TECHNICAL UNIVERSITY OF DENMARK

Using deep learning for improving TCR homology modeling and its application to immunogenicity prediction

Master Thesis

MSc student Biotechnology
Ida Kristine Sandford Meitil
s153020

Supervisor:
Associate Professor Paolo Marcatili

Technical University of Denmark
Department of Health Technology
AI for Immunological Molecules group
Kgs. Lyngby, Denmark

June 4, 2021

Preface

This is the written work of the Master's thesis ”Using deep learning for improving TCR homology modeling and its application to immunogenecity prediction”. The thesis was conducted at the AI for Immunological Molecules group at DTU from 4th of January to 4th of June 2021. The thesis corresponds to 30 ECTS points.

I would like to thank professor Paolo Marcatili for the valuable supervision. I am very grateful for all the time you have taken to discuss and guide me in the right direction.

I would also like to express my gratitude to Anna-Lisa Schaap-Johansen for patiently guiding me in building the deep learning framework for the project. I would also like to thank Magnus Haraldson Høie for idea sharing and good discussions.

Data, scripts and results from the project are available at https://github.com/idameitil/master-thesis.

Abstract

The binding of a T-cell receptor to a peptide-MHC-I is a key step in the adaptive immune system. Being able to computationally predict the target of a T-cell receptor would be a major advancement in the research of cancer immunotherapy and cancer vaccines.

In this project, we develop a prediction tool for TCR-pMHC binding using molecular modelling and force field energies in a deep learning framework. A benchmarking dataset is constructed by modelling $\sim$10,000 binding and non-binding TCR-pMHC complexes. We use two different force fields, FoldX and Rosetta, to relax and score the models. In Rosetta, we score them both globally and per-residue and we include both the total score and the breakdown into individual energy terms. A CNN-biLSTM network is built that predicts binding based on the amino acid sequence and the energy terms.

We show that including energy terms substantially improves the model's ability to generalize compared to using the sequence alone. While the sequence alone shows very good predictive power in a nested cross-validation setup, it completely fails in a leave-one-out setup, showing that it is largely overfitting. Including the energy terms, on the other hand, makes the model much more stable and able to predict peptides that it has not been trained on.

We further show that including Rosetta per-residue energies makes the model perform better compared to using only global energy terms.

The performance of the model is limited by the amount and diversity of training data, but it is reasonable to expect that as more diverse data becomes available in the future, the model will improve with respects to its ability to generalize.

Contents

Acronyms

ANN Artificial neural network.

AUC Area under the (ROC) curve.

CDR Complementarity determining region.

CNN Convolutional neural network.

FN False Negatives.

FNN Feedforward neural network.

FP False Positives.

FPR False Positive Rate.

HLA Human leukocyte antigen.

LSTM Long short-term memory.

MCC Matthews correlation coefficient.

MHC Major Histocompatibility Complex.

RNN Recurrent neural network.

ROC Receiver operating characteristic.

TCR T-cell receptor.

TN True Negatives.

TP True Positives.

TPR True Positive Rate.

List of Figures

List of Tables

1 Introduction

T-cells play a central role in protecting the body against infectious diseases as well as cancer. Being able to accurately predict the target of a T-cell receptor would be a big step forward in the research on cancer T-cell therapy and T-cell vaccines. The focus of this project is to make a deep learning model that can make such predictions.

In the following sections we will give a brief introduction to the immune system, immunoinformatics, the state of the art of TCR-pMHC prediction models, and finally the scope of this project.

1.1 The immune system

This section gives a brief introduction to the human immune system and is mainly based on chapter 1, 3 and 4 of the book ”Basic Immunology: Functions and Disorders of the Immune System” by Abul Abbas et al. [1].

The immune system is the defense guard of the body, which protects against disease, for example from pathogens or cancer cells. The immune system needs to be able to distinguish healthy cells (self) from sick cells, which can be a difficult balance. An under-active immune system can leave the body vulnerable to infections and cancer. An over-active immune system, on the other hand, can lead to allergies and auto-immune diseases.

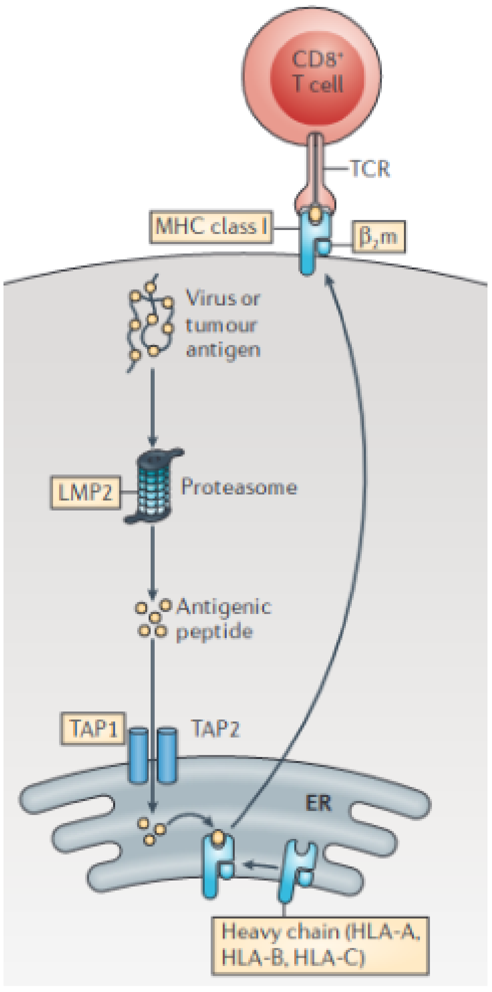
The immune system can be divided into the innate and the adaptive immune system. The innate immune system is fast, but non-specific and less potent, whereas the adaptive immune system is slow, specific and more potent. The cells of the immune system are the white blood cells, also called leukocytes.

The innate immune system is present since birth. The first line of defense are the outer barriers, i.e. the skin, respiratory-, digestive-, urinary-, reproductive- and mucosal surfaces. The second line of defense is the cell-mediated response. The cells responsible for the cell-mediated response in the innate immune system are a subgroup of leukocytes called phagocytes. Some types of phagocytes are responsible for recognizing and eliminating pathogens, others process antigens and present them on the cell surface for recognition by T-cells. The innate immune system is also responsible for producing cytokines that start an inflammatory response.

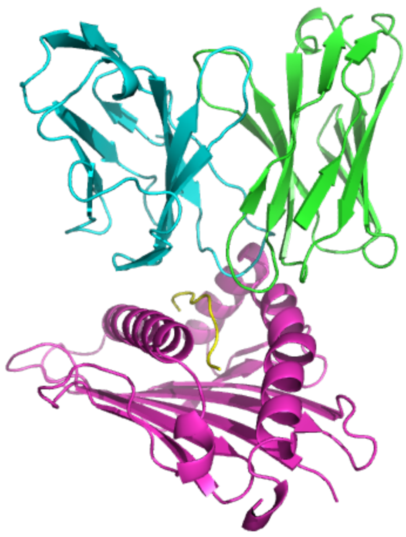
The adaptive immune system is tailored to adapt to specific diseases and to remember their antigens in case of a future infection with the same pathogen. The cells responsible for the adaptive immune system are a special kind of leukocytes called lymphocytes. The two main kinds of lymphocytes are B-cells and T-cells, which both have immunoglobulin receptors that recognize antigens: B-cell receptors (BCR) and T-cell receptor (TCR).

The B-cells are part of the humoral response and their receptors bind to unprocessed antigens, which can be all kinds of molecules, such as proteins, lipids and polysaccharides. Upon binding to an antigen, B-cells can be activated and produce many copies of the antibodies that will circulate in the blood and mark the cells for destruction.

T-cells are involved in the cell-mediated immune response. T-cell receptors recognize peptide antigens that have been processed and are presented by antigen presenting cells by the Major Histocompatibility Complex (MHC).



(a)



(b)

Figure 1: a) CD8+ T-cell recognition of an infected cell. Proteins in the cell are being degraded and some of these will bind to MHC-I in the ER, which will then be presented to T-cells on the cell surface. If a CD8+ T-cell binds to it, it will release cytotoxins that kill the cell. Figure adapted from [2]. b) Chemical structure of a TCR-pMHC-I complex. TCRα in green, the TCRβ in blue, peptide in yellow, MHC in purple.

The cytotoxic T-cells, also known as CD8+ T-cells, recognize antigens presented by class-I MHC (figure 1). In the cell, the proteasome constantly chops proteins into smaller peptides. Some of these peptides are transported by TAP (transporter associated with antigen processing) to the endoplasmic reticulum. Some of these peptides will here bind to MHC-I and form a peptide-MHC complex (pMHC), which will then be transported to the plasma membrane. If a T-cell receptor binds to it (forming a TCRpMHC complex), the T-cell releases cytotoxins that kill the cell.

Helper T-cells, also known as CD4+ bind to antigens presented by class-II MHC. Upon binding, the helper T-cell releases cytokines that regulate many cells; they activate antibody production by B-cells and enhance the function of cytotoxic T-cells and phagocytes. In this project we will not focus on CD4+ T cells.

When T- and B-cells are activated, they both produce long-lived offspring: memory cells. The memory cells will remember the specific antigen and can start a fast and strong response if the body encounters that pathogen again later in life.

1.1.1 Major histocompatibility complex

As already mentioned, the major hisotcompatibility complex is the protein complex that presents peptide antigens to T-cells. This project focuses on class-I MHC (MHC-I), which presents peptides to cytotoxic T-cells. MHC-I is present on all nucleated cells.

MHC-I is a heterodimer consisting of an α-chain and a β-2-microglobulin. The α-chain has three domains, α1, α2 and α3. α3 is attached to the membrane and is conserved. α1 and α2 are highly polymorphic and form the binding cleft for the peptide and the binding site for the T-cell receptor.

Human MHCs are called Human leukocyte antigen (HLA), and there are three groups of MHC-I: HLA-A, HLA-B and HLA-C, each group having thousands of different alleles.

1.1.2 T-cell receptor

The T-cell receptor is the surface protein on T-cells that bind to MHC. It is a heterodimer consisting of an α- and a β-chain linked together by a disulfide bridge (figure 1b). A minority of TCRs consist of a γ- and a δ-chain, but they are not relevant for this project. The α- and β-chain both consist of a constant domain located closest to the membrane and a variable domain located furthest from the membrane. The variable domain of each chain contains three Complementarity determining regions (CDRs): CDR1, CDR2 and CDR3 (figure 2). These regions are highly variable and compose the antigen binding site. The CDR3 is the most variable of the loops, and is also the most important for binding to the peptide antigen. Both the α- and the β-chain participate in the recognition and binding of a pMHC.

Most of the framework of the TCR is thus very conserved and the majority of the variety lies in the CDR regions. Although CDRs have a large sequence diversity, they have been shown to assume only a limited number of structures, named the canonical structures, which are determined by key residues in certain positions [4, 5]. This property makes it relatively easy to predict the structure of TCRs, which we will return to later.

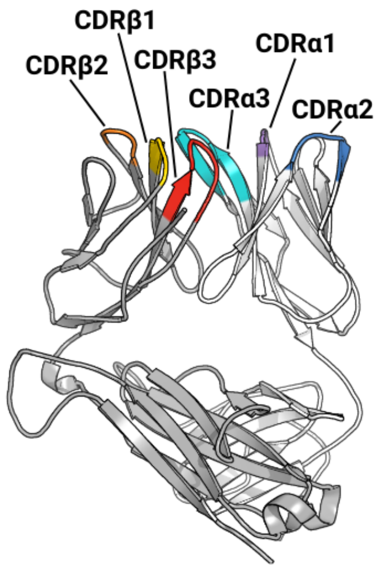


Figure 2: Complementarity determing regions (CDRs) of the T-cell receptor. The CDR3β and CDR3α are the most important for binding to pMHC. Figure adapted from [3]

The diversity of the TCRs arises from a process called V(D)J recombination that occurs in the maturation of T-cells. The β-chain is encoded by the Variable (V), Diversity (D), and Joining (J) genes, and the α-chain is encoded by the V and J gene. These gene segments are rearranged randomly, resulting in a high diversity of receptors being able to recognize many different antigens.

