MHC, peptide and the entire TCRs represented by both a sequence encoding and predicted energy features to include the peptide and CDR3s from the TCRs represented by a sequence encoding.

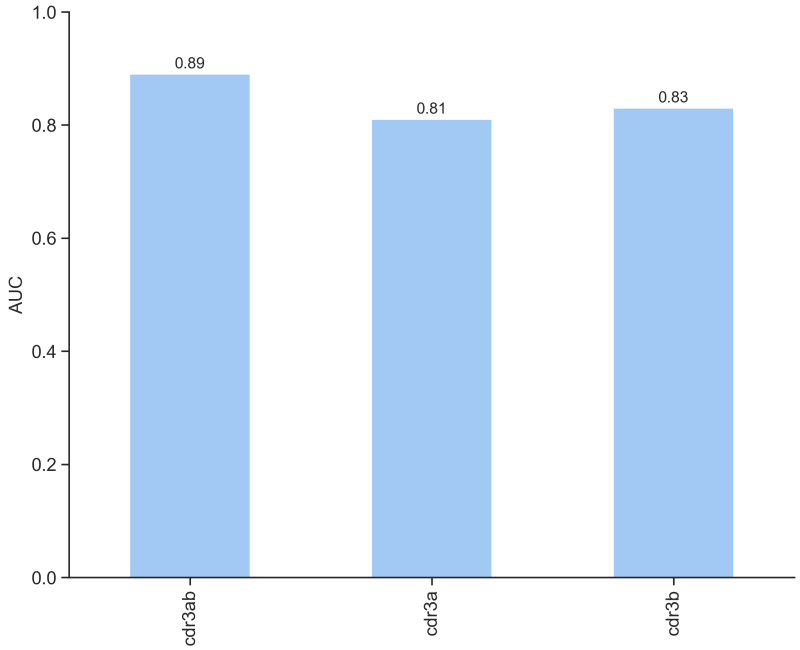


Figure 20: AUC for CNN models trained on both chains (cdr3ab) or single chains. AUC was calculated using nested cross-validation.

3.3 Attention Mechanisms Focuses On Residues Specific For Positives

This section describes which areas of the CDR3s the model chooses to focus on and investigate if there is any relationship to differences between positive and negative sequences.

3.3.1 Positions Selected By CNN Maxpooling

CNN based models have previously been used for predictions of TCR-pMHC interactions [30, 17]. The CNN models contain a max pooling operation, which allows the network to select the most important positions for predicting T cell interactions, and also allows the model to disregard specific areas on the sequence. The positions selected from filters might provide insight into important positions used to identify positive TCRs and how the model determines what should be classified as positive and negative.

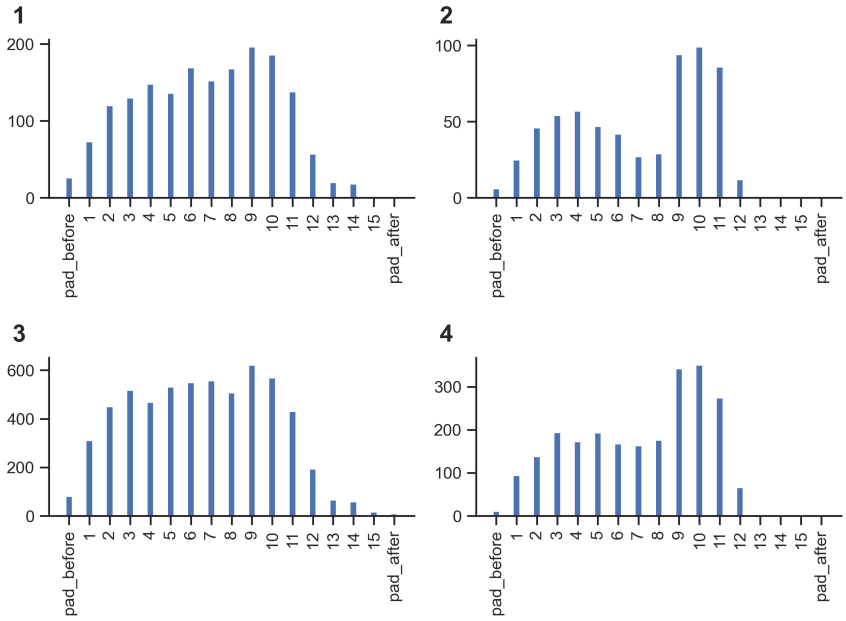
As mentioned in Section 1.2.1, the convolution operation incorporates information from adjacent positions (For this model, three positions). The max pooling operation then selects a

single convoluted value by selecting the maximum value. Each filter in the network represents information about three of the original positions used to create the convolutional output value. The number of times a filter selects a specific position can therefore be used to evaluate how important each position is relative to other positions in that sequence.

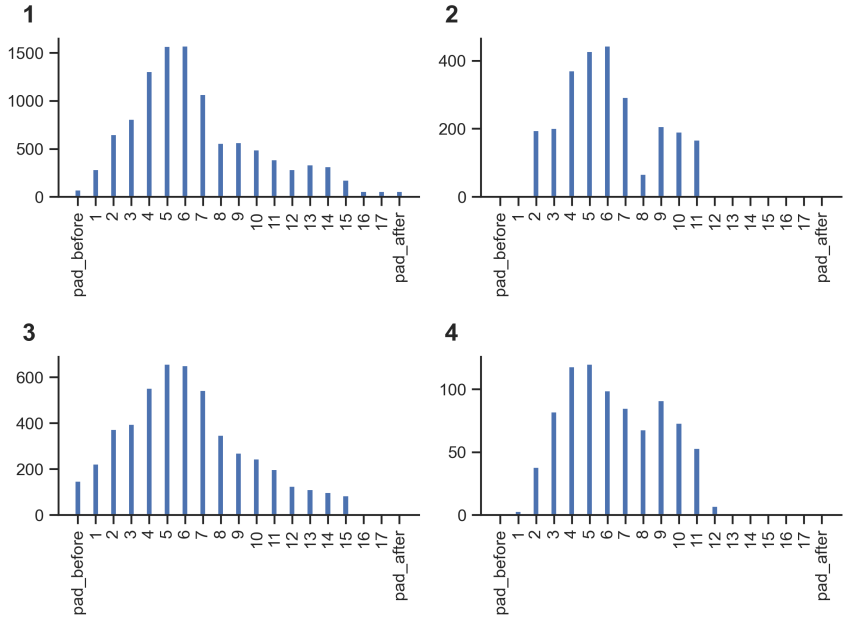
The counts for each position can be seen in Figure 21a for CDR3α and Figure 21b for CDR3β. The figures show CDR3s that were either positive (graph 1) or 10x negative (graph 3) towards GIL. As the filters used has size 3, each position is convoluted three times and can be selected at three different positions. The first and last two positions are convoluted only once or twice. Therefore, it is expected that these positions will have lower counts than others. There is, however, a clear difference in what the important regions for the CDR3α and the CDR3β are. The CDR3β primarily focuses on positions 4-7, while the CDR3α has no enrichment, except for a slight increase for positions 9 and 10. However, no clear distinction can be made between the positives and 10x negatives for either CDR3s. There are slightly more counts at positions 8 and 9 for the 10x negative CDR3β relatively to the same positions for the positive CDR3β.

The model contains many filters, and some of these filters might contribute noise or only a little information for making predictions. The convolution value is passed through the sigmoid function before max pooling. Since the sigmoid squishes values to the interval $(0, 1)$, then large values will have a bigger impact in the dense network, and are therefore deemed more important by the network. Unimportant filters can therefore be removed by only counting convolutions whose value is larger than a threshold (0.9) When comparing the number of filters per position without setting a threshold (graphs 1 and 3) to with a threshold (graphs 2 and 4), the most substantial change can be seen on the CDR3α. Here, there is a larger enrichment at positions 9-11 than without the threshold. The counts for the CDR3β with threshold are similar to without, except for a drop at position 8. For both CDR3s, the model has learned which sections contain padding (positions 13 and up) and actual information. The number of filters placed in the padding is substantially smaller with a threshold than without the threshold. Filters that sit in the padding likely have a low influence on the prediction due to the low activation value it carries over into the dense layers. The same analysis was performed on NLVPMVATV (NLV), with similar results (Appendix A).

The model has learned to focus on areas with information and learned to limit the information from padding. However, there was no clear distinction between positions chosen for positives and negatives. Therefore binding probability might be determined by differences in residues at these positions or interactions between individual filters in the hidden layer.



(a) Histograms for CDR3α.



(b) Histograms for CDR3β.

Figure 21: Histograms of the number of times a position is selected to contribute information by the max pooling operation. Positions are considered contributing if they are used in the convolution selected by max pooling. **(1)** Positions selected for CDR3s with positive interaction. **(2)** As (1), but only considering filters with an activation above 0.9 **(3)** Positions selected for negative CDR3s from the 10x dataset. **(4)** As (3), but only considering filters with an activation above 0.9. Data shown for partition held out during training limited to CDR3α and CDR3β with length 14 and 12 respectively and specificity for GIL.

3.3.2 Positional Attention On biLSTM Positions

While max pooling a convoluted sequence can be regarded as attention, the attention is learned purely based on the weights inside the convolutional filters. Instead, the attLSTM model uses a learnable weight vector and matrix to calculate the most important positions. It can incorporate information from multiple areas of the sequence using a single context vector, whereas max pooling only transfers information about 3 adjacent residues.

The capabilities of the attention mechanism was shown by training The attLSTM using TCRs instead of CDR3s. The positional attention for each positive sequence is shown in Figure 22. We expect to find the attention primarily on areas important for binding predictions, and it should ignore other areas. For the TCRα the attention on the sequences is almost contained to the CDR3, with 48% of the attention being put on the CDR3. This percentage is much higher than the 15% expected for a uniform attention across the entire sequence. For the TCRβ the attention is more spread, and some attention is put on framework regions that are unlikely to have importance for binding. However, there is still a clear tendency to focus on the CDR3, which contains 25% of the attention in contrast to the expected 15% for uniform attention. The noise is perhaps due to a large germline bias in the positive CDR3 observations (Figure 16-17), which might allow the model to learn positives purely based on conserved areas in the framework regions for that germline. Therefore, attention allows the model to capture information from the areas in the sequence known to be important for binding [10].

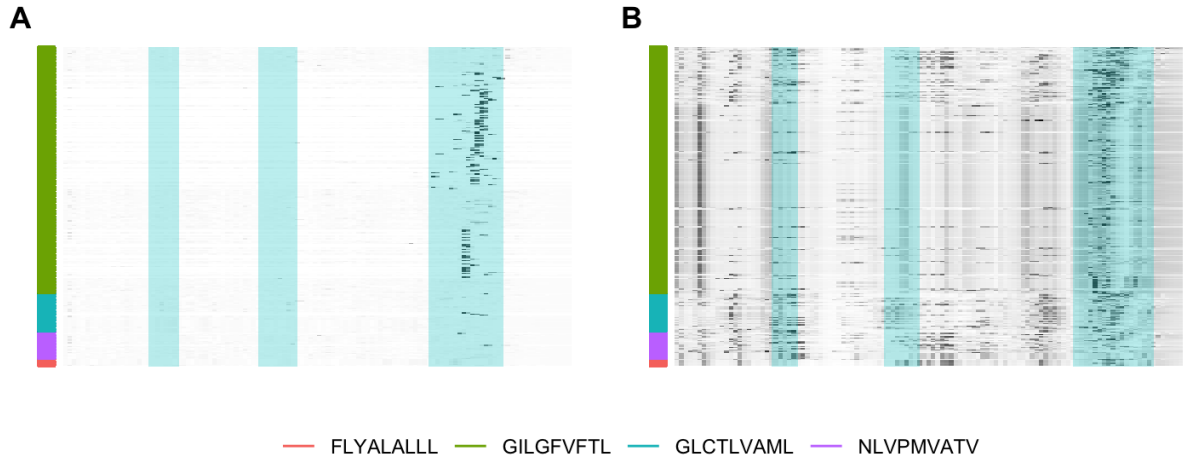


Figure 22: Importance of positions for generating the weighted hidden state for the forward direction of positive **(A)** TCRα and **(B)** TCRβ. Blue annotations reflect approximate positions of CDRs. Data shown for partition held out during training.