It is important neither to have T-cells that do not bind to any pMHCs, nor to have T-cells that bind too strongly to self pMHC. This is ensured by the processes of positive and negative selection. In positive selection, T-cells in the thymus are presented to self-antigens presented by MHCs, and only those that bind will survive. In negative selection, T-cells that bind too strongly to a pMHC are killed.

1.2 Applications of TCR-pMHC predictions

Understanding the rules of binding between a TCR and pMHC is fundamental for our understanding of immunogenicity. With the millions of different TCRs and pMHCs, it is very extensive to experimentally determine the target of a T-cell receptor. Thus, being able to computationally determine this binding and estimate the affinity could provide a starting point for potential targets that can then be screened experimentally.

One research area that could benefit from such a tool is cancer immunotherapy. One form of immunotherapy that is showing good effects is where tumor-infiltrating lymphocytes (TILs) from the patient are expanded ex vivo and reintroduced into the patient [6]. However, not all tumors contain TILs that are effective against the tumor. Another approach is therefore to use genetically engineered T-cells, where T-cells are taken from the blood of the cancer patient and modified with genes that encode for cancer-specific antigen receptors [7, 8]. The challenge is to find TCRs that are specific for the relevant pMHC. Cancer antigens are difficult for the immune system to detect, because they are often similar to human-self antigens. Being able to predict a repertoire of potential TCRs for the target, would potentially improve the therapy and save money and time from the laboratory work.

Another application for TCR-pMHC binding predictions is peptide T-cell vaccines against cancer. Vaccination would be a simpler and more cost-effective therapy than gene-engineered T-cell therapy. However, the attempts so far have yielded poor results [9]. The development of peptide T-cell vaccines relies on identifying tumor antigens. For this application, a computational TCR-pMHC prediction model could serve as a tool to screen for potential antigens.

Other applications are TCR peptide vaccines against allergies and autoimmune diseases such as multiple sclerosis, rheumatoid arthritis and psoriasis [10, 11].

1.3 Immunoinformatics

In this section, we will go through some of the methods of immunoinformatics that are used in this project.

1.3.1 Protein modelling

Knowing the structure of a protein is fundamental for understanding its function. The Protein Data Bank contains about 150,000 protein structures [12], while the Genbank contains about 200 million protein sequences, a number that doubles approximately every 18 months [13]. Thus, if we can predict the structure of proteins we have access to a much bigger amount of data.

Methods for secondary structure predictions include using statistical propensities of amino acids towards specific secondary structure elements, using a sliding window of neighbouring residues, using evolutionary information and deep learning [14]. Secondary structure predictions are now approaching the theoretical limit of accuracy [14].

Tertiary and quaternary structure prediction are more complicated tasks and can be done either by homology modelling or ab initio. In homology modelling, the protein is structurally aligned to a template structure. Homology modelling is very effective and computationally fast, but it is dependent on having a template with a high sequence similarity. A widely used tool for homology modelling is MODELLER [15].

Ab initio modelling builds a model without any template and is useful when no structural homologs are available, but is often very computationally demanding. Methods for ab initio modelling are very diverse, but often use a scoring function and a method for conformational sampling [16].

1.3.2 Force fields

Calculating the potential energy of a protein structure can be useful for indicating how stable it is. A chemical force field is a set of functions and their parameters used to calculate the potential energy of a chemical structure. The two force fields used in this project are FoldX [17, 18] and the Rosetta Energy Function 2015 [19, 20].

FoldX is an empirical force field with the main functionality to calculate the free energy of a macromolecule based on its structure [18], and has been shown to be good at predicting the impact of a mutation in a protein [21].

The Rosetta force field is also an empirical force field and has been widely used for protein modelling and molecular dynamics simulations [20]. Rosetta calculates the energy of a molecule or complex as a weighted sum of different individual energy terms:

$$\Delta E_{total} = \sum_{i} w_i E_i \tag{1}$$

1.3.3 Machine learning

The prediction of binding between two proteins is a very complex task that involves many different parameters. For this task, machine learning methods can be helpful, because of their ability to find patterns in complex data.

Machine learning is a method of data analysis, where a computer automatically builds a model based on training data. The foundation of modern machine learning is artifical neural networks (ANNs). Artificial neural networks mimic biological neural networks and consist of nodes called artificial neurons that pass on information between each other and converts an input to an output.

The simplest version of an ANN is a Feedforward neural network (FNN). In an FNN, the information moves only forward, i.e. from the input layer, through a number of hidden layers and to the output layer (figure 3). All layers are fully connected meaning that every neuron is connected to all neurons in the previous and the subsequent layer.

The process of calculating the output from the input is called the forward pass. In the forward pass, for a given neuron, the neurons in the previous layer have their values multiplied to their weights. They are then all summed and added a bias. An activation function such as sigmoid or sofmax is then applied to the value. This procedure is continued through all the layers.

In order to train the network, the weights and biases need to be adjusted based on the difference between the predicted values and the target values. This is done through the process of backpropagation. First, the loss is calculated via a loss function, such as cross-entropy loss for classification tasks. The weights are then adjusted using an optimizer function, most often a gradient descent optimizer, such as stochastic gradient descent (SGD) or Adaptive Moment Estimation (Adam). The process of forward passing and backpropagation is repeated for a certain amount of epochs, decreasing the training loss for each epoch.

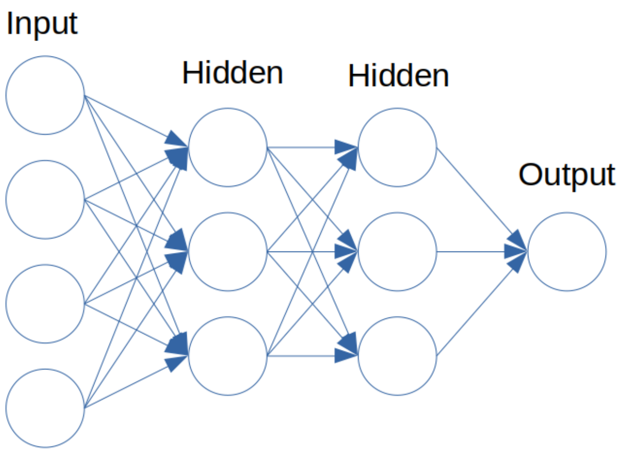


Figure 3: Representation of a feedforward neural network. Each circle represents a neuron.

1.3.3.1 Convolutional neural networks

Convolutional neural networks (CNN) are a type of neural networks which are useful for analysing data of sequential nature. Common applications for CCNs are image analysis and natural language processing, but they have also been shown to be very suited for finding motifs in protein sequences [22, 23, 24].

CNNs consist of alternating convolutional and pooling layers (figure 4). In the convolutional layer, a kernel or filter slides along the sequence calculating the convolution. This generates a feature map. Then, in the pooling layer, the dimensions are reduced. A common type of pooling is max pooling which takes the maximum value of the neurons in the feature map cluster. A common activation function for convolutional layers is Rectified Linear Unit (ReLu).

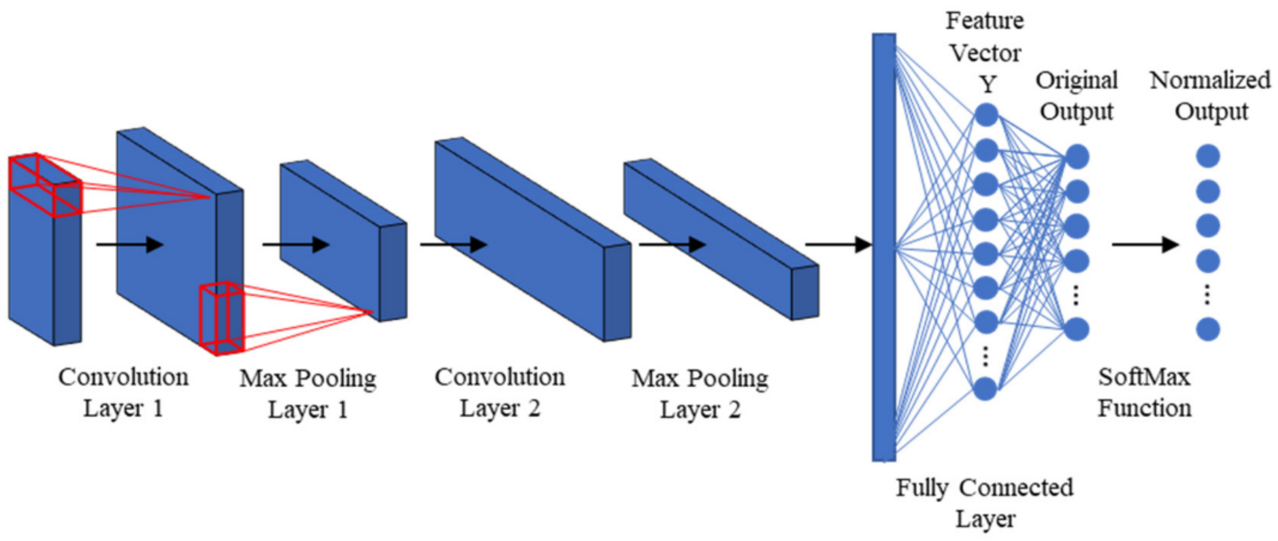


Figure 4: Schematics of a Convolutional neural network. Adapted from [25].

During training, the filters are trained to learn specific patterns in the sequence. In the context of protein sequences, the filter can for example learn to recognize a certain motif which is important for binding. The filter size will determine the size of the pattern that is being detected.

After a number of alternating convolution and pooling layers, the resulting feature vector goes through a fully connected layer and outputs the predictions.

1.3.3.2 Long short-term memory networks

While CNNs are good at detecting local patterns, they often fail when it comes to learning long-range interactions. In the context of predicting TCR-pMHC complexes, it is necessary to detect interactions between certain residues in the TCR and residues in the peptide, which may be far away from each other in the sequence. For this purpose, recurrent neural networks (RNN) and in particular Long short-term memory networks (LSTMs) are well suited.

Recurrent neural networks have loops that allow information to pass from one neuron to the next which makes them suitable to process sequences. However, simple RNNs often come short when it comes to remembering positions far back in the sequence (long-term dependencies). For this purpose, LSTM networks are very useful. LSTMs were first introduced by Hochreiter and Schmidhuber in 1997 [26], and they are designed to remember information for a long period of time.

LSTMs consist of a chain of repeating modules. The most simple LSTM modules consist of a cell state which receives information from a forget gate, an input gate and an output gate, but many variants of LSTMs exist. A special kind of LSTM is the bidirectional LSTM (biLSTM), which consists of two LSTMs: one that takes the input data in a forward direction and one that takes the input in a backward direction (figure 5). BiLSTMs can often improve the performance, since they also take into account what comes after the current position.