The approach was then applied only CDR3s to locate important positions in the CDR3s, and see if there were any similarities to the results found for the CNN model.

The attention vectors for positive and negative CDR3α and CDR3β is plotted in Figure

23. The positive CDR3α (Figure 23A) has a clear pattern in the sequences specific for GIL, where the model focuses around position 11-12 for the longest sequences (length 14). Since sequences have been padded to the same length on the right, the positions focused by the attention are located at the same relative position in the sequence. These positions may therefore be important for discriminating between positives and negatives for GIL. The pattern for CDR3αs is less clear on peptides other than GIL The attention is more spread and does not contain the same length-dependent pattern. When that is said, some sequences seem to have a preference for position eight in some manner. The length-dependent pattern cannot be recovered for any of the peptides for the negatives. Instead, there is some preference for the first position, which is also seen in some of the positive observations. The explanation for this could be that the attention layer primarily looks for motifs in positive sequences, and something more arbitrary is chosen if these motifs cannot be found. The first position is unlikely to provide much information as it is fairly conserved for all sequences.

The length-dependent pattern indicates that CDR3αs positive for GIL have some motif in the latter part of the sequence not found in the negatives. The GIL's clear pattern and absence in other peptides could indicate overfitting to the GIL peptide, which would not be surprising, as most of the data is specific for this peptide.

The positive CDR3β also contains the length-dependent pattern (Figure 23C). However, for a large number of the positive sequences, the attention is primarily focused on positions 6 and 7. These are CDRs with length 12, and have a large germline bias (Figure 15-17). Compared to the CDR3α, where the length-dependent did not repeat for peptides other than GIL, the CDR3β has the same pattern reappearing in the less frequent peptide as well. This could mean the learned attention is more general than for the CDR3α and that these positions are important for peptide binding predictions for more than just GIL. The negative CDR3β similarly has a (Figure 23D) length-dependent pattern. The positions selected by attention are roughly the same for positive and negative CDR3β. The difference in the plots is due to differences in length distributions (Figure 15), as the positives have a huge overweight of peptides of length 12.

Attention on the CDR3β seems to capture some signal at the middle of the sequence, and these positions carry most of the information required. The strong attention put on specific positions for peptides with a length of 12 could point towards the model overfitting or finding biological relevant residues for predictions on these peptides. These constitute a large section of the positive dataset, whereas noticeably fewer of the corresponding negative CDR3βs with length 12 exist. This indicates that the model is learning some of the dataset's bias. Therefore, even though there is clear attention at positions 6 and 7, it is probably not a general way to discriminate between positives and negatives TCRs. The attention of the reverse directions was more unstable between runs and contained a lot of noise (Appendix B). Training the model without the reverse LSTMs also gave comparable results to training it as a biLSTM (not shown). The model, therefore, seems to get most of its information from the forward directions, and the reverse directions do not contribute much, which could explain the high levels of noise and unclear patterns in the attention vector.

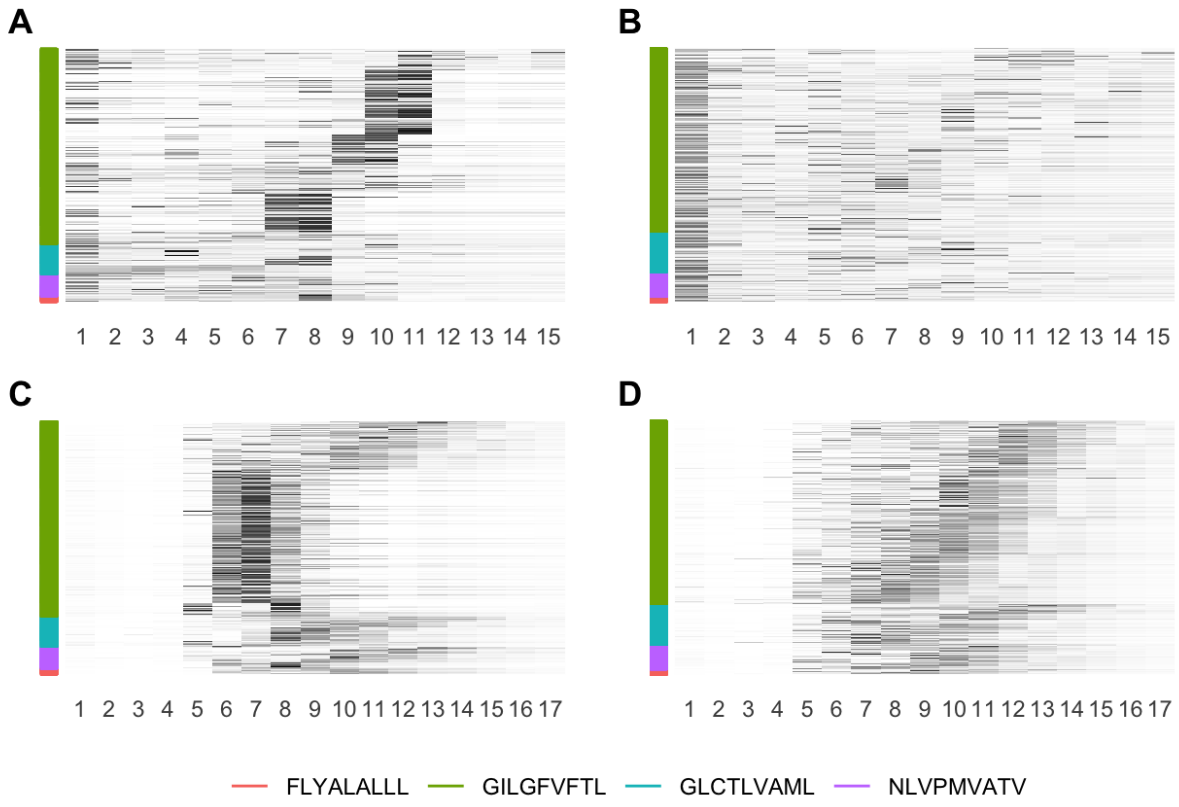


Figure 23: Attention on positions for generating the weighted hidden state for the forward direction of **(A)** Positive CDR3α, **(B)** Negative CDR3α, **(C)** Positive CDR3β and **(D)** Negative CDR3β. Sequences have been sorted by CDR length in descending order. Data shown for partition held out during training.

3.3.3 Sequence Logos For CDR3s

The positions with strong attention may contain specific motifs in the sequence, which the model focuses on and uses to distinguish between positives and negatives. The attention vectors (Figure 23) could be indicating where these positions are located. A sequence logo was generated for both positive and negative CDR3α of length 14 and CDR3β of length 12 to see if the attention patterns correspond to differences in the motifs. The attention for CDR3α of length 14 primarily focuses on positions 10 and 11, while attention for CDR3β with length 12 mostly focuses on positions 6 and 7 (Figure 23).

Comparing the logos for CDR3α with length 14 (Figure 24a and Figure 24b) shows that generally glycines are more conserved in the central positions for both the positive and negatives. The positions 9, 10 and 11 seem to differ the most between the two logos and concur with the difference observed between the two attention vectors for CDR3α sequences with length 14. The motif QGN is present in the positives but not in the negatives. Especially the glycine is quite conserved at position 10 and not present for the negatives. Other positions,

while more conserved in the positives, are also present in the negative logo.

Logos for GIL positive CDR3α with lengths 13 and 11 (Appendix C) has the same motif as found in sequences of length 14. However, the position of the motif has moved due to the shorter length, as the length-dependent pattern indicated in Figure 23. In these logos, the over-represented glycine instead resides in position 7 or 9 and the asparagine in position 8 or 10 for the 11 and 13 long CDR3α. The QGN motif is a central part of the TRAJ:42*01 gene, meaning this is not part of junction residues, but a result of gene bias present in the data.

The logos for CDR3β with length 12 primarily differs in arginine and serine present in the positive logo (Figure 24c), but not present in the negative logo (Figure 24d). However, these differences are located at positions 5 and 6 and do not line up completely with the results found in Figure 23, where the focus was primarily on positions 6 and 7. Instead, it lines up with the results seen for the CNN model (Figure 21b). Why this occurs is unknown, but the LSTM might pass the information from the input at positions 5 and 6 to position 7 through the hidden and cell state passed to position 7, where the attention mechanism captures the signal.

The attention vectors have learned some of the key differences in the sequence between the positive and negative CDRs. The QGN motif in the CDR3α is a central part of the discrimination between positive and negative for GIL. The motif is a part of TRAJ:42*01 and is therefore not limited to a single length but can be found in sequences with multiple lengths as seed by the attention vectors (Figure 23). For the CDR3β the model either passes the information about the sequence to the next position in the LSTM and incorporates it there or learns to differentiate in a more complex manner not captured by the sequence logos.

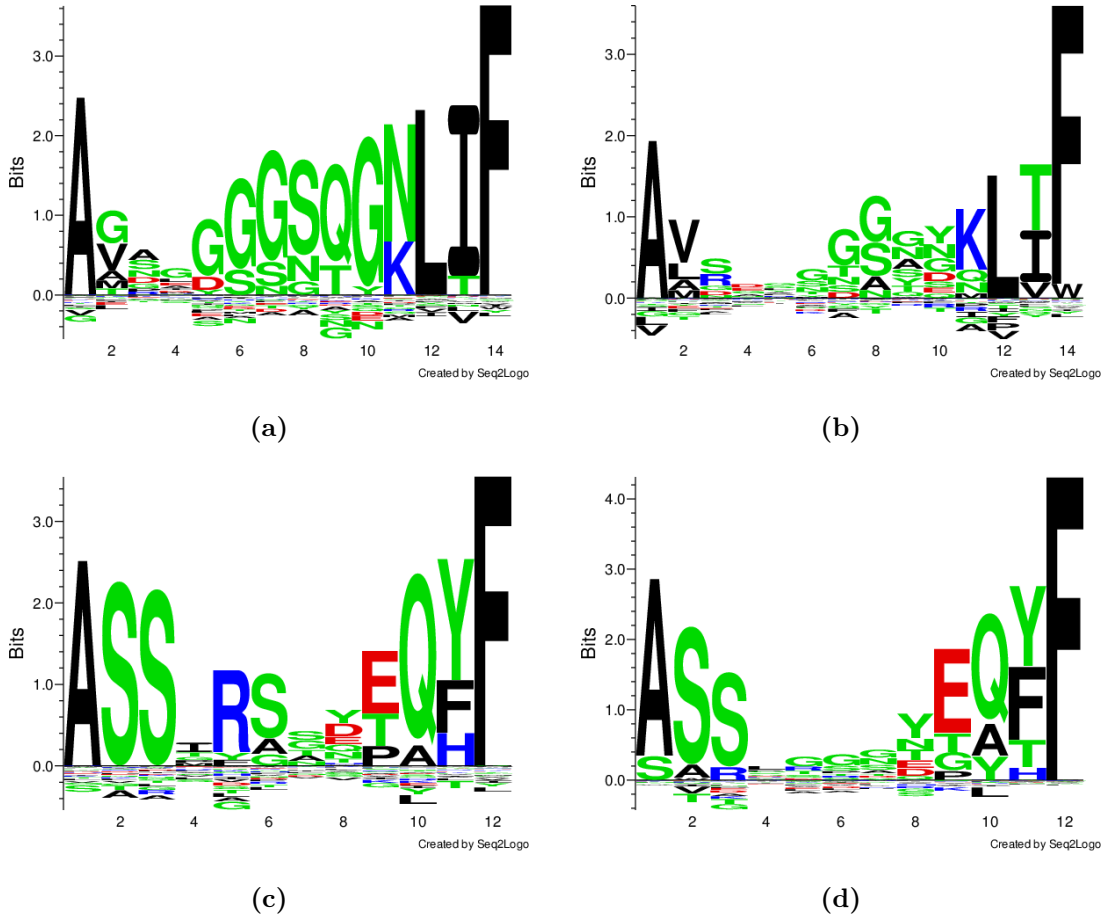


Figure 24: Logo plots for CDR3s of **(a)** positive and **(b)** negative CDR3α with length 14, and **(c)** positive and **(d)** negative CDR3β with lengtg 12. All plots were created using Seq2Logo webserver with 50 as pseudocount weight and no sequence clustering [52].

3.4 Word2Vec embeddings capture biochemical features of amino acids

The Word2Vec model can generate residue-specific embeddings for each amino acid by training the sequence to predict surrounding words. Related amino acids are used in similar contexts within the sequence and therefore predicts the same surrounding residues, and result in similar embeddings. This method is first validated on 20000 proteins randomly sampled from the Swiss-Prot database as described in Section 2.6. A visualisation of the embedding space reduced to two dimensions by t-SNE can be seen in Figure 25.

The plot shows amino acids cluster roughly according to the biochemical properties of the amino acids. Aromatic residues (F, Y and W) are closely related, while the charged residues cluster together (K, R, E and D). The only group not clustering together is polar amino acids. Unique amino acids are not expected to cluster together, as these are not related but generally have distinct properties from all other amino acids.

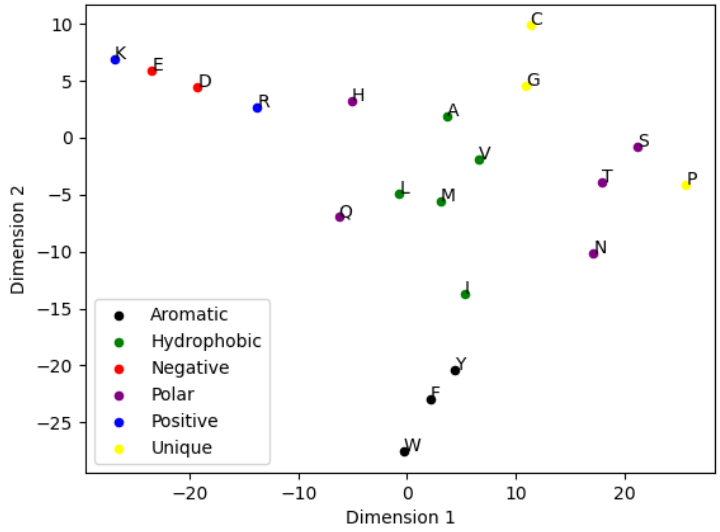
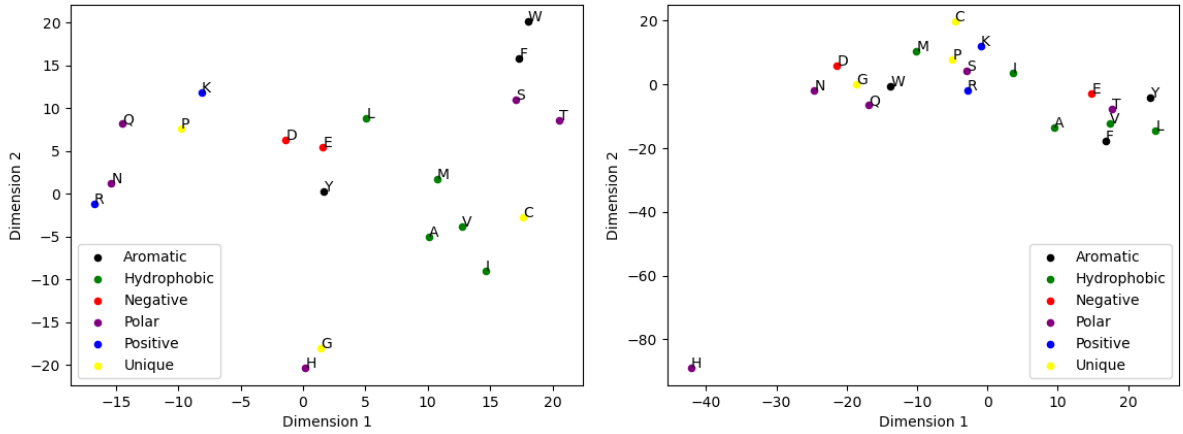


Figure 25: t-SNE Visualisation of Word2Vec embeddings trained on random proteins.

This Word2Vec model only capture rather general aspects of residues. A model trained specifically on CDR3s may generate embeddings allowing to easier separate a positive from a negative TCR. Two models were trained separately for CDR3α and CDR3β sequences. The embeddings are visualised in Figure 26. Both sets of embeddings do not contain the same clusters as seen in Figure 25. The CDR3α still has some similar residues close to one another (F and W, D and E, and hydrophobic residues). In contrast, CDR3β creates seemingly random groups and has histidine as a large outlier. The embeddings for the CDRs do not seem to capture biochemical features. This does not mean they are of poor quality and bad for predicting TCR-pMHC interactions. They may instead have learned a relationship between amino acids more specific to CDR3s than the general biochemical features.



(a) CDR3α sequences.

(b) CDR3β sequences.

Figure 26: t-SNE Visualisation of Word2Vec embeddings trained on CDR3 sequences.

3.5 Network performances reduces by decreasing redundancy

The performance of networks depends on the amount of redundancy present in the data. If the data is completely redundant, information leakage between partitions will increase, making predictions on a test set easier. Reducing redundancy also removes data examples, further increasing the difficulty of the prediction problem. The ideal model would be resilient towards redundancy reduction and retain performance as the redundancy is reduced.

The models described in the Methods section were all trained on datasets with decreasing redundancy to investigate which models would be most resilient towards redundancy reduction (Figure 27). All models were trained on the same data for each selected redundancy value. The CNN model at 0.9 redundancy is the only model not significantly better than the baseline trained on the same dataset (P-value = 0.23 using bootstrapping test with 10000 replications). Every other model is significantly better than the baseline model at all levels of redundancy (P-value $<$ 0.0001 using bootstrapping test with 10000 replications). The LSTM based models significantly outperform the CNN based model at all levels of redundancy (P-value $<$ 0.05 using bootstrapping test with 10000 replications).

The LSTM based models obtain almost equal performance at the highest level of redundancy. At low amounts of redundancy, the basic LSTM is significantly worse than the more complex models (P-value $<$ 0.0001 using bootstrapping with 10000 replications). Lastly, the effect of incorporating the CDR3 specific embeddings versus a BLOSUM encoding can be seen by comparing the performance of the attLSTM and the Embedded attLSTM (Figure 27). There was no significant difference between the two (P-value = 0.175), meaning using CDR3 specific embeddings provides no significant performance increase but instead has slightly worse performance than with the more general BLOSUM encoding.

Therefore, the attLSTM outperforms the other model types, and incorporating attention to

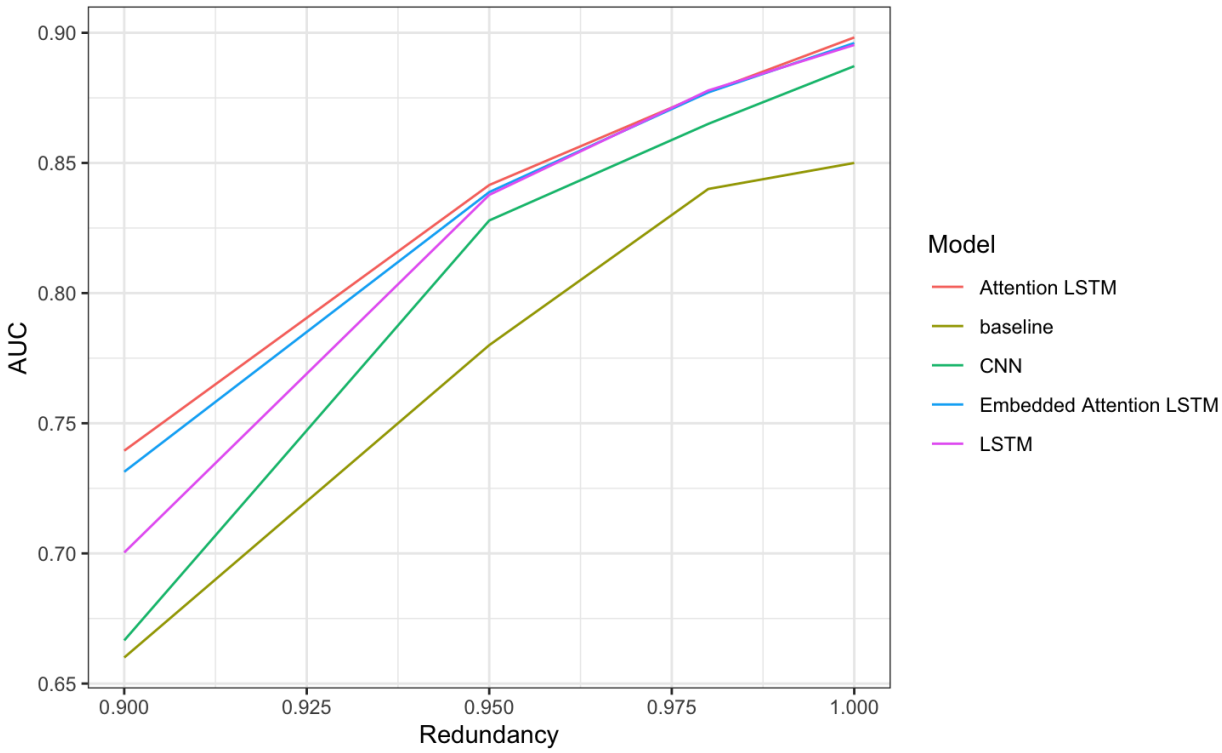


Figure 27: AUCs for the networks tested in this thesis. AUCs were calculated using nested crossvalidation.

better focus on important areas outperforms the other types of models tested. Incorporating more advanced embeddings into this model did not give a performance boost, and therefore the more interpretative BLOSUM encoding is preferable to the CDR specific embeddings.

3.6 attLSTM Network Performances

The attLSTM outperformed the other model architectures at low redundancy and was equivalent to the base LSTM at high redundancies and obtained an AUC score of 0.88 for the full dataset. To further describe the performance of the attLSTM model, the per peptide performances were investigated (Figure 28). As previously shown, the performance drops as redundancy is reduced. However, only a few peptides drive the total performance. Only the peptides with a high number of observations obtain performance beyond random (AUC of 0.5). The model achieves good performance on both the GIL and GLC peptides (AUC of 0.91 and 0.84) and even retains some performance at high levels of redundancy (AUC of 0.76 and 0.69). The NLV achieves some performance at high redundancy (AUC of 0.64), but the performance becomes random when the redundancy is reduced.