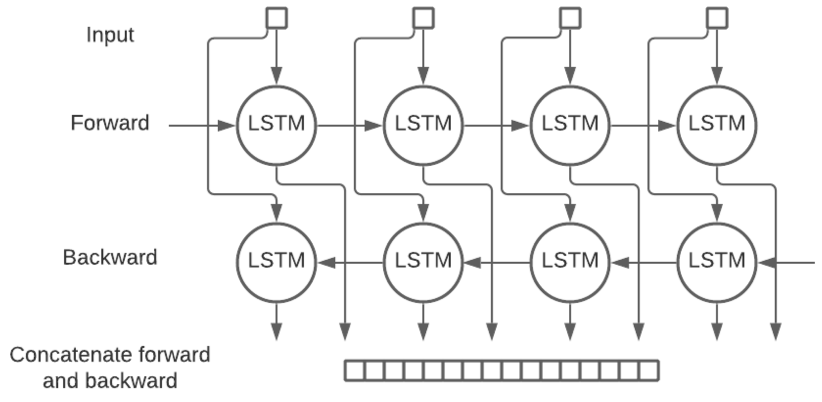


Figure 5: Schematic of a biLSTM. The sequence is being processed in a forward and a backward direction and the outputs are concatenated.

1.3.3.3 Overfitting

A common problem in machine learning is overfitting. Overfitting occurs when the model starts to learn not only the signal, but also the noise in the training data. It will thus not be able to generalize and will perform poorly on new data.

One tool for avoiding overfitting is early stopping. Normally, the performance on the training set will keep improving if we keep training. On the other hand, the performance on the validation set will at some point start to level off and eventually decrease. In early stopping, the training is stopped when the loss on the validation set has not decreased for a given number of epochs.

Another way to prevent overfitting is by using regularization. The principle of regularization is to add a penalty when the complexity of the model increases. There are several types of regularization, but they usually include adding a penalty to the loss function for each parameter in the model.

A way to detect overfitting is by using cross-validation. A commonly used type of cross-validation is k-fold cross-validation, where the dataset is partitioned into k partitions. One partition is retained for validation and the remaining samples are used for training. This is repeated k times, each time using a new partition for validation.

In this project, we use nested cross-validation (figure 6). In this setup, the dataset is split into 5 partitions and in an outer loop, one by one, a set is selected as test set. Then, in an inner loop, one by one, a set is selected out of the remaining 4 sets as validation set. The model is then trained on the remaining 3 sets using the validation set for early stopping and the test set for testing. This results in 20 models, and their test predictions are then concatenated for performance evaluation.

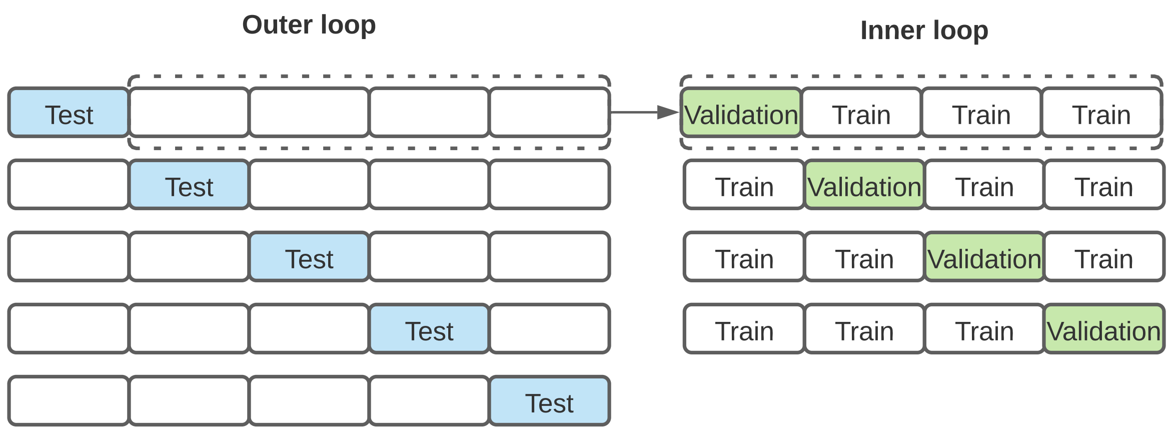

Outer loop
Inner loop
Test
Test
Test
Test
Test
Validation Train Train Train
Train Validation Train Train
Train Train Validation Train
Train Train Train Validation

Figure 6: Nested cross-validation as used in this project. In the outer loop one-by-one a testset is selected. In the inner loop, one by one a validation set is selected from the four remaining sets and the model is trained on the three remaining sets. The predictions from the 20 models are concatenated and the performance is evaluation.

1.3.3.4 Performance measures

This section provides an overview of the performance measures used in the project. The performance measures are based on the number of True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN), which can be illustrated in the confusion matrix (table 1).

	Predicted negative	Predicted positive
Actual negative	True Negative (TN)	False Positive (FP)
Actual positive	False Negative (FN)	True positive (TP)

Table 1: Definition of the confusion matrix as used in this project.

Accuracy is the fraction of correct predictions among all the predictions:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{2}$$

Recall, also known as the True Positive Rate (TPR) or sensitivity is the fraction of positives that are correctly predicted:

$$Recall = \frac{TP}{TP+FN} \tag{3}$$

Precision is the fraction of True Positives among the predicted positives:

$$Precision = \frac{TP}{TP+FP} \tag{4}$$

The F1 score is the harmonic mean of the precision and the recall:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{5}$$

A measure that is useful for unbalanced datasets is the Matthews correlation coefficient (MCC), defined as:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{6}$$

The MCC can take on values between -1 and 1; 1 meaning perfect predictions, 0 meaning random predictions and -1 meaning total disagreement between predictions and actual classes.

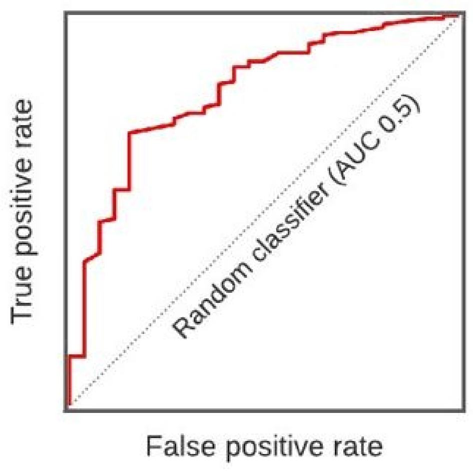


Figure 7: The receiver operating characteristic (ROC) curve. The ROC curve depicts the TPR against the FPR for varying thresholds. The area under the ROC curve (AUC) shows how good a classifier is. An AUC of 0.5 indicates a random classifier and an AUC of 1 indicates a perfect classifier.

Finally, we use the Receiver operating characteristic (ROC) curve and the area under the ROC curve (AUC). The ROC curve depicts the True Positive Rate against the False Positive Rate for varying thresholds of a binary classifier (figure 7). The AUC is the area under the ROC curve and thus an AUC of 1 indicates a perfect classifier and an AUC of 0.5 indicates a random classifier.

1.4 TCR-pMHC modelling

In this project we want to use molecular models of TCR-pMHC complexes for the prediction of their binding. As already mentioned, it is possible to make reasonably accurate models of TCRs because of their conserved framework and the canonical structure model. The tools for modelling of antibodies are especially well-developed because of the many applications of antibodies in pharmaceutical and other industries. One such tools is Rosetta Antibody, which identifies templates for the framework regions and each of the CDRs, assembles them and optimizes the side-chains [27].

One of the main challenges in modelling antibodies is the third hypervariable loop of the heavy chain (H3), which is the longest of the CDRs. Messih et al. developed a method that uses a Random Forest model to select the structural template for the H3 loop and were able to get a faster and more accurate model than Rosetta Antibody [28].

Methods for modelling TCRs are more scarce and the models that exist are less precise because there are fewer solved structures of TCRs. One tool for modelling of both BCRs and TCRs is LYRA (Lymphocyte receptor automated modelling), developed by Klausen et al. in 2015 [29]. LYRA is able to model TCRs with an average RMSD of 1.48Å.

Jensen et al. developed in 2019 a tool for TCR-pMHC modelling called TCRpMHCmodels which we will use in this project [30]. TCRpMHCmodels uses LYRA to model the TCR and MODELLER to model the pMHC and to assemble the two. It produces models with a median Cα RMSD of 2.31Å.

1.5 TCR-pMHC binding predictions

In this section, we will give a brief overview of the previous studies on TCR-pMHC binding predictions.

Tools for predicting the binding of a peptide to an MHC-I are already reasonably accurate. One tool for this prediction is NetMHCpan, first developed by Ilka Hoof et. al in 2009 [31, 32]. The method uses a neural network model and is able to make accurate affinity predictions.

The prediction of TCR-pMHC binding is a more complicated task and the tools that exist are still inadequate. One tool for TCR-pMHC predictions is NetTCR, a sequence-based model developed by Jurtz et al. in 2008 [33]. This tool predicts binding between TCRs and peptides presented by HLA-A*02:01 using a CNN. The model takes only the sequence of the peptide and the CDR3b as input. A submitted paper by Montemurro et al. further develops NetTCR and finds that including both CDR3a and CDR3b increases performance considerably [34].

Another study by Lanzarotti et al. from 2018 uses molecular modelling and force-field energy terms to predict the binding of a TCR and a pMHCII [35]. They model TCR-pMHCII complexes and determine immunogenicity by calculating interaction energies using FoldX and pyRosetta. They find that the force field energies are a good indicator of binding, but that a modified model is needed to get more accurate predictions.

In the Master's thesis by Olsen from 2018, a model is proposed which combines force field energy terms and deep learning [36]. The energy terms are derived from FoldX and used to train a convolutional neural network together with the template identities. The network consists of a separate CNN for each part of the complex (MHC, peptide, TCRα, TCRβ) which are concatenated and fed to an additional CNN (figure 8).

In the Master's thesis by Høie from 2019, a siamese-like convolutional neural network was built for TCR-pMHC predictions [37]. This network has a CNN for the pMHC and a CNN for the TCR, a random projection module for each and which is then combined using the hadamard product (figure 9). The study finds that training on energy terms outperforms training on the amino acid sequence alone and that training on both energy terms and sequences gives the best performance.

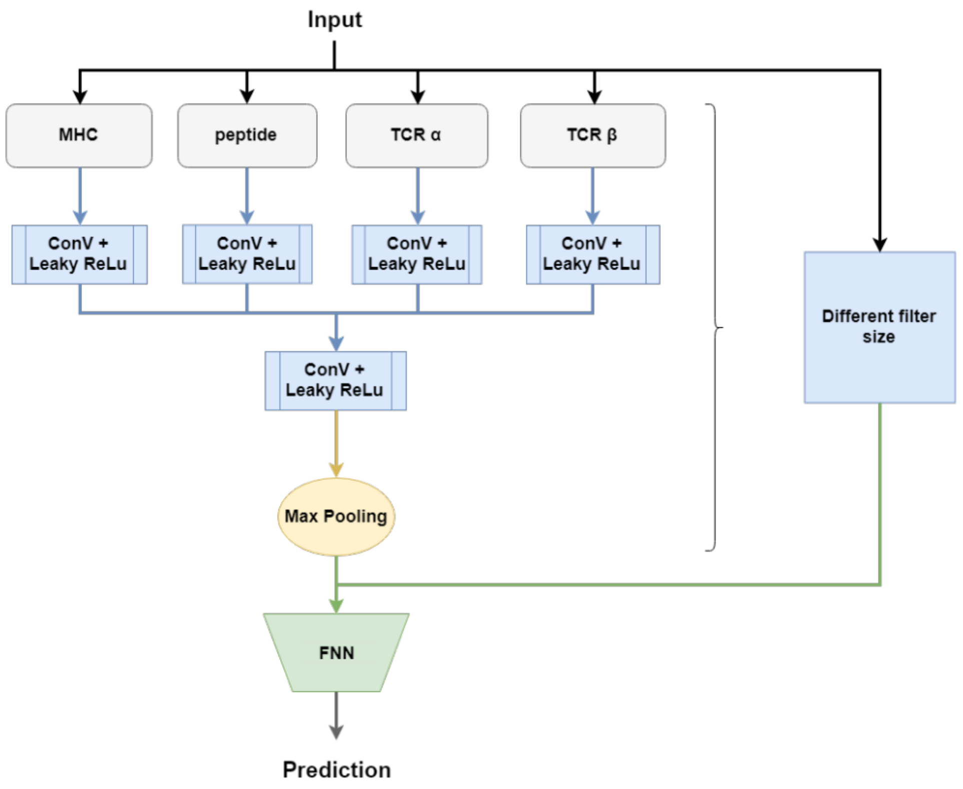


Figure 8: Architecture of the neural network used in the Master's thesis of Olsen to predict TCR-pMHC complexes from FoldX energy terms and template identity. The network consists of a separate CNN for each subunit and a combined CNN. Figure adapted from [36].