The remaining peptides generally obtain random or worse than random performance. The FLYALALLL peptide is an exception and obtains high performance with only a few positive peptides. This is because the sequence dissimilarity for positive and negative sequences is

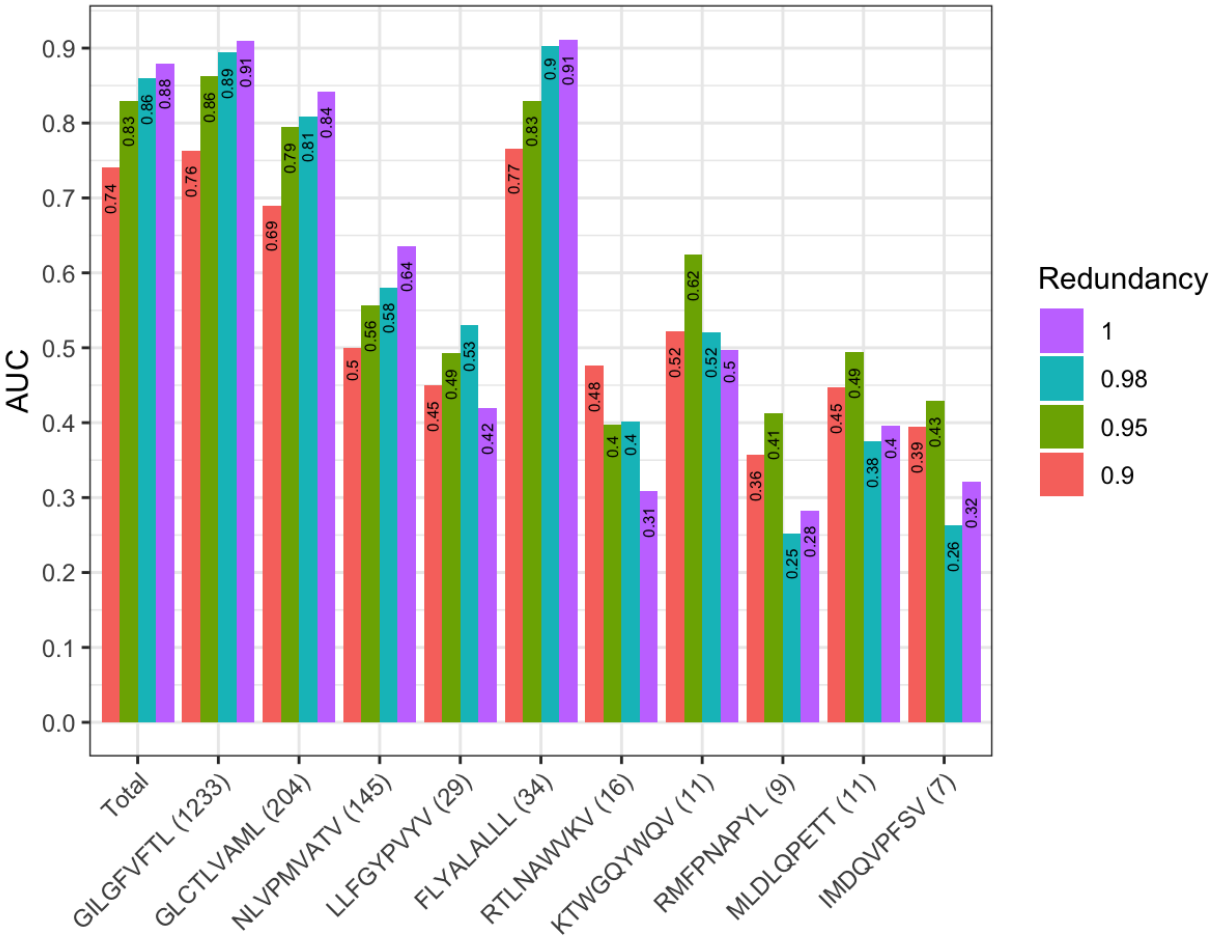


Figure 28: AUCs for each peptide, at each level of redundancy. AUCs were calculated using nested crossvalidation. Number of positive TCRs in the complete dataset is shown in parenthesis.

large, meaning distinguishing between positives and negatives is simpler for the model [17].

While the model obtains great performance, this performance is restricted to only a few peptides. The trend is generally that peptides with a high number of positive sequences obtain a better performance. Therefore, obtaining more data for a larger number of peptides is likely required to make the model better at predicting the peptides with limited data.

3.6.1 Swapped negative performance versus 10x negative performance

The swapped negatives were created to ensure the model learns that TCRs only bind a specific peptide and not all possible peptides. The model could have issues with discriminating between the swapped and positives, leading to low performance on swapped observations. The AUCs for the attLSTM were calculated using all observations, excluding 10x negatives and excluding the swapped observations (Figure 29a). The swapped negatives seem easier

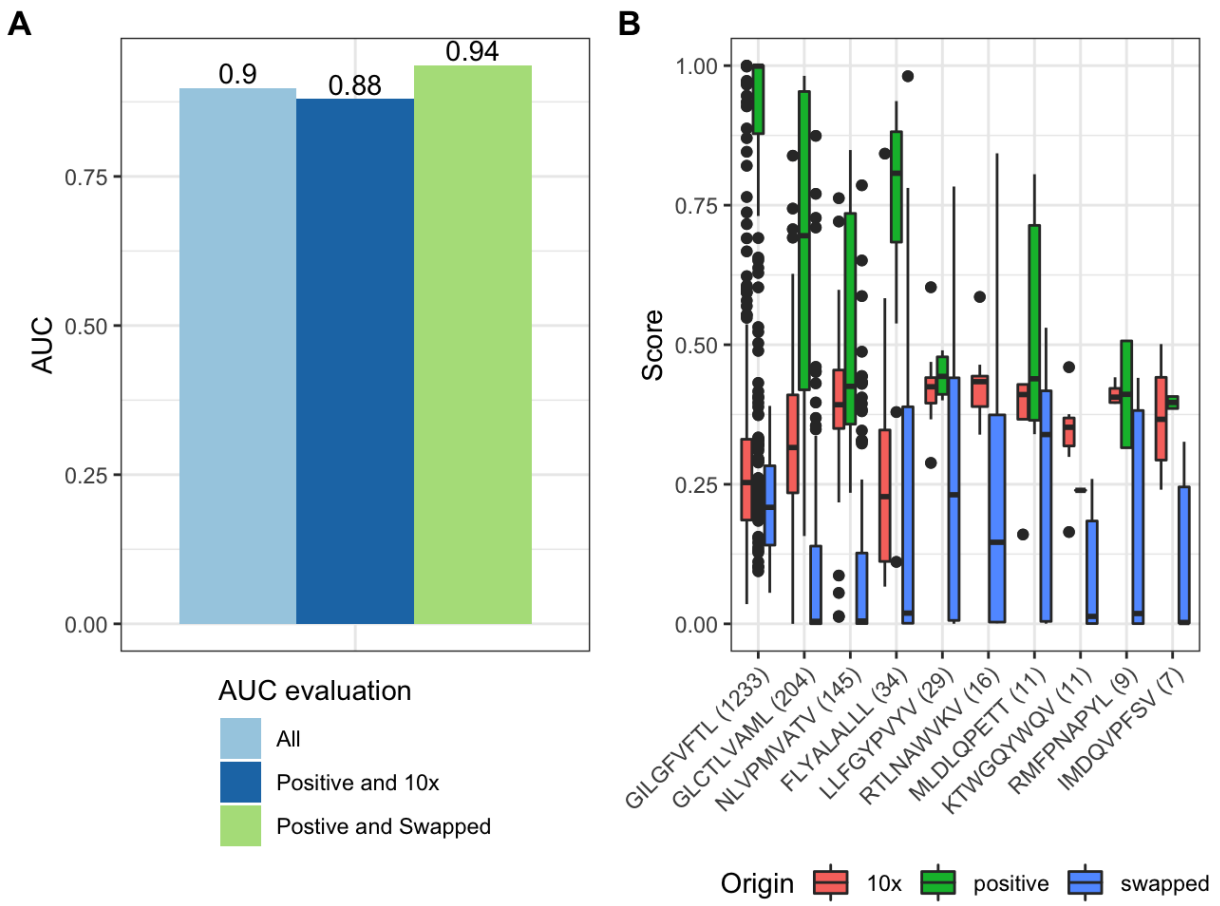


Figure 29: (A) AUCs evaluated on either all data, excluding swapped negatives or excluding 10x negatives.**(B)** output scores for each type of observation. Scores were obtained from a held out partition. Number of positive observations for each peptide is denoted in parenthesis.

for the model to predict, which is counterintuitive considering these TCRs resemble positive TCRs more than a 10x negative.

The output scores for each type of negative are also different (Figure 29b). The swapped negatives generally obtain lower scores than the 10x negatives. Swapped negatives could be easier to predict, as most of the swapped were originally GIL positive. Since the model can pick out GIL positive TCRs, it may have learned that if this type of TCR is observed with another peptide than GIL, it is negative. Additionally, positive observations have high scores for peptides with many observations but low scores for peptides with limited data (Figure 28), again supporting that the model only learns to recognize positive TCRs for a small number of peptides.

3.7 Cross peptide information boosts performance when data is limited

The previously trained models were all pan-specific models, where all peptides were trained and predicted using the same model. In this way, information on all peptides is available for the entire training, and the model can be used to predict all peptides. A peptide-specific model might achieve better performance for individual peptides, considering the model would only have to learn what makes TCRs bind to a single peptide. However, if there is any information, such as binding positions, which are universal for TCRs regardless of peptide specificity a pan-specific model could be better on individual peptides as well.

Th best way of incorporating new peptides were tested by making three types of models. A pan-specific model, which is the general model used in this thesis and is trained on data from all available peptides. The peptide-specific model, which is used to test if there are any benefits from adding data from other peptides, or if a model works better trained on data only from the specific peptide. The last model is considered as a compromise between the two. First a pan-specific model is trained without the peptide in question, similarly to a leave peptide out setting. This model is then used to initialize a peptide-specific model for the peptide left out (i.e. a pan-specific model is trained using all peptides except GIL, and then used to initialize a peptide-specific GIL model). These are the three models used for Figure 30.

Hopefully, the pre-trained peptide-specific model could learn some of the universal aspects of TCR binding from the pan-specific model and then the specifics of that specific peptide, resulting in the best of both models. The pre-trained peptide-specific model simulates the situation where a general pan-specific model could be trained and then fine-tuned to predict a new peptide not currently in the pan-specific model using a small number of observations for this peptide.

The three models mentioned above were trained on a decreasing amount of observations for the peptide in question. All data from other peptides were used for training the pan-specific and for the pre-training of the pre-trained peptide-specific model. This means we observe the drop in performance directly as a consequence of the number of observations for this peptide in the training set, as opposed to Figure 27, where all observations were subsampled.

Figure 30 shows how the performance of a specific peptide is affected by reducing the number of observations for that peptide in the training set. For all models, the performance drops as the number of observations are reduced. The performance on GIL is almost constant with many positives, and all models seem to drop performance around 100-150 positive observations. This drop is not as clear for the other peptides, but the number of positives is also much lower, and the performance drop might not be visible in these plots.

Comparing the individual models on the GIL peptide, the peptide-specific and pan-specific models significantly outperform the pre-trained peptide-specific model (P-value < 0.05 using bootstrapping test with 10000 replications) when the number of positives is high (724 and 550). At a low number of positives (15 and 35), the pre-trained peptide-specific and pan-

specific model significantly outperforms the peptide-specific model (P-value < 0.05 using bootstrapping test with 10000 replications). Therefore, the peptide-specific model seems best for peptides with a high number of positives. In contrast, the pan-specific model (and the pre-trained peptide-specific model) is better when few observations are available for that peptide. At the same time, the peptide-specific models (both the pre-trained and not pre-trained) are more unstable with a low number of observations and could yield spurious predictions.

For the GLC, there is a trend for the two peptide-specific models to have a slight edge compared to the pan-specific models, but no differences were significant. Here the number of observations is also drastically lower, which might explain why the trend from the GIL model is not observed. Lastly, the NLV models have an even lower amount of observations. However, the pre-trained peptide-specific and pan-specific models are significantly better than the peptide-specific model at all numbers of positives, except for 72 positives (P values < 0.05 with bootstrapping test using 10000 replications). This again supports that both the pre-trained peptide-specific and the pan-specific model are better with fewer observations than a peptide-specific model.

3.8 Specificity Predictions

Up till now, the models have only been evaluated on how good they are at predicting whether a given TCR is positive or negative for a given peptide. However, an ideal model should also be able to distinguish between what peptide a given TCR would be bind. In theory, a pan-specific model should be better at this than a peptide-specific model since it has seen information about other peptides. However, Using the pre-trained peptide-specific model, the model has previously seen information about other peptides during pre-training and could perhaps remember this when predicting specificity. A test set was created using all positives from a held-out test set and 10 swapped negatives for each of these positive TCRs. Therefore, the test set has 11 observations for each unique TCR, but only one of which is positive. An ideal model should give the positive a high score, while the swapped should be recognized as negative and given a low score.

An AUC was calculated for each unique TCR using the predictions from each observation. If the positive observation had the highest prediction score, the AUC would be one and zero if all the swapped observations achieved higher scores than the positive. The pan-specific model generally achieves better performance in predicting specificity than the pre-trained peptide-specific model (Figure 31A). As the pan-specific model is trained on all peptides and contains swapped negatives, the model is forced to learn what makes a TCR positive towards GIL, whereas the peptide-specific model only learns what makes a TCR positive.

AUCs for GIL TCRs show that the pan-specific model can almost completely predict the correct peptide, while the pre-trained peptide-specific model tends to predict the correct peptide most of the time, but not as often as the pan-specific model. The difference is even clearer for the GLC peptide, where only a few TCRs are correctly identified as GLC positive for the peptide-specific model. Lastly, the NLV specific TCRs are rarely predicted correctly

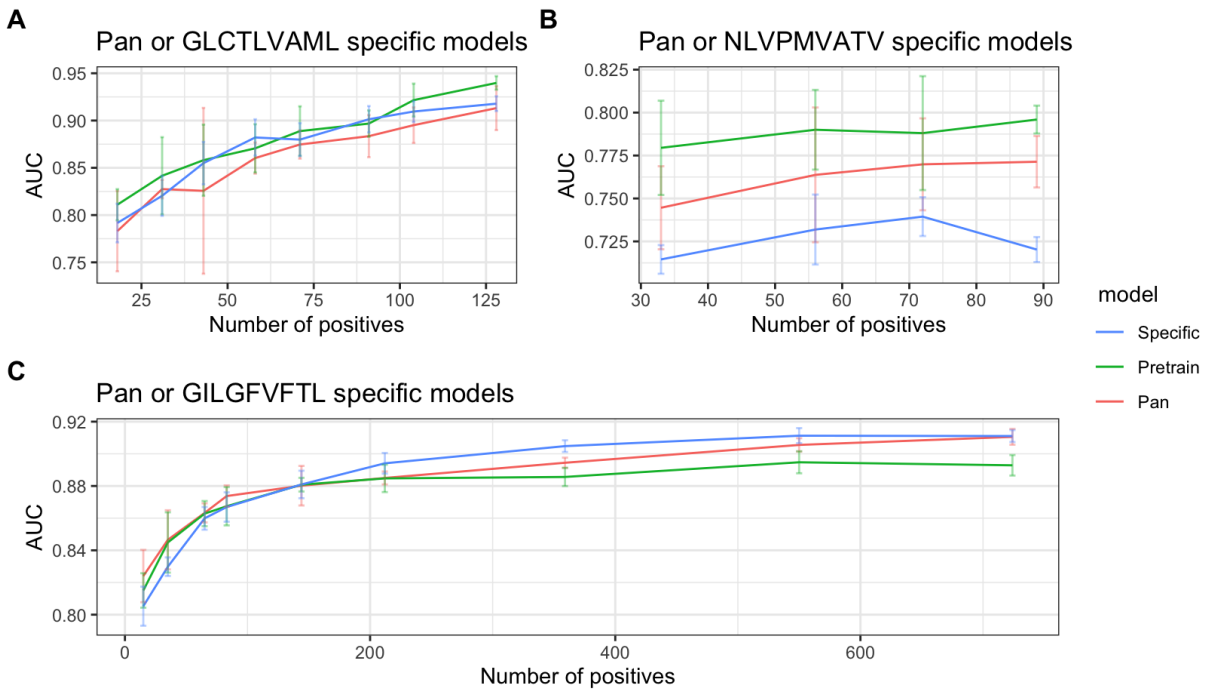


Figure 30: Model performance as a function of the number of positives in the dataset for the three most prevalent peptides in the data. For each of the three peptides, three types of models (peptide specific, pan-specific specific and pre-trained peptide specifc) were trained on varying amounts of data for the specific peptide. The pan-specific model contained a constant amount of data for all other peptides and only varied the peptide in question. The pre-trained peptide specific model was initialized from a pre-trained pan-specific model (trained on the same data as the pan, but without the peptide in question). Lastly the peptide specific model was only trained on the peptide in question. AUCs were calculated from observations with same peptide specificities (**A**) GLC, (**B**) NLV or (**C**) GIL taken from a constant held out partition which was not subsampled. Each model was trained 5 times, average AUC and standard deviation is reported.

using either model, meaning specificity predictions are only really feasible for peptides with a high number of TCRs.

The difference between the two models can be found in their distributions of scores (Figure 31B). The pan-specific model has a much better separation in GIL and GLC scores than the pre-trained peptide-specific model, where the scores seem to distribute almost identically. This supports the idea that the pan-specific model is better at specificity predictions, as it learns to distinguish between peptides. In contrast, a peptide-specific model is not trained with this in mind and only learns what distinguishes a positive and negative TCR for that specific peptide. Therefore, even with pre-training on other peptides, the pre-trained peptide-specific model cannot compete with a pan-specific model when it comes to specificity predictions.

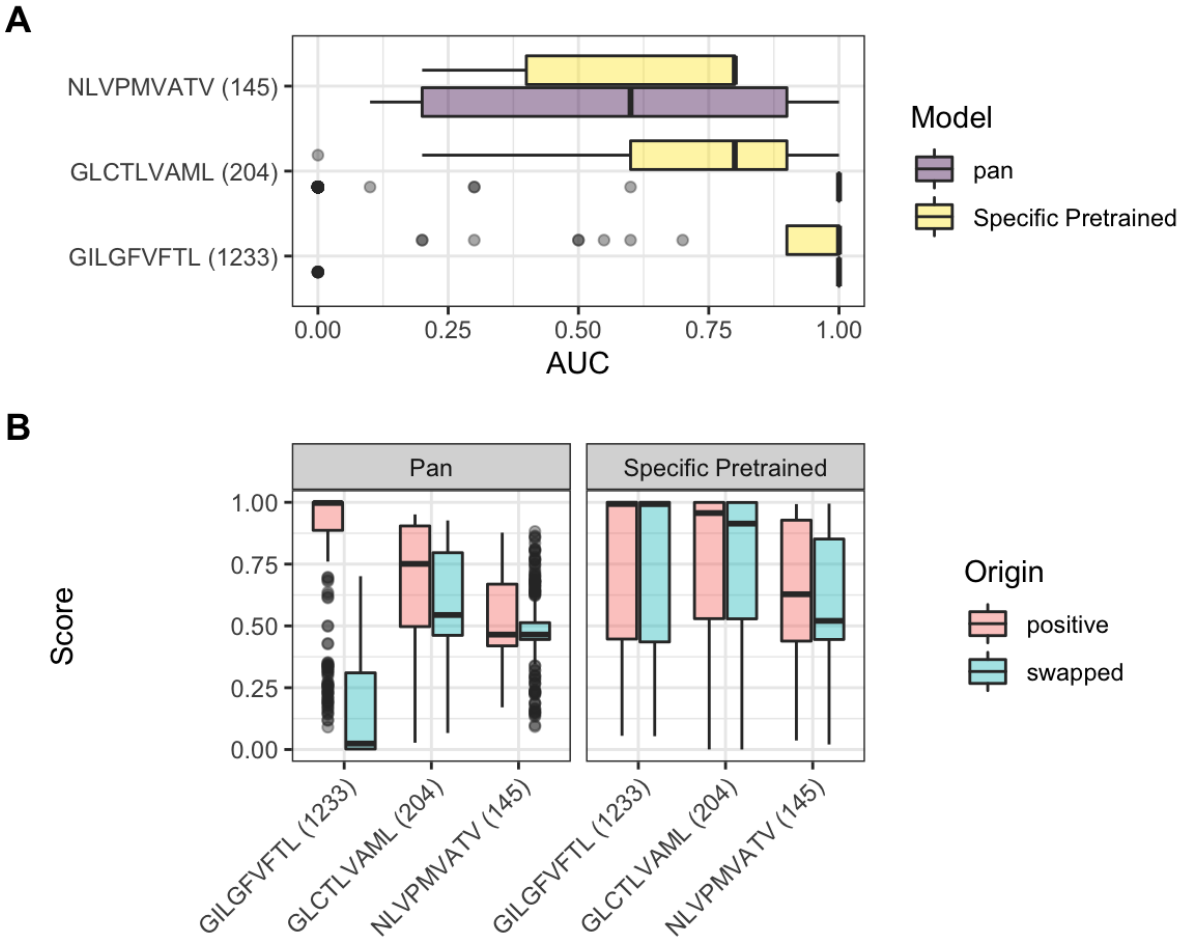


Figure 31: TCR specificity predictions for the 3 most prevalent peptides in the dataset. Number of positives in the dataset is noted in parenthesis. The trained models were evaluated on a test set with all positives from a held out test set, along with 10 randomly sampled swapped negatives created from each positive in the held out set. **(A)** AUCs evaluated for each unique TCR in the testset. **(B)** output scores for binding generated for the testset.

4 Discussion

During this thesis, multiple models have been trained to test different architectures' ability to predict TCR-pMHC interactions, where all neural network models outperformed a sequence similarity-based model. However, all models experienced a severe drop in performance as the redundancy was reduced. Here the LSTM with attention outperformed the other models and managed to retain some performance. The attLSTM achieved a performance of 0.88 across all peptides and up to 0.91 on certain individual peptides. The model focuses on areas containing the largest difference in sequences logos between positive and negative sequences. Attention may therefore explain why the model outperformed the normal LSTM, that is forced to propagate the signal through the entire sequence.

Experimenting with architectures is important when making machine learning models. However, equally important is ensuring the input data is of good quality and relevant for creating predictions. Using features that do not contribute information makes the model larger and more prone to overfit biases in the dataset. By training models and varying the number of features and sequences in the model, we showed that a small model containing a BLOSUM encoding of the peptide and CDR3s was significantly better than a model containing all features available from the dataset.

Lastly, peptide-specific models generally outperformed pan-specific models when a large number of observations were available for that peptide. in situations where training data was limited, the pan-specifc model and the pre-trained peptide-specific model regularly outperformed the peptide-specific model.

For further comparisons, it would be interesting to how the attLSTM model compared to the full NetTCR-2.0 model [17] or another LSTM based model ERGO [29]. The CNN baseline used in this thesis is a simplified model compared to NetTCR-2.0, as the full NetTCR-2.0 contains filters of multiple lengths (1, 3, 5, 7 and 9) as opposed to the simpler model used here with filters of length 3. The full NetTCR-2.0 obtains similar performance in their paper to what is achieved here. However, as the dataset has been modified during this thesis, the comparison is not completely valid, and a benchmarking on the same dataset would be required to compare performances accurately.