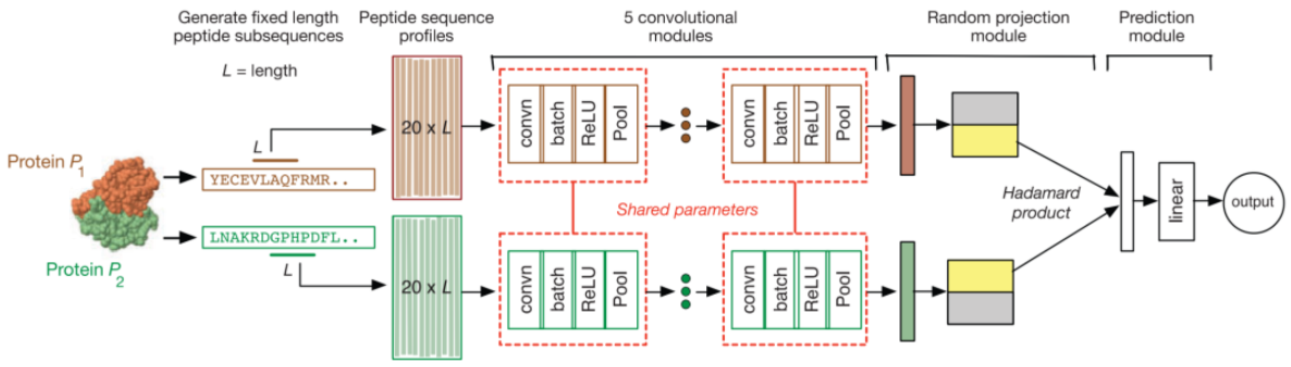


Figure 9: Siamese-like convolutional neural network architecture used in Høie [37]. Originally designed by [38].

1.6 Immunological data

The prediction tools for TCR-pMHC binding depend highly on the available training data. Unfortunately, the available data on TCR sequences and their pMHC targets is limited in amount and diversity.

There are two immunological databases with epitope information, which we will use as positive data in this project, the IEDB [39] and the VDJdb [40]. The IEDB (Immune Epitope Database) contains information on TCRs with known antigen specificities from published literature manually curated by national institutes of health. The IEDB only contains the CDR3 region of the TCRs and not the germlines. The VDJdb also contains TCR-pMHC pairs and includes both the CDR3 sequences and germlines.

For negative complexes, we will use the 10X Genomics dataset [41]. The 10x Genomics dataset is a dataset made from single cell immune profiling of four healthy humans.

1.7 Project scope

The aim of this project is to make a model that can predict the binding of a T-cell receptor to a peptide-MHC. Building upon the findings of the studies mentioned in section 1.5, we want to use both the amino acid sequence and force field energy terms as input to a state of the art deep learning model.

We will be using a benchmarking dataset from [34] with positives from the IEDB and VDJdb and negatives from the 10x Genomics dataset. We will model these complexes and relax and score them using the FoldX and Rosetta force fields. Jurtz et al. states that a good starting point for machine learning on biological sequence data is a seq-to-CNN-to-biLSTM-to-Attention-to-Dense-to-Output architecture [24]. We will use a simplified version of this architecture, namely a seq-to-CNN-to-biLSTM-to-Dense-to-Output.

The main differences between this project and the Master's theses by Høie and Olsen [37, 36] (as referred to in section 1.5) are that we use molecular models instead of only experimentally determined structures, which gives us a much bigger dataset. Furthermore, we include also Rosetta energy terms apart from FoldX energy terms, and we use both global and per-residue energy terms. Finally we use a state-of-the-art deep learning architecture including a biLSTM network.

Some of the questions that we seek to answer are the following:

- Is a combined CNN-biLSTM a good architecture for prediction of TCR-pMHC binding?
- Can force field energies improve the performance compared to training on the amino acid sequence only?
- Can Rosetta per-residue energy terms improve the performance?

- How does it affect the performance to exclude the MHC, peptide and TCR one by one from the training?

A flowchart of the project work is seen in figure 10.

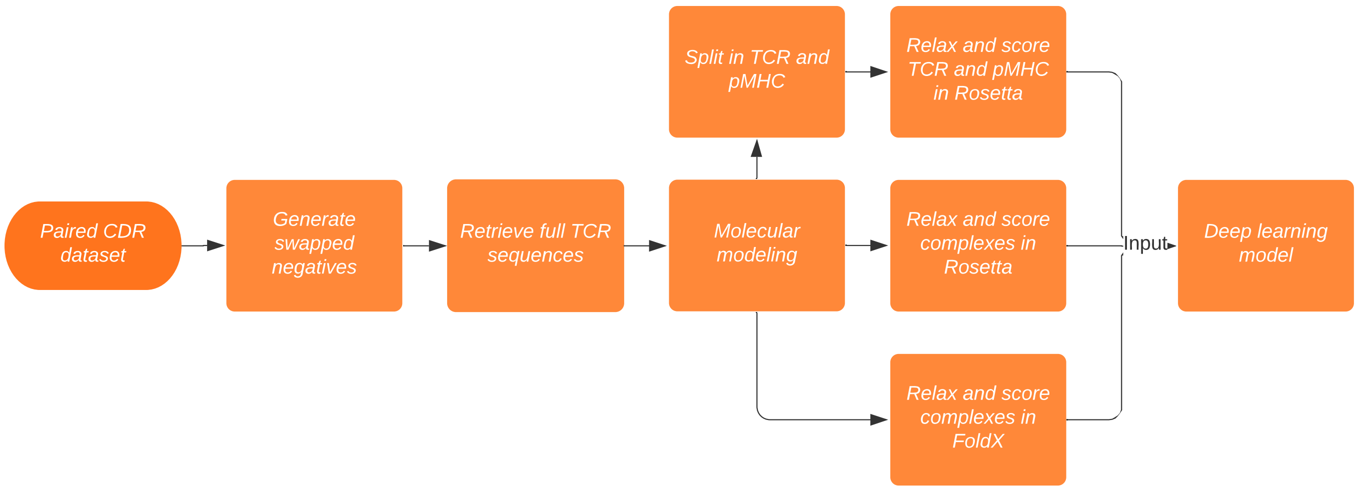

Paired CDR dataset
Generate swapped negatives
Retrieve full TCR sequences
Molecular modeling
Split in TCR and pMHC
Relax and score TCR and pMHC in Rosetta
Relax and score complexes in Rosetta
Relax and score complexes in FoldX
Input
Deep learning model

Figure 10: Flowchart of the project work.

2 Materials and Methods

2.1 Creation of benchmarking dataset

2.1.1 Paired CDR3 sequence dataset

The starting point of the benchmarking dataset created in this project was a dataset of paired CDR3α and β sequences from a submitted paper by Montemurro et al. [34]. This dataset was constructed in the following way.

The positive data was downloaded from IEDB on August 26th, 2020 and from VDJdb on August 5th 2020. The download was restricted to TCRs that bind to HLA-A*02:01-specific peptides of length 9, where both CDR3α and CDR3β are available and of length 8 to 18. The negative data was derived from the 10X Genomics dataset [41] using the same restrictions as for the positives and restricting to 0 UMI counts.

Redundancy reduction was performed using the Hobohm 1 algorithm for the positive and negative datasets separately. The positives and negatives were reduced to data points containing their shared set of peptides. Hereafter, the positives were partitioned into 5 sets using single-linkage clustering. As similiarity measure for the clustering, the average of CDR3α and CDR3β Levenshtein similarity was used with a threshold of 95%.

Negatives were added to each partition by adding 5 negative CDR3 pairs for each peptide in the partition. Only TCRs with a similarity lower than 95% to all TCRs in the other partitions were added. Throughout this project the original five partitions are used.

2.1.2 Generation of swapped negatives

In order not to have TCRs that were only present as positives, additional ”swapped” negatives were generated. One swapped negative was added for each positive, by matching the TCR with a random peptide from the same partition.

2.1.3 Retrieval of full TCR sequences

In the original dataset, only the CDR3 region of the two TCR chains and the peptide sequence were provided. Since the whole TCR sequence was needed for the modeling, these were generated in the following way.

The V and J genes were already available for the negatives in the dataset, but not for the positives. To find the gene information for the positives, these were mapped on to the VDJdb. Not all entries could be found, since some of the ones from IEDB were not present in VDJdb, and some of the entries were missing the gene information in VDJdb.

Based on the V and J gene names, the V and J sequences of each TCR chain were retrieved. The V sequence, CDR3 and J sequences were then combined and aligned to the pre-compiled HMM from Lyra [29]. In cases where the J sequence did not have an F or W followed by a G, the entry was discarded.

For the MHC, the HLA-A*02:01:01 sequence was retrieved from the IPD-IMGT/HLA database [42].

2.1.4 Molecular modelling

For each entry a fasta file was created with the TCRα, TCRβ, peptide and HLA*02:01 sequences. TCRpMHCmodels was run for each fasta file.

2.1.5 FoldX energy calculations

Each model pdb file was relaxed in FoldX5.0 using the `RepairPDB` command with the flags `--ionStrength=0.05 --pH=7 --water=CRYSTAL --vdwDesign=2 --out-pdb=1 --pdbHydrogens=false`. Hereafter, the energy terms were calculated using the `AnalyseComplex` command on each repaired pdb-file.

For each complex, the six interaction energy terms were retrieved from the output file `Interaction_X_Repair_AC.fxout`.

2.1.6 Rosetta energy calculations

The models were relaxed in Rosetta Energy Function 2015 using the `relax.default.linuxgccrelease` command. We experimented with the options for relaxation, for example by varying `-fa_max_dis`, by including `-relax:cartesian` and `ex1`, `ex2` and `-flip_HNQ`. We ended up using the default options.

Global energy terms were calculated for the relaxed models using the `score_jd2.linuxgccrelease` command. The per-residue energy terms were calculated for each complex using the `per_residue_energies.linuxgccrelease` command.

The model PDB was then split into TCR and pMHC. These were then each relaxed and scored in the same way as described above.

2.1.7 Final dataset

The benchmarking dataset was assembled as a 2D array for each complex, having residue position as the first dimension, and feature as the second dimension. The TCRs were padded by adding rows of zeroes so all of them had the same length. The features included the one-hot-encoding of the amino acid, the Rosetta per-residue energies for complex and separated, the Rosetta global energies for complex and separated, and finally the FoldX interaction energy terms.

2.2 Neural network

A new architecture was coded in Python3.7 with Pytorch. The architecture consists of two CNNs and a BiLSTM (figure 11).

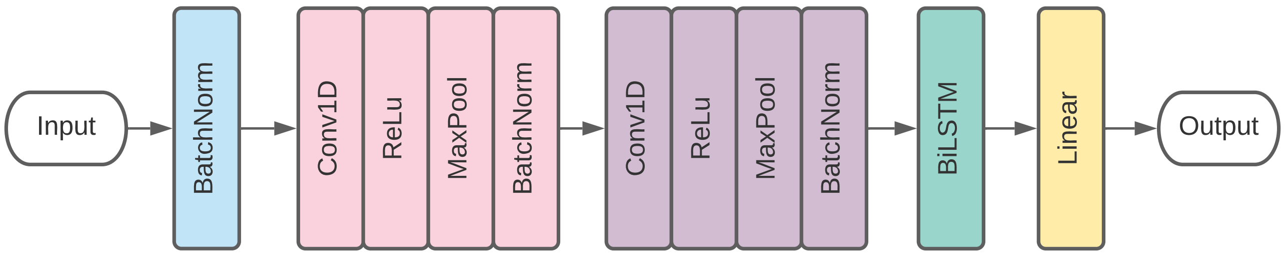


Figure 11: Architecture of the neural network. The input data has 54 features. Both convolutional layers have 100 output channels, a kernel size of 3, and a stride of 2. The MaxPool layers have a kernel size of 2. The biLSTM has 3 layers and a hidden size of 16. The linear layer has 52 nodes. Dropouts are included after each MaxPool, inside the biLSTM and afer the biLSTM with a probability of 10%. The Adam optimizer is used with L2 regularization.

The models were trained using nested cross-validation for maximum 100 epochs using early stopping after 5 epochs without a decrease in loss. The Adam optimizer was used to update the weights with a learning rate of 0.001 and a weight decay of 0.0005. BCEWithLogitsLoss was used as loss function and a batch size of 128 was used.

Different experiments were performed where, for example, parts of the sequence were removed from the input data or only parts of the features were used.

Per-residue performances were determined by training and testing the model in the usual way and hereafter retrieving the predictions for each peptide and calculating the performance.

Unless otherwise stated, the swapped negatives were removed from the test set so that the performance was evaluated only on the positives and the 10X negatives.

A leave-one-out experiment was performed where each peptide was removed from the training set one-by-one. The model was then trained on the remaining entries, and performance was evaluated for the peptide that was removed from the training.

3 Results

3.1 Characterization of benchmarking dataset

We have created a benchmarking dataset of sequences and force field energy terms for binding and non-binding TCR-pMHC complexes for the HLA-A*02-01 allele. The binding complexes (positives) are from the VDJdb and IEDB. For each positive there are five non-binding complexes (negatives) from the 10x Genomics dataset. Additional swapped negatives were constructed for each positive by matching the TCR with a different peptide in order not to have any TCR that was only present in the positives. We decided to have this 1:6 positive to negative ratio, because it reflects the actual scenario, where a TCR only binds to very few pMHC complexes. Adding more negatives might be too big of a class imbalance, so that the model would easily collapse into predicting everything as negatives.

It should be noted that the modelling and energy calculations failed for some entries, so not all the entries from the original sequence dataset are present in the final dataset. The final dataset consists of 9991 entries, out of which 1726 are positives and 8265 are negatives.

There are 18 different peptide antigens in the dataset, all of which are 9-mers. There is a large bias in the frequencies of the peptides (figure 12). By far the most frequent entry is the peptide "GILGFVFTL" from Influenza virus, which constitutes 60.3 % of the entries. The two other frequent peptides are "GLCTLVAML" and "NLVPMVATV" from Herpes virus, which constitute 16.1% and 11.8% respectively.

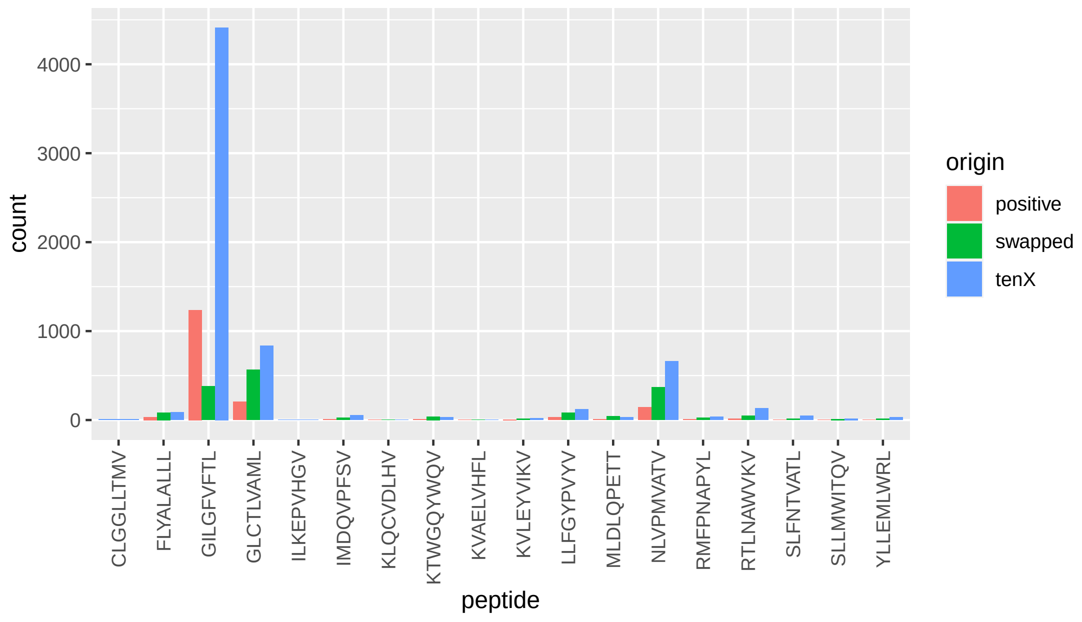
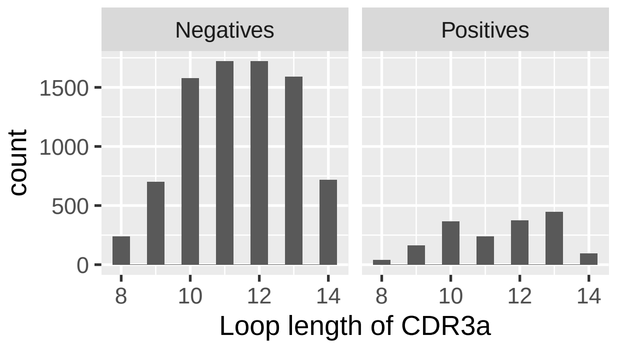
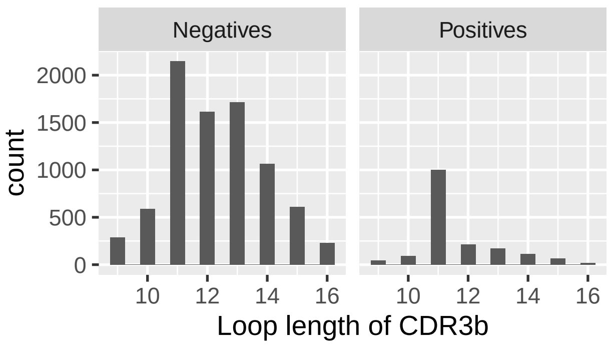


Figure 12: Counts of peptides in the dataset. The majority of the peptides are "GILGFVFTL" from Influenza virus. The other two frequent peptides are "GLCTLVAML" and "NLVPMVATV" from Herpes virus.

There are 6500 and 6735 unique CDR3α and CDR3β sequences in the negatives and 1106 and 1017 in the positives. The length of the CDR3α loop ranges from 8 to 14, while the length of the CDR3β loop ranges from 9 to 16 (figure 13).



(a)



(b)

Figure 13: Distribution of CDR3 loop length in the dataset. a) Length of CDR3α. Loop length ranges from 8 to 14. b) Length of CDR3β. The loop lengths are slightly longer than for CDR3α, ranging from 8 to 16.

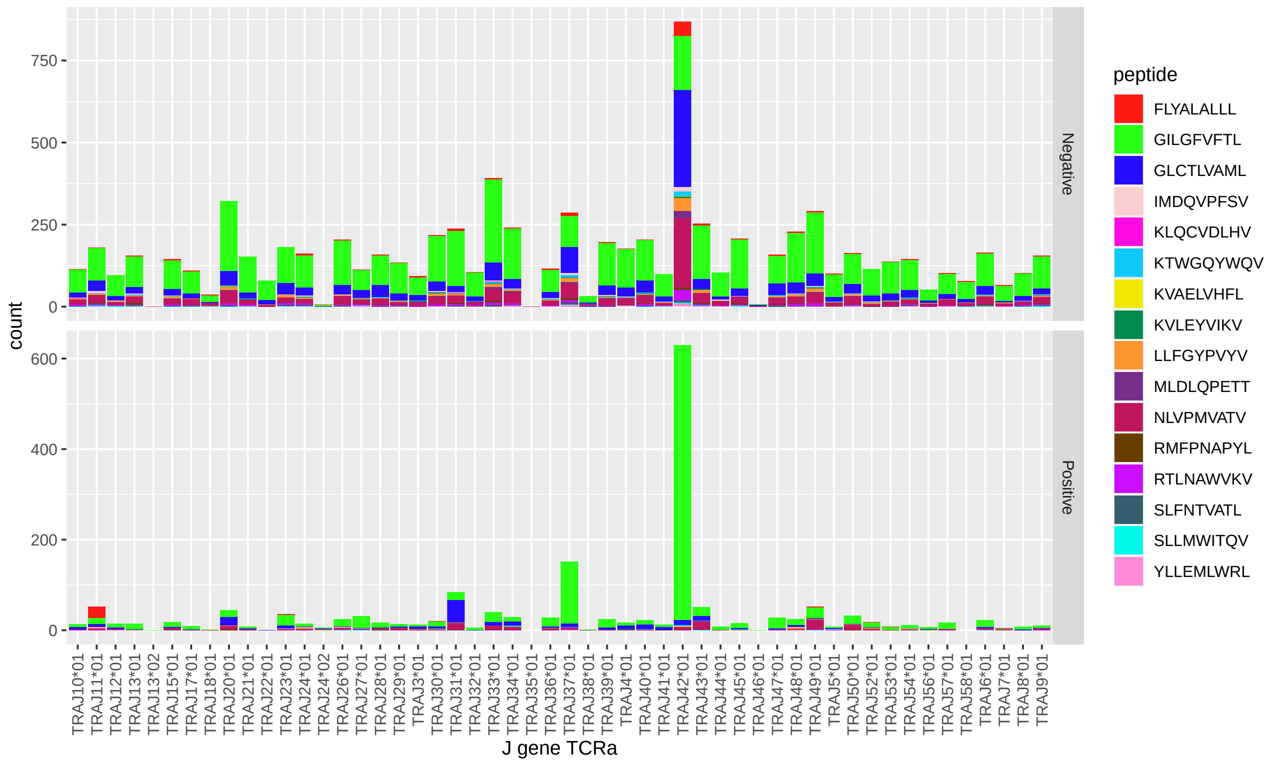
There is a considerable bias in the germlines in the dataset, especially for ”GILGFVFTL” and ”FLYALALLL” (figure 14 and 15). For the TCRα J gene, most of the positive ”GILGFVFTL” bind to TCRs with germline TRAJ42*01, whereas the negatives are distributed much more equally among all the germlines (figure 14a). For ”FLYALALLL”, almost all of the positives are TRAJ11*01 and almost all of the negatives are TRAJ42*01. Likewise, for the TCRα V gene, there is a strong bias for both ”GILGFVFTL”, ”FLYALALL”, and ”GILCTVAML” (figure 14b).