4.1 Data biases

The data used for this project have a heavy bias towards specific peptides. This lack of data for other peptides does limit the model's applicability to only a few peptides. A way to solve this issue is to develop tools and models that perform well in cases where few observations for specific peptides are available. Here the pan attLSTM model showed promise by being more resilient to having less training data than other models. Another solution is generating additional data for both the current and all-new peptides, which would bypass the problem by simply having more data available.

By increasing the number of peptides in the data, there would be more peptides with higher

similarity. It is reasonable to assume peptides with high similarity will bind similar TCRs. The model could have an easier time incorporating information from other peptides than currently, as the available peptides might be too dissimilar to provide information helpful in predicting other peptides. Increasing the amount of data will thereby make models better at generalizing and predicting less frequent peptides by leveraging information from other peptides.

The germline genes also contain large biases especially for positive TCRs (Figure 16-17). These biases can occur for different reasons. The experiments conducted to generate the data may introduce technical biases in the data if there is a preference toward TCRs with specific germlines due to the experimental design. The bias could also naturally occur in TCRs binding a specific peptide, meaning the data is representative of the general population of peptides binding that peptide. If this is the case and the model learns to separate the positive and negative TCRs, then the model has learned something general about TCR-pMHC interactions and not just an artefact in the dataset.

Therefore, checking if the observations with the common germlines originate from the same experiment or if multiple experiments find the same germlines to be overrepresented could give some insight into whether this is a technical bias in the data generation or a natural bias in TCRs binding these peptides.

4.2 Introducing additional features to input

Using only CDR3 and peptides gave the best results in the current dataset. CDR3 is the TCR loop closest to the peptide and contains the most sequence variation, but all CDR loops interact with the pMHC complex in some way. However, the information gained about TCR-pMHC binding is limited for CDR1 and CDR2 as they primarily interact with the MHC molecule [6].

The model currently only predicts TCR-pMHC binding for peptides binding the HLA-A02*01. Ideally, models should include predictions for peptides specific to other MHCs. In this setting, there may be a gain in introducing the MHC, CDR1 and CDR2, as they can contribute information specific to interactions between the MHC and TCRs. These types of information are probably not required for the current dataset, as all observations contain the same MHC, and the model does not need to understand these interactions to function.

In the thesis by Meitil, the energy features gave a performance increase in a leave peptide out setting [32]. Here, we saw that all features gave the same performance as only using the sequence in a redundant setting and excluded the energy features for simplicity. The attLSTM has similar properties to the energy features in the thesis by Meitil, as it is superior when data becomes less redundant. The energy features may be able to boost performance further when the amount of data is limited compared to only using the sequence encoding and therefore be helpful on peptides with limited data.

4.3 Capturing multiple sequence motifs

We saw the attention managed to capture one specific motif in the CNN and attLSTM model. In NetTCR-2.0, the model performance is boosted by having multiple filter lengths, which allows the model to capture multiple motifs. In CNNs, this is quite simple as the number of filters can be increased to capture different motifs from the ones already selected by max pooling. In the current form, attLSTM only captures one motif for each CDR3 and peptide. This was especially clear for the CDR3α, where it specifically found a QGN motif in TCRs positive towards GIL.

To extend the current implementation of attention and be able to capture multiple motifs in a sequence, the q (query) vector could be turned into a matrix (Figure 32). Here each row vector would be the high-level representation of a good motif same way a q is in the current implementation (Figure 7). Doing this would increase the output from containing the weighted average according to one motif to have multiple weighted averages and allow the model to capture multiple types of motifs from the same sequence.

Increasing the number of motifs searched for may not give the desired increase in performance on the less frequent peptides in the current dataset. This is because the data is quite biased toward a specific GIL pattern, and there is no guarantee that the model learns motifs for other peptides. Instead, the model might learn other spurious details about GIL to increase performance on the most abundant peptide further. However, only searching for one motif is probably too simplistic, considering the level of variability between TCRs [7].

4.4 Further exploring sequence embeddings

CDR3-specific embeddings did not give the performance boost compared to a BLOSUM encoding. Here we trained the embeddings using a Word2Vec model generating per residue embeddings. Previous studies have also experimented with generating embeddings per 3-mer [53, 54] and found these embeddings to capture physio-chemical properties of the sequences. The 3-mer embeddings used in these papers were created using the same method as here. These motif embeddings could either replace or supplement the per residue embeddings, which could prove superior to only the per residue embeddings used in this thesis.

Within NLP, transformers have gained much attention for their ability within language modelling [20]. The transformer has also led to the BERT model [21], which is capable of creating word embeddings for the English language. The embeddings are learned in a self-supervised (unsupervised) manner and used for downstream applications such as text classification. BERT has also gained traction within protein language modelling with ESM [55] trained on 250 million evolutionary diverse sequences. Transformer based models are all based on the same idea of self-attention, where words/amino acids are predicted from their context. The models are significantly larger than the relatively small Word2Vec model. These more complex embeddings might be a better way to represent amino acids and could provide a performance boost to epitope prediction.

Recently TCR-BERT [31] a CDR3β specific BERT model has been released for predicting

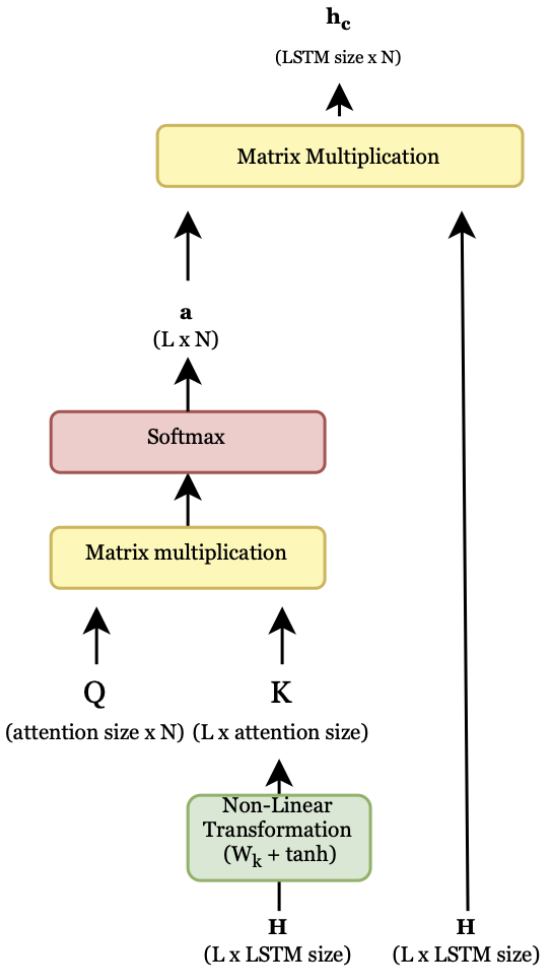


Figure 32: Simple modification on the attention mechanism used in this project (Figure 7). Q is now an N x attention size matrix capable of capturing N motifs instead of one.

TCR specificity. TCR-BERT uses an overall embedding for the CDR3β instead of per residue embeddings. The global CDR3β embeddings are used as input to a PCA-SVM classifier trained to predict a specific peptide. This approach differs from the one taken in this thesis, where we created per residue embeddings and used them to train a neural network classifier. So while a BERT model has been developed for TCR-pMHC prediction, it only utilizes the CDR3β and would probably benefit from incorporating CDR3α as seen in this thesis. Additionally, it would be interesting to see if the global embeddings of the sequences are better for predicting TCR-pMHC binding than training a neural network classifier using sequence embeddings generated from a TCR-specific BERT model.

4.5 Pan specific vs peptide specific models

We saw that pan-specific models generally rivalled the peptide-specific models' performance, especially when data became limited. As paired TCR sequencing is expensive, it is important that models work with small amounts of data. The number of TCRs required for a peptide to obtain meaningful performance needs to be reduced, as the number of potential epitopes is huge and obtaining 100-150 positive TCRs for each is unrealistic. Therefore, the pan-specific model is probably the best-suited model for this purpose. Another clear advantage of the pan-specific model is the convenience of only having one single model and being able to do specificity predictions using the model. Training peptide-specific models for the peptides available now is possible but will become infeasible as data for more peptides becomes available. Therefore, pan-specific models are preferred in the future due to the added stability and performance when the amount of data is limited.

The pre-training of peptide-specific models does rescue some of the issues when only a few observations are available for that peptide. However, the small gain in performance might not justify the increased number of models and losing the ability to make specificity predictions. It is unknown if pre-training a peptide-specific model on a pan-specific dataset even helps with specificity predictions. It would be nice to see whether this was the case, as that could prove the pre-training affects the model's ability to learn peptide specificity.

We saw all models drop in performance when there were fewer than 100-150 positive observations for the GIL peptide. This range was also mentioned by Montemurro et al. as the number of TCRs required to obtain meaningful performance [17]. It is unknown whether this hold for other peptides, as too few positives were available for these peptides to test it.

5 Conclusion

In this project, we have modelled TCR-pMHC interactions by training multiple deep learning models. By leaving out specific sequences, we showed that the CDR3 and peptide sequences are sufficient to obtain performance as good or better compared to using the entire sequence and predicted energy features. The attLSTM also showed an above-average amount of attention put on the CDR3α and CDR3β, further supporting that these sequences contain the largest amount of information for predicting TCR-pMHC interactions.

The attention-based model was able to significantly outperform a similar model not utilizing attention when the amount of data was limited. The attention-based model's success could be because of the increased focus on residues that predominately occurs within positive sequences.

While the network obtains a performance of 0.88, this performance is limited to only four peptides. For the remaining peptides the model only obtains random performance. The generation of additional data should help the model obtain meaningful performance for additional peptides and allow the model to generalize better to unseen data.

Lastly, pan-specific models obtained comparable performance to peptide-specific models and even outperform peptide-specific versions when the number of training peptides was limited. Furthermore, pan-specific models had better performance when used for specificity predictions and only require one model in total, whereas peptide-specific models require one model for each peptide. These facts make pan-specific models preferred, especially in current settings where the number of observations is limited for most peptides.

6 References

[1] Kenneth Murphy et al. *Janeway's Immunobiology*. New York: Garland Science, 2008.

[2] Georg Gasteiger et al. "Cellular Innate Immunity: An Old Game with New Players". In: *Journal of Innate Immunity* 9.2 (Feb. 2017), pp. 111–125. ISSN: 1662-811X. DOI: 10.1159/000453397.

[3] David D. Chaplin. "Overview of the immune response". In: *Journal of Allergy and Clinical Immunology* 125.2 (Feb. 2010), S3–S23. ISSN: 00916749.

[4] Rupert Abele and Robert Tampé. "The ABCs of immunology: Structure and function of TAP, the transporter associated with antigen processing". In: *Physiology* 19.4 (2004), pp. 216–224. ISSN: 15489213. DOI: 10.1152/PHYSIOL.00002.2004/ASSET/IMAGES/LARGE/Y-0002-4-05.JPEG.

[5] Michelle Krogsgaard and Mark M. Davis. "How T cells 'see' antigen". In: *Nature Immunology* 6.3 (2005). ISSN: 15292908. DOI: 10.1038/ni1173.

[6] Mark M. Davis and Pamela J. Bjorkman. "T-cell antigen receptor genes and T-cell recognition". In: *Nature 1988 334:6181* 334.6181 (Aug. 1988), pp. 395–402. ISSN: 1476-4687. DOI: 10.1038/334395a0.