The TCRβ also has a strong bias in the germlines for ”GILGFVFTL”, ”FLYALALLL”, and ”GLCTLVAML”, especially in the V gene (figure 15). Also, some of the less frequent peptides have a bias here, e.g. ”LLFGYPVYV” and ”KTWGQYWQV”. This bias in germlines poses a risk of the deep learning model to overfit, because it may learn the germline bias by heart instead of learning from physical properties in the sequence.

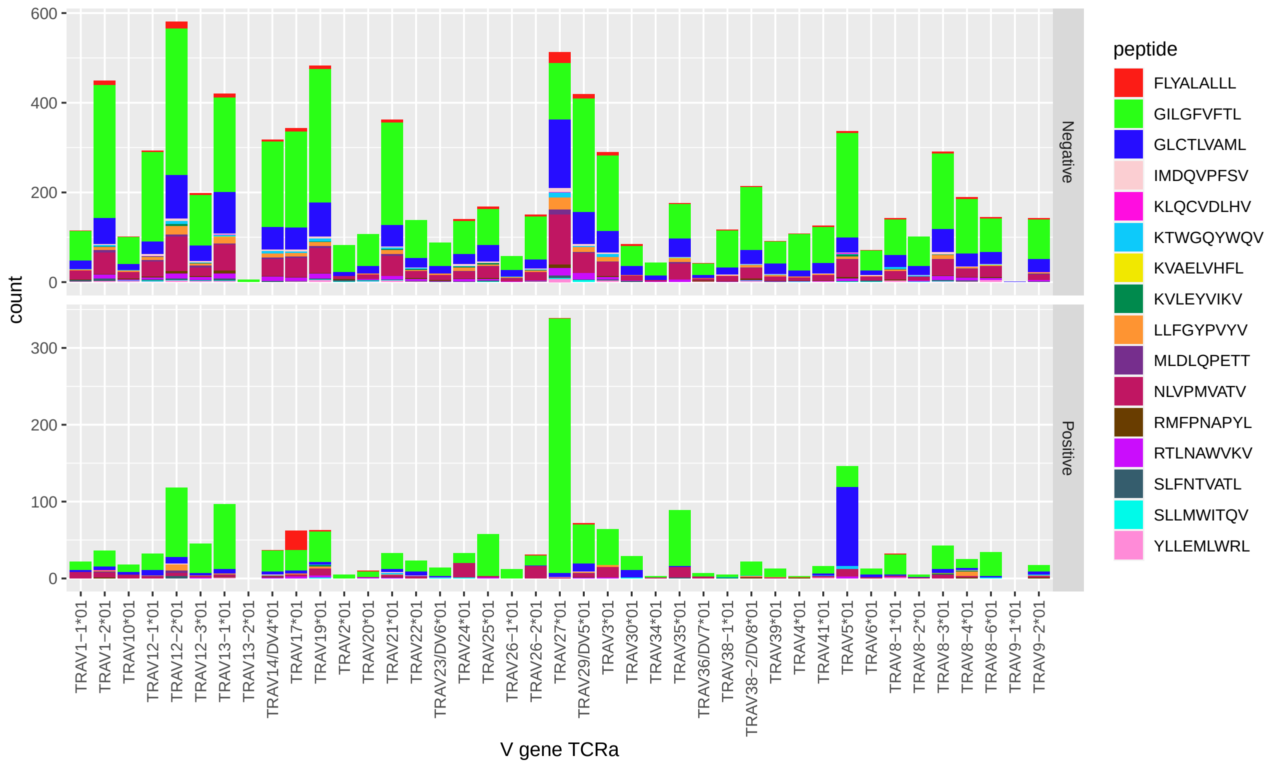
The dataset is divided in 5 partitions based on the CDR3 sequence similarity with a threshold of 95%. All peptides are present in all partitions with similar frequencies (Appendix A). Likewise, the germlines are present with similar frequencies in all partitions (Appendix B). This can also lead to overfitting for the model.

3.2 Molecular modeling

The complexes were modeled using the in-house pipeline TCRpMHCmodels. The modeling succeeded for 82.9% of the positive complexes and 97.5% of the negative complexes. In the cases where it failed, the reason was that it was not able to find a template for the CDR3 loop for either the alpha or beta chain.

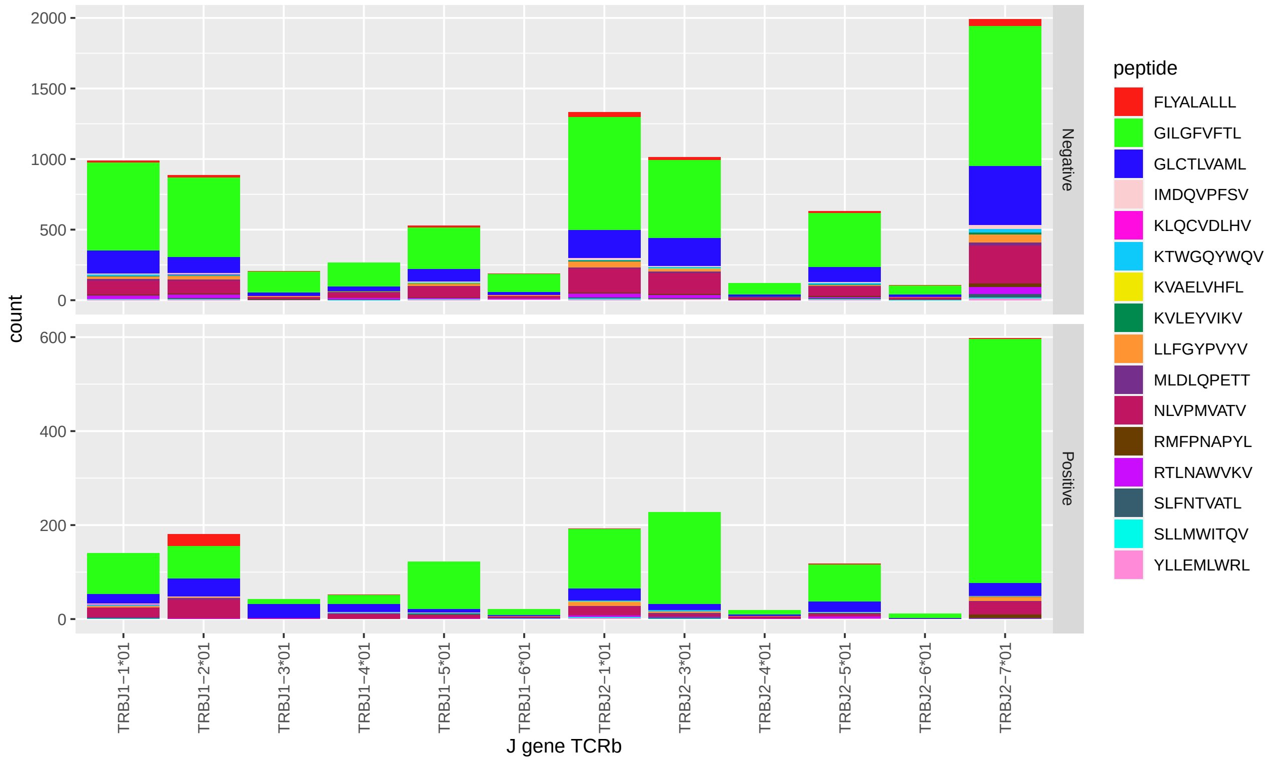


(a)

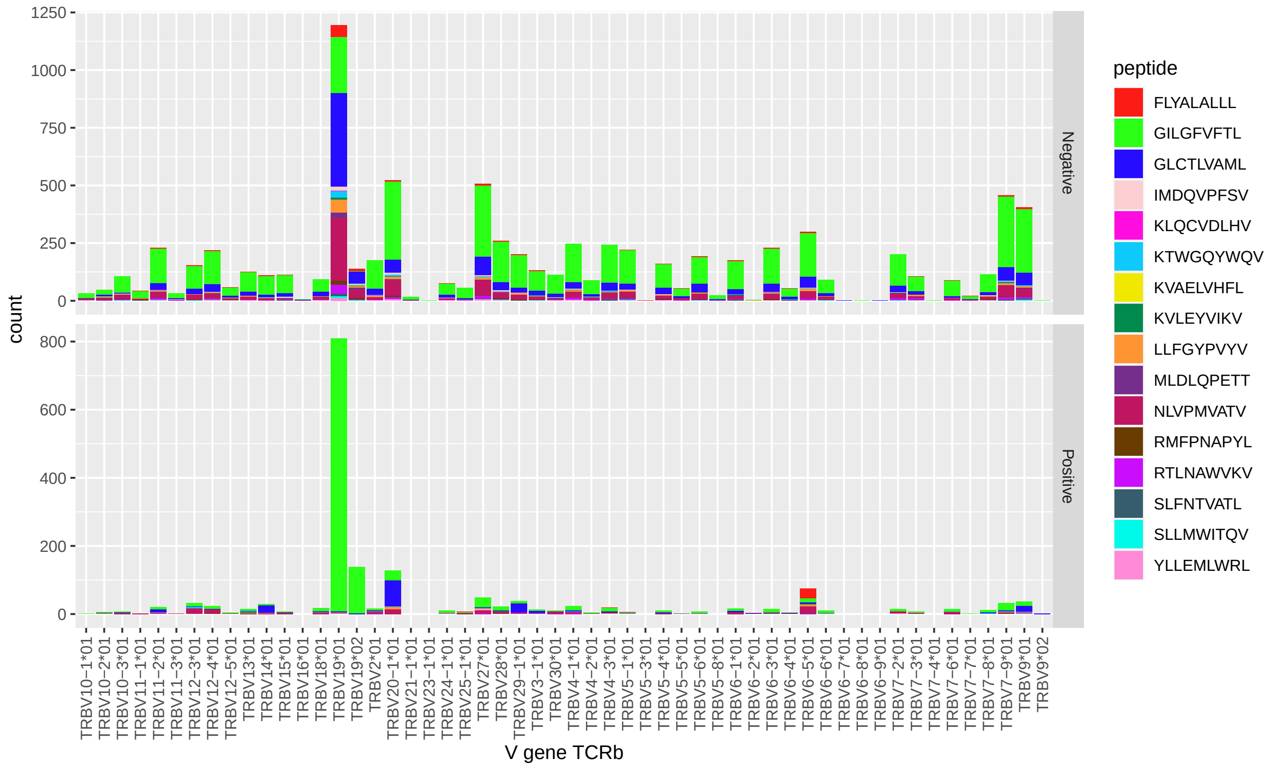


(b)

Figure 14: TCRα germline distribution. a) J gene TCRα. There is a strong bias for ”GILGFVFTL”, which is mostly in complex with TRAJ42*01 germlines for the positives. ”FLYALALLL” is only in positive complex with TRAJ11*01 and almost exclusively in negative complex with TRAJ42*01. b) V gene alpha. ”GILGFVFTL” in complex with TRAV27*01 germline is much more frequent than other germ lines. ”FLYALALLL” is almost exclusively in complex with TRAV17*01 germline in the positives. ”GLCTLVAML” is almost exclusively in complex with TRAV5*01 in the positives.



(a)



(b)

Figure 15: TCRβ germline distribution. a) J gene TCRβ. There is a strong bias for TRBJ1-2*01 for ”FLYALALLL” in the positives. b) V gene TCRβ. There is a strong bias for TRBV19*01 for ”GILGFVFTL” in the positives. There is also a strong bias for TRBV6-5*01 for ”FLYALALLL” in the positives.

3.3 Energy terms

The positive and negative complexes were relaxed and scored using the FoldX and Rosetta force fields. In Rosetta, we experimented with using different options for the relaxation, for example relaxing it in cartesian space. When studying the output PDBs, we found that the different relaxation procedures all improved the model and that their output was very similar. We therefore decided to relax them with the default procedure.