[7] Andrew K. Sewell. "Why must T cells be cross-reactive?" In: *Nature Reviews. Immunology* 12.9 (Sept. 2012), p. 669. ISSN: 14741733. DOI: 10.1038/NRI3279.

[8] Jacob Glanville et al. "Identifying specificity groups in the T cell receptor repertoire". In: *Nature 2017 547:7661* 547.7661 (June 2017), pp. 94–98. ISSN: 1476-4687. DOI: 10.1038/nature22976.

[9] P. Bork, L. Holm, and C. Sander. "The Immunoglobulin Fold: Structural Classification, Sequence Patterns and Common Core". In: *Journal of Molecular Biology* 242.4 (Sept. 1994), pp. 309–320. ISSN: 0022-2836. DOI: 10.1006/JMBI.1994.1582.

[10] Yuko Tsuchiya and Kenji Mizuguchi. "The diversity of H3 loops determines the antigen-binding tendencies of antibody CDR loops". In: *Protein science : a publication of the Protein Society* 25.4 (Apr. 2016), pp. 815–825. ISSN: 1469-896X. DOI: 10.1002/PRO.2874.

[11] Jamie Rossjohn et al. "T Cell Antigen Receptor Recognition of Antigen-Presenting Molecules". In: *Annual Review of Immunology* 33 (Apr. 2015), pp. 169–200. ISSN: 15453278. DOI: 10.1146/ANNUREV-IMMUNOL-032414-112334.

[12] Schrödinger LLC. "The PyMOL Molecular Graphics System, Version 1.8". Nov. 2015.

[13] Jarrett J. Adams et al. "T cell receptor signaling is limited by docking geometry to peptide-Major Histocompatibility Complex". In: *Immunity* 35.5 (Nov. 2011), p. 681. ISSN: 10747613. DOI: 10.1016/J.IMMUNI.2011.09.013.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[15] Vanessa Isabell Jurtz et al. "An introduction to deep learning on biological sequence data: examples and solutions". In: *Bioinformatics* 33.22 (Nov. 2017), pp. 3685–3690. ISSN: 1367-4803. DOI: 10.1093/BIOINFORMATICS/BTX531.

[16] Genta Aoki and Yasubumi Sakakibara. "Convolutional neural networks for classification of alignments of non-coding RNA sequences". In: *Bioinformatics* 34.13 (July 2018), pp. i237–i244. ISSN: 1367-4803. DOI: 10.1093/BIOINFORMATICS/BTY228.

[17] Alessandro Montemurro et al. "NetTCR-2.0 enables accurate prediction of TCR-peptide binding by using paired TCRα and β sequence data". In: *Communications Biology* 4.1 (2021). ISSN: 23993642. DOI: 10.1038/s42003-021-02610-3.

[18] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[19] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings* (Sept. 2014). DOI: 10.48550/arxiv.1409.0473.

[20] Ashish Vaswani et al. "Attention is all you need". In: *Advances in Neural Information Processing Systems*. Vol. 2017-December. 2017.

[21] Jacob Devlin et al. "BERT: Pre-training of deep bidirectional transformers for language understanding". In: *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*. Vol. 1. 2019.

[22] José Juan Almagro Armenteros et al. "DeepLoc: prediction of protein subcellular localization using deep learning". In: *Bioinformatics* 33.21 (Nov. 2017), pp. 3387–3395. ISSN: 1367-4803. DOI: 10.1093/BIOINFORMATICS/BTX431.

[23] Ada Kazi et al. *Current progress of immunoinformatics approach harnessed for cellular- and antibody-dependent vaccine design*. 2018. DOI: 10.1080/20477724.2018.1446773.

[24] Atanas Patronov and Irini Doytchinova. "T-cell epitope vaccine design by immunoinformatics". In: *Open biology* 3.1 (2013). ISSN: 2046-2441. DOI: 10.1098/RSOB.120139.

[25] D. S. DeLuca and R. Blasczyk. *The immunoinformatics of cancer immunotherapy*. 2007. DOI: 10.1111/j.1399-0039.2007.00914.x.

[26] Ilka Hoof et al. "NetMHCpan, a method for MHC class i binding prediction beyond humans". In: *Immunogenetics* 61.1 (2009). ISSN: 00937711. DOI: 10.1007/s00251-008-0341-z.

[27] Birkir Reynisson et al. "NetMHCpan-4.1 and NetMHCIIpan-4.0: Improved predictions of MHC antigen presentation by concurrent motif deconvolution and integration of MS MHC eluted ligand data". In: *Nucleic Acids Research* 48.W1 (2021). ISSN: 13624962. DOI: 10.1093/NAR/GKAA379.

[28] William D. Chronister et al. "TCRMatch: Predicting T-Cell Receptor Specificity Based on Sequence Similarity to Previously Characterized Receptors". In: *Frontiers in Immunology* 12 (Mar. 2021), p. 640725. ISSN: 16643224. DOI: 10.3389/FIMMU.2021.640725/FULL.

[29] Ido Springer et al. "Prediction of Specific TCR-Peptide Binding From Large Dictionaries of TCR-Peptide Pairs". In: *Frontiers in Immunology* 11 (Aug. 2020), p. 1803. ISSN: 16643224. DOI: 10.3389/FIMMU.2020.01803/BIBTEX.

[30] Vanessa Isabell Jurtz et al. "NetTCR: sequence-based prediction of TCR binding to peptide-MHC complexes using convolutional neural networks". In: *bioRxiv* (Oct. 2018), p. 433706. DOI: 10.1101/433706.

[31] Kevin E. Wu et al. "TCR-BERT: learning the grammar of T-cell receptors for flexible antigen- binding analyses". In: *bioRxiv* (2021).

[32] Ida Meitil. *Using deep learning for improving TCR homology modeling and its application to immunogenicity prediction*. 2021.

[33] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958.

[34] S. Henikoff and J. G. Henikoff. "Amino acid substitution matrices from protein blocks". In: *Proceedings of the National Academy of Sciences of the United States of America* 89.22 (1992). ISSN: 00278424. DOI: 10.1073/pnas.89.22.10915.

[35] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013.

[36] Dmitry V. Bagaev et al. "VDJdb in 2019: Database extension, new analysis infrastructure and a T-cell receptor motif compendium". In: *Nucleic Acids Research* 48.D1 (2020). ISSN: 13624962. DOI: 10.1093/nar/gkz874.

[37] Randi Vita et al. "The immune epitope database (IEDB) 3.0". In: *Nucleic Acids Research* 43.D1 (2015). ISSN: 13624962. DOI: 10.1093/nar/gku938.

[38] 10X Genomics. "A New Way of Exploring Immunity - Linking Highly Multiplexed Antigen Recognition to Immune Repertoire and Phenotype". In: *10x Genomics* D (2019), pp. 1–13.

[39] Uwe Hobohm et al. "Selection of representative protein data sets". In: *Protein Science* 1.3 (Mar. 1992), pp. 409–417. ISSN: 1469-896X. DOI: 10.1002/PRO.5560010313.

[40] Michael Schantz Klausen et al. "LYRA, a webserver for lymphocyte receptor structural modeling". In: *Nucleic Acids Research* 43.W1 (2015). ISSN: 13624962. DOI: 10.1093/nar/gkv535.

[41] Kamilla Kjærgaard Jensen et al. "TCRpMHCmodels: Structural modelling of TCR-pMHC class I complexes". In: *Scientific Reports* 9.1 (2019). ISSN: 20452322. DOI: 10.1038/s41598-019-50932-4.

[42] Joost Schymkowitz et al. "The FoldX web server: An online force field". In: *Nucleic Acids Research* 33.SUPPL. 2 (2005). ISSN: 03051048. DOI: 10.1093/nar/gki387.

[43] Rebecca F. Alford et al. "The Rosetta All-Atom Energy Function for Macromolecular Modeling and Design". In: *Journal of Chemical Theory and Computation* 13.6 (2017). ISSN: 15499626. DOI: 10.1021/acs.jctc.7b00125.

[44] Jian Ye et al. "IgBLAST: an immunoglobulin variable domain sequence analysis tool." In: *Nucleic acids research* 41.Web Server issue (2013). ISSN: 13624962. DOI: 10.1093/nar/gkt382.

[45] Wen-Jun Shen et al. *Towards a Mathematical Foundation of Immunology and Amino Acid Chains*. May 2012. DOI: 10.48550/arxiv.1205.6031.

[46] Radim Rehurek and Petr Sojka. "Gensim–python framework for vector space modelling". In: *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic* 3.2 (2011).

[47] Alex Bateman et al. "UniProt: the universal protein knowledgebase in 2021". In: *Nucleic Acids Research* 49.D1 (Jan. 2021), pp. D480–D489. ISSN: 0305-1048. DOI: 10.1093/NAR/GKAA1100.

[48] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019.

[49] Diederik P Kingma and Jimmy Lei Ba. *ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION*. Tech. rep.

[50] Kaiming He et al. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. 2015. DOI: 10.48550/ARXIV.1502.01852.

[51] Wing Ki Wong, Jinwoo Leem, and Charlotte M Deane. "Comparative Analysis of the CDR Loops of Antigen Receptors". In: *Front. Immunol* 10 (2019), p. 2454. DOI: 10.3389/fimmu.2019.02454.

[52] Martin Christen Frolund Thomsen and Morten Nielsen. "Seq2Logo: a method for construction and visualization of amino acid binding motifs and sequence profiles including sequence weighting, pseudo counts and two-sided representation of amino acid enrichment and depletion". In: *Nucleic acids research* 40.Web Server issue (July 2012). ISSN: 1362-4962. DOI: 10.1093/NAR/GKS469.

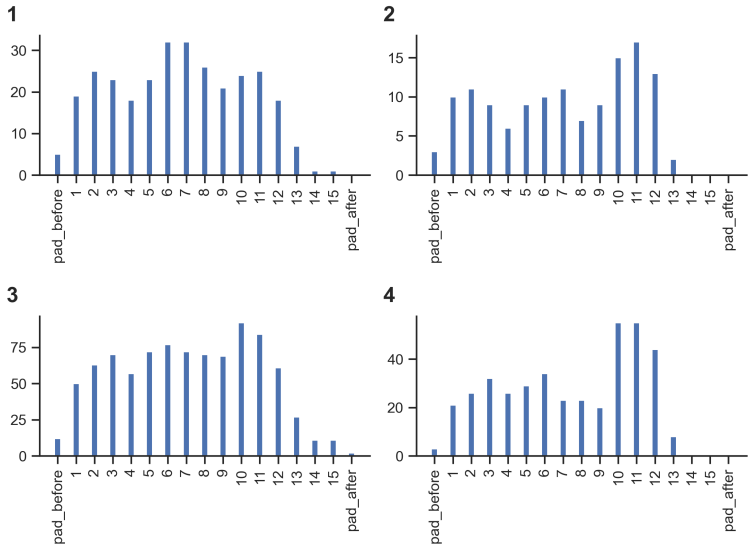
[53] Ehsaneddin Asgari and Mohammad R.K. Mofrad. "Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics". In: *PLOS ONE* 10.11 (Nov. 2015), e0141287. ISSN: 1932-6203. DOI: 10.1371/JOURNAL.PONE.0141287.

[54] Kevin K. Yang et al. "Learned protein embeddings for machine learning". In: *Bioinformatics* 34.15 (Aug. 2018), p. 2642. ISSN: 14602059. DOI: 10.1093/BIOINFORMATICS/BTY178.