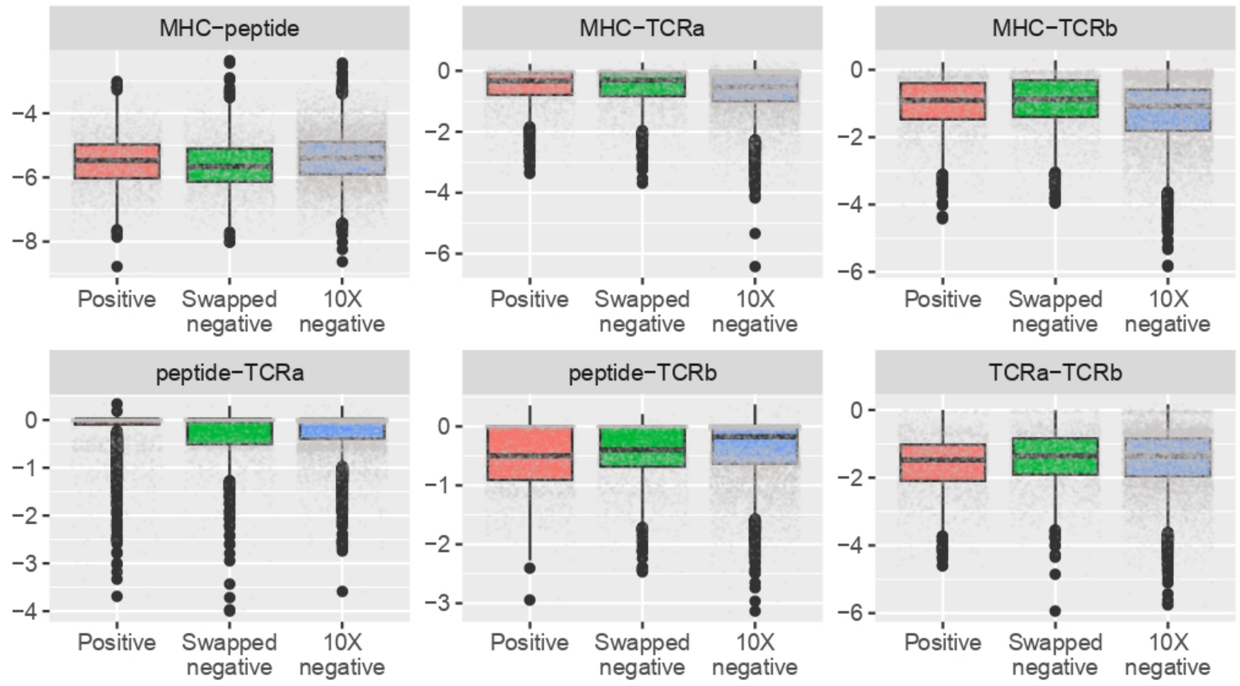
From FoldX, the interaction energies were used, i.e. the binding energies between each protein in the complex. Boxplots of the FoldX interaction energy terms are seen in figure 16a. The lower the interaction term, the stronger is the binding. The biggest difference between positives and negatives is seen for the peptide-TCRβ interaction energy. The median is notably lower for the positives than for the negatives, which means that the peptide-TCRβ binding is stronger for the positives. This is expected, since the TCRβ is the most responsible for binding to the peptide. The swapped negatives are in between the positives and the 10x negatives. The other interaction energies are all very similarly distributed for positives and negatives. The fact that MHC-peptide interaction term is relatively low for all the complexes shows that there is a strong MHC-peptide binding for both the positives and the negatives.

From Rosetta, the total score was used, as well as six individual energy terms, fa_atr, fa_rep, fa_sol, fa_elec, fa_dun, and p_aa_pp (table 2). Boxplots of the Rosetta energy terms are seen in figure 16b. The distributions show that there are only subtle differences in the distributions between the positives and the negatives. The median ΔE_{total} is slightly lower for the 10x negatives than for the positive. This is expected, as a lower ΔE_{total} means that the complex is less stable than the separated complex. For the individual energy terms, there are also small differences in the distributions between positives and negatives.

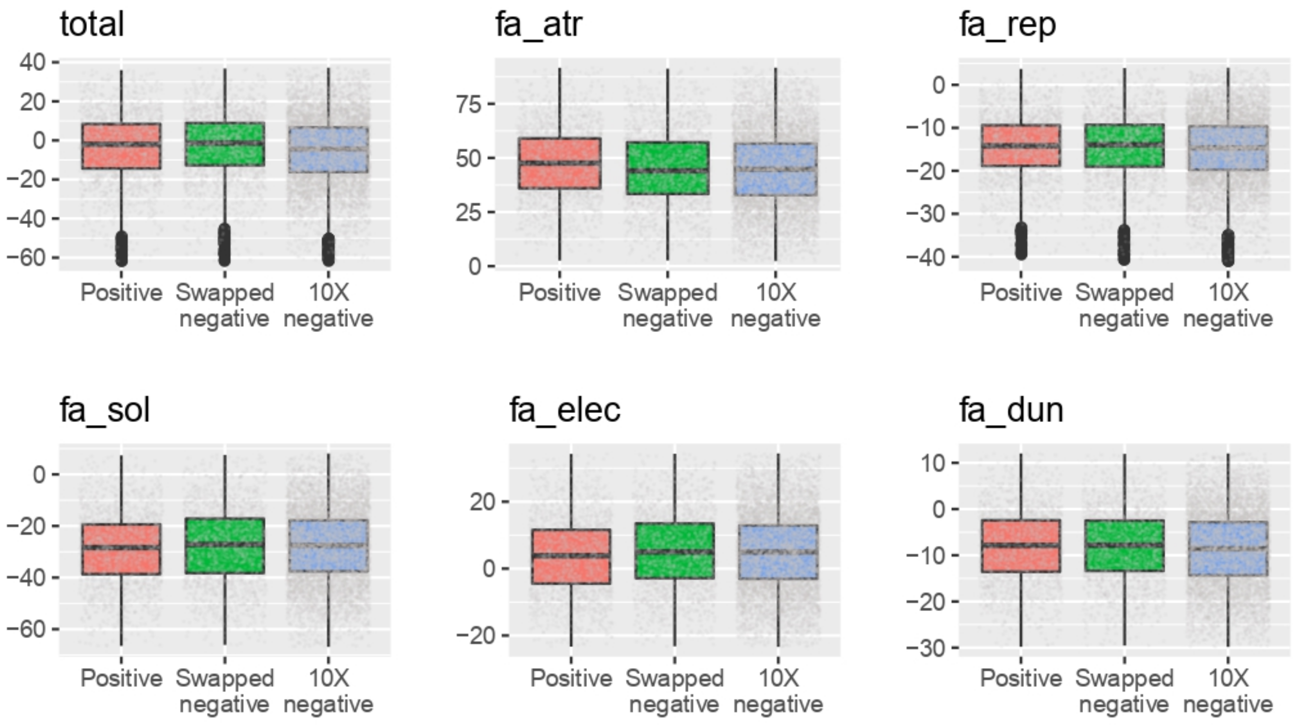
To sum up, the distributions of the energy terms indicate that there are differences between the positives and negatives, although the differences are subtle. The FoldX peptide-TCRβ interaction energy and the Rosetta total energy have the biggest difference between positives and negatives.

fa_atr	Lennard-Jones attractive between atoms in different residues
fa_rep	Lennard-Jones repulsive between atoms in different residues
fa_sol	Lazaridis-Karplus solvation energy
fa_elec	Coulombic electrostatic potential with a distance-dependent dielectric
fa_dun	Internal energy of sidechain rotamers as derived from Dunbrack's statistics
p_aa_pp	Probability of amino acid, given torsion values for phi and psi

Table 2: Definitions of the selected Rosetta individual energy terms from the beta_nov15 score function [20].



(a)



(b)

Figure 16: Boxplots of force field energy terms. a) FoldX interaction energy terms. The peptide-TCRβ interaction energies are slightly higher for the 10x negatives and swapped negatives than for the positives. b) Δ Rosetta energy terms. The energy term for the separated complex minus the energy term for the complex. The ΔE_{total} is slightly lower for the 10x negatives than for the positives. There are only small differences in the other Rosetta terms.

3.4 Prediction model

A deep learning model was constructed for prediction of TCR-pMHC binding. The input to the network is a 2D array for each entry with the residue position as the first dimension and features as the second dimension (figure 17). The features consist of the one-hot encoding of the amino acid, the Rosetta per-residue energy terms, the FoldX interaction energy terms and the Rosetta global energy terms (table 3). The FoldX terms and Rosetta global terms are added as constant values for all positions. The output of the model is a binary output indicating binding/non-binding.

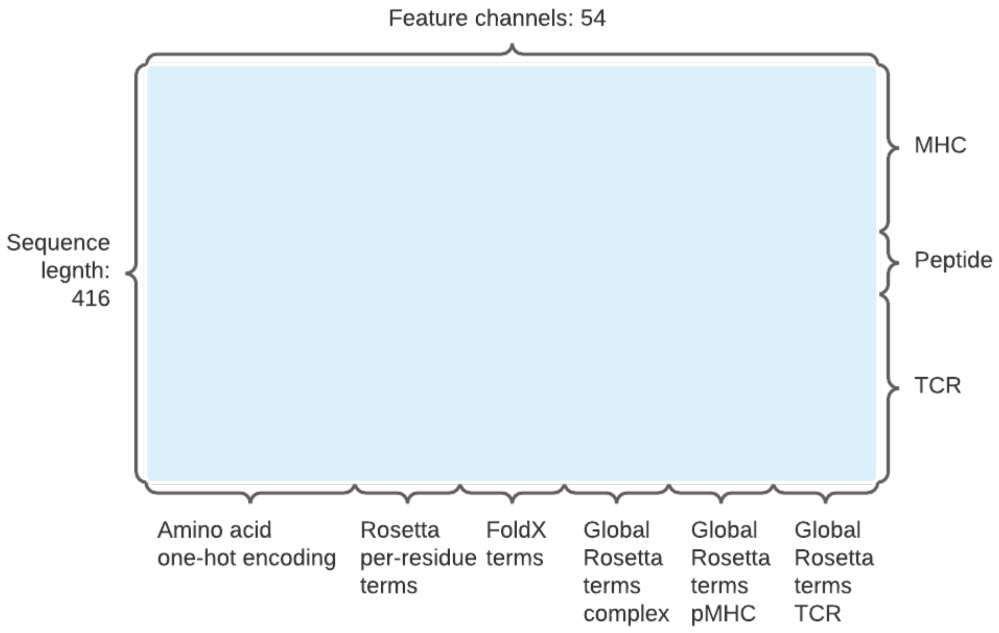

Feature channels: 54
Sequence legnth: 416
MHC
Peptide
TCR
Amino acid one-hot encoding
Rosetta per-residue terms
FoldX terms
Global Rosetta terms complex
Global Rosetta terms pMHC
Global Rosetta terms TCR

Figure 17: Dimensions of the input data. Each entry is represented by a 2D array with positions × features. The TCR sequences are padded so that they all have the same length. FoldX and global Rosetta terms are constant for all positions.

Feature	Number	Encoding
Amino acid sequence	20	One-hot encoded
Rosetta per-residue energy terms	6	Per-residue
FoldX interaction energy terms	6	Constant
Rosetta global energy terms complex	6	Constant
Rosetta global energy terms TCR	6	Constant
Rosetta global energy terms pMHC	6	Constant

Table 3: The features used in the neural network.