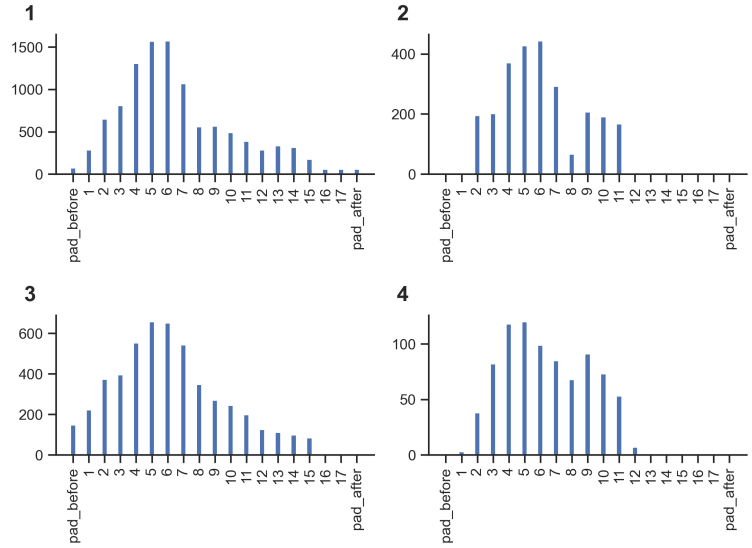
[55] Alexander Rives et al. "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences". In: *Proceedings of the National Academy of Sciences* 118.15 (2021), e2016239118. DOI: 10.1073/pnas.2016239118.

7 Appendix

Appendix A - Maxpool positions for TCRs with NLV specificity



(a) Histograms for CDR3α.



(b) Histograms for CDR3β.

Figure 33: Histograms of number of times a position is selected to contribute information by the maxpooling operation. Positions are determined as contributing if it is used in the convolution selected by maxpooling. **(1)** Positions selected for CDR3s with positive interaction. **(2)** Only considering filters with an activation above 0.95 **(3)** Positions selected for CDR3s with negative interaction. **(4)** Only considering filters with an activation above 0.95 Data shown for partition held out during training limited to CDR3s with length 12 and specificity for NLV.

Appendix B - BiLSTM attention for reverse direction

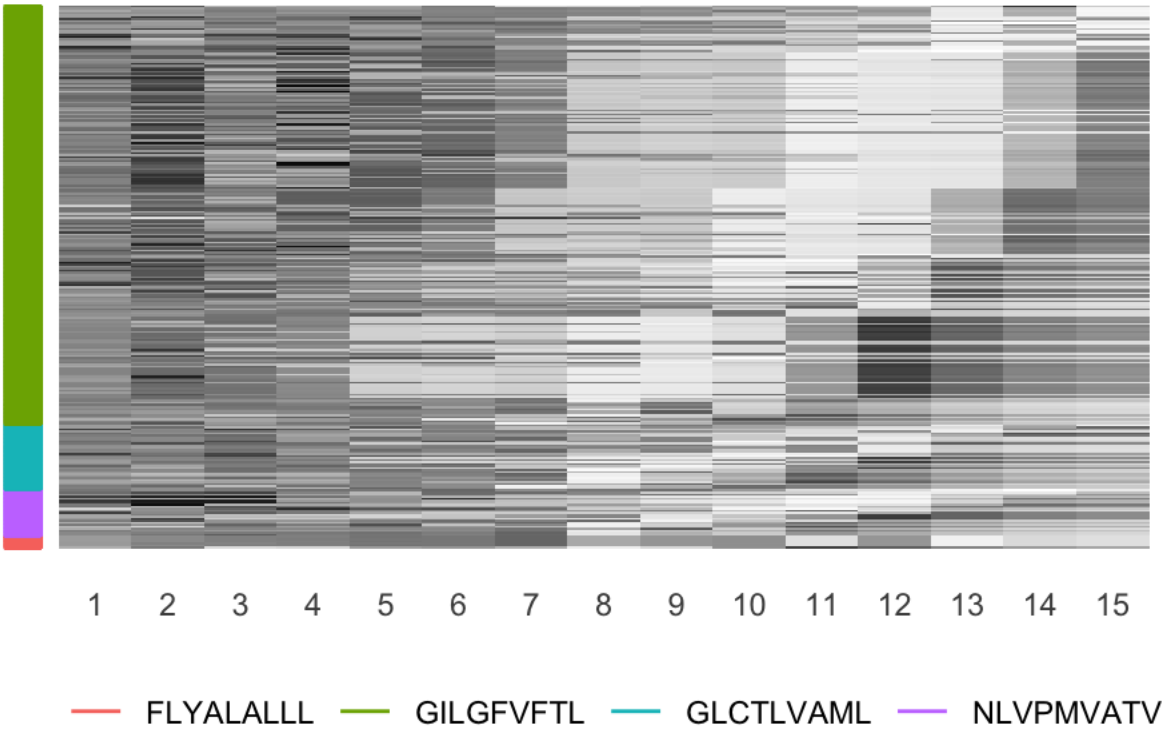


Figure 34: attention on positions for generating the weighted hidden state for the reverse direction of the CDR3α. Sequences have been sorted by CDR length in descending order. Data shown for partition held out during training.

Appendix C - Sequence logo for GILGFVFTL positive CDRα with length 13 and 11

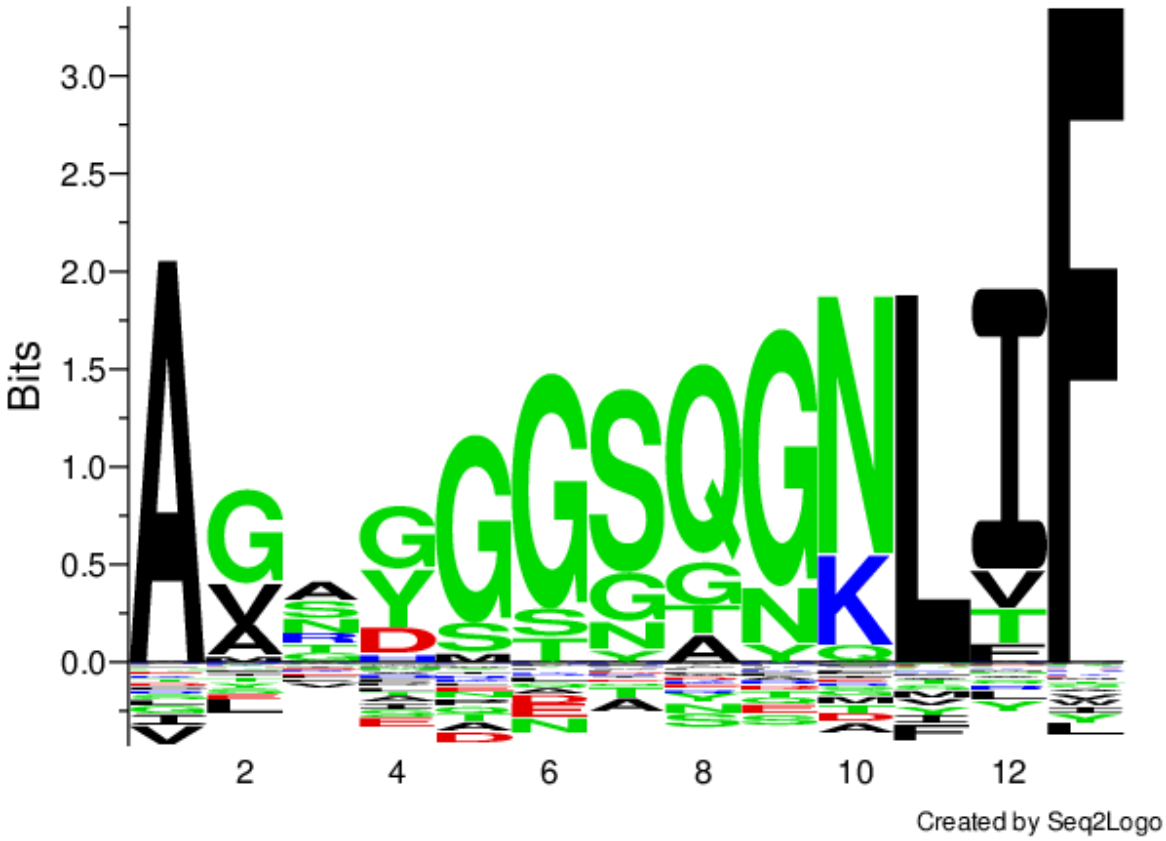


Figure 35: Logo plot for GILGFVFTL positive CDR3α with a lenght of 13. The sequence contains the same motif as found in 24a, but at one position earlier as expected.

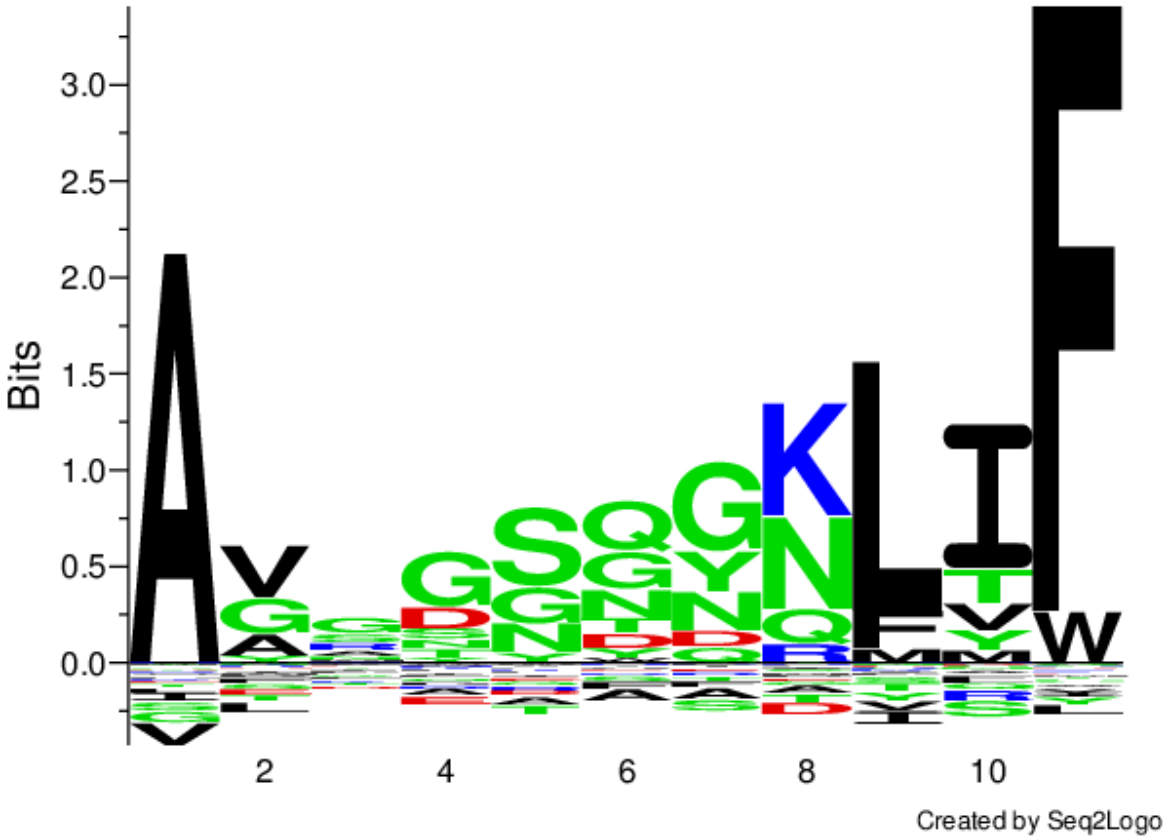


Figure 36: Logo plot for GILGFVFTL positive CDR3α with a lenght of 11. The sequence contains the same motif as found in 24a, but at position 7 and 8.