The architecture of the neural network is a seq-to-CNN-to-biLSTM-to-Dense-to-Output model. We found that adding dropout layers after each MaxPool, inside the biLSTM and after the biLSTM helped a lot to stabilize the training and to prevent overfitting (data not shown). Adding L2 regularization also helped to prevent the model from overfitting.

3.4.1 Overall performance

The model achieves a very good overall performance with an AUC of 0.84 and an MCC of 0.63 when training on all features and excluding the MHC (table 4). The model has a very good precision of 0.79 and a decent recall of 0.61.

AUC	MCC	Accuracy	Precision	Recall	F1	Confusion matrix	
0.84	0.63	0.88	0.79	0.61	0.69	25039	1105
						2717	4187

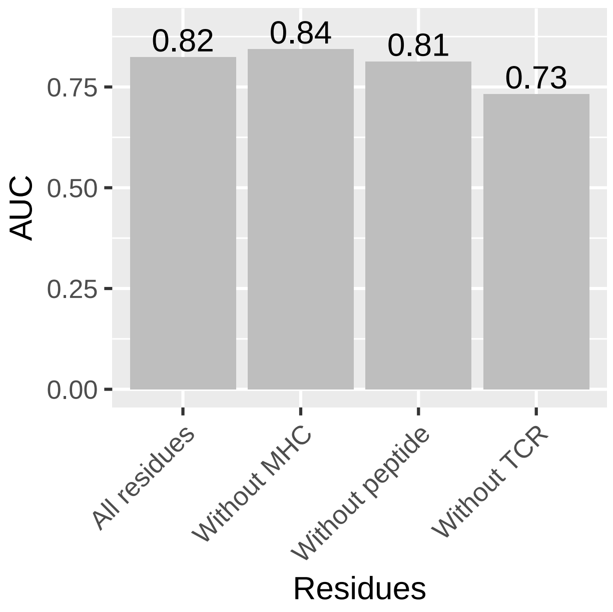
Table 4: Performance when using all features and excluding the MHC from the training. Performance is evaluated using nested cross-validation. The swapped negatives were excluded from the test set. The confusion matrix is defined in table 1.

In order to investigate whether it was necessary to include the whole complex in the input, we trained the model excluding MHC, peptide and TCR one-by-one. We see that the best performance, with an AUC of 0.84, is achieved when excluding MHC from the training (figure 18a). This is expected since the MHC is HLA-A*02-01 for all the entries, so the only difference is the Rosetta per-residue energies. Excluding the peptide from the training only resulted in a slightly worse performance with an AUC of 0.81. This is surprising, and indicates that the model is either learning from the global energy terms, or that it is overfitting on the bias in germlines. On the other hand, removing the TCR from the training gives a significantly lower performances with an AUC of 0.73. The fact that the model performs much better without the peptide than without the TCR further indicates that the bias in the germ lines is driving the predictions.

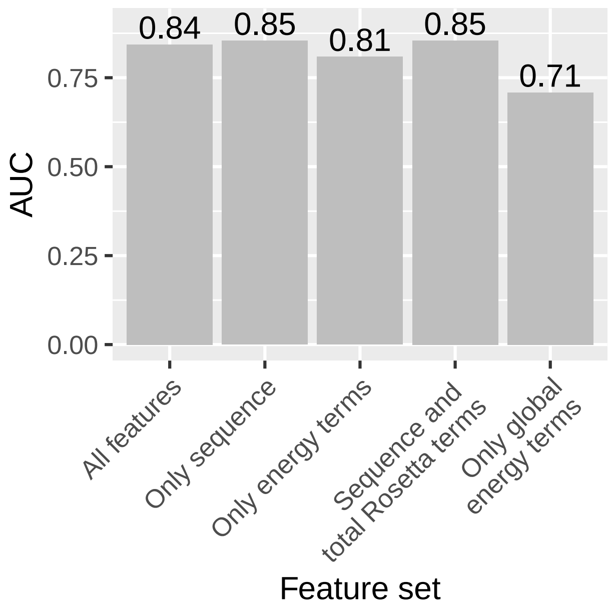
In order to study how the sequence and energy terms each contribute to the model, we trained the model on different subsets of features. Surprisingly, including the sequence alone showed a slightly better performance (AUC of 0.85) than including all features (figure 18b). Again, this is most likely because the model is learning from the bias in the germlines.

When training on the energy terms alone, the model also shows a decent performance of 0.81. This means that the energy terms do have a strong predictive power. Comparing all energy terms with only global energy terms, we see a much better performance for all energy terms (0.81 compared to 0.71). This means that the per-residue energies significantly improve the performance.

Sequence and total Rosetta energy terms (i.e. without the Rosetta individual energy terms) gave a better performance than with all features. This indicates that the individual energy terms do not



(a)



(b)

Figure 18: Overall performance of the deep learning model. a) Performance when excluding MHC, peptide and TCR one-by-one. The best AUC is achieved when MHC is excluded from the training. All models were trained on all features. b) Performance on different subsets of features. The best AUC is achieved when training on sequence only. The MHC was excluded from the training. The swapped negatives were excluded from the test set in both experiments.

improve the performance, but rather lead to overparameterization. The overall MCC for all the models are seen in Appendix C.

3.4.2 Performance on swapped and 10x negatives

Next, we set out to investigate how the model performs on the swapped negatives compared to the 10x negatives. In this setup, the model was trained on all the complexes, but evaluated on positives and swapped negatives and then on positives and 10x negatives. We expected the model to perform worse on the swapped negatives, because these TCRs are also present in the positives, but surprisingly, the model turned out to perform more or less equally when including all features and slightly better on the swapped negatives when training on sequence only (figure 19).

3.4.3 Per-peptide performance

In order to investigate how the model performed on each peptide, we made a setup to test the performance on each peptide separately. These results show that the model performs best on ”GILGFVFTL” which by far is the most frequent peptide in the dataset (figure 20). This is expected as the model is trained extensively on this peptide. In general there is a trend that the more frequent the peptide is, the better the model is at predicting it.

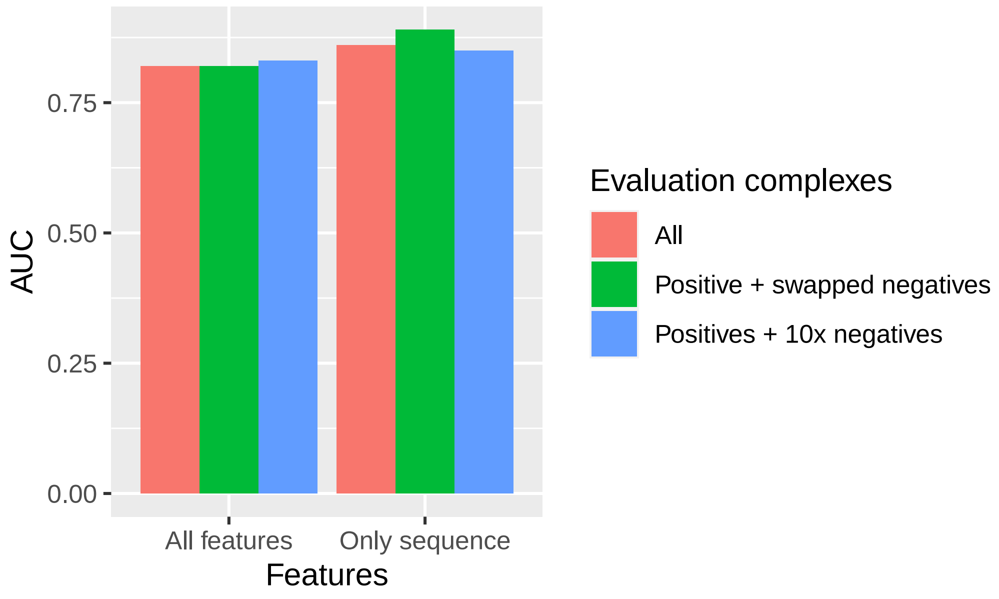


Figure 19: Performance on swapped and 10x negatives. The model performs equally well on the swapped and 10x negatives when training on all features. It performs slightly better on the swapped negatives when trained on sequence only. The MHC was excluded from the training.

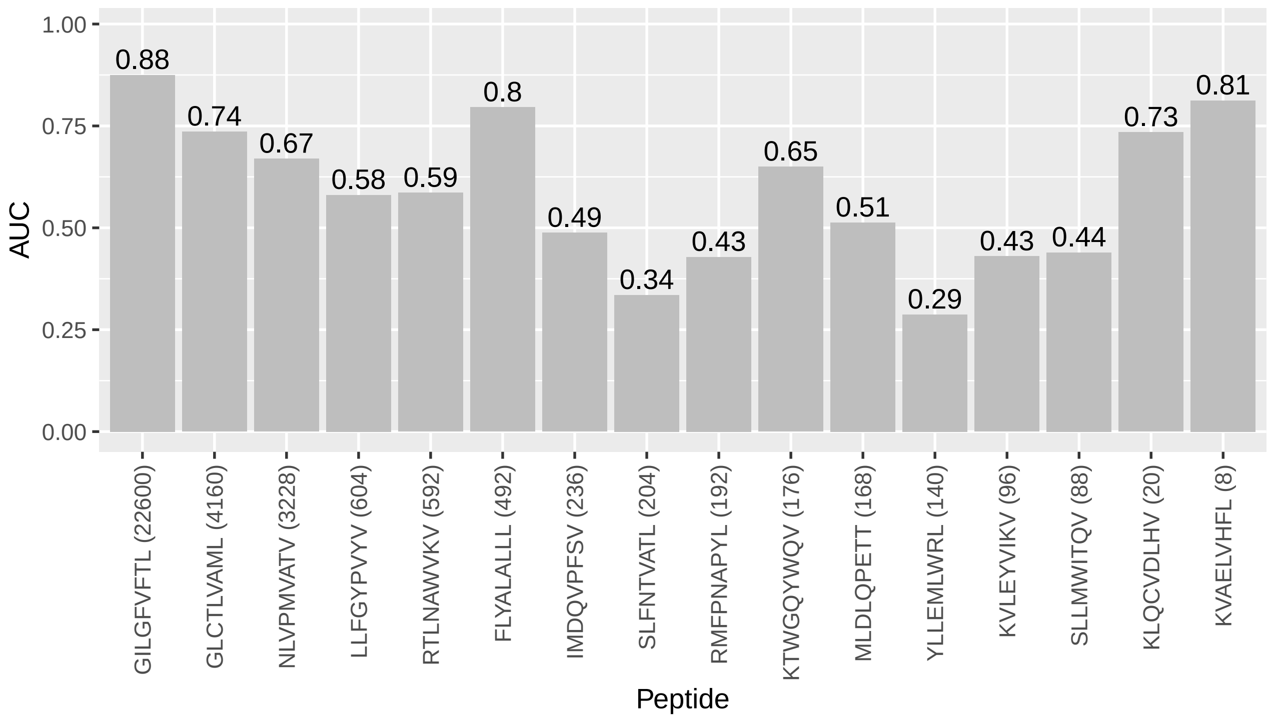


Figure 20: Per-peptide performance when training on all features and excluding MHC. The peptides are ordered according to their frequency in the dataset, and in parenthesis is the number of each peptide in the dataset. The model performs better on the more frequent peptides. The model also performs very well on ”FLYALALLL”, which had a large bias in germlines. The swapped negatives were excluded from the test set.

The model also performs very well on ”FLYALALLL”, even though there are only 492 of them in the dataset. However, ”FLYALALLL” had a very strong bias in germlines and probably this is the reason for the good performance on it. Thus it seems that the model overfits on the peptides that are very frequent in the dataset and on the germlines.

Next, we turned to investigate how training on different subsets of the features would affect the performance for each peptide. We hypothesized that while the energy terms did not improve the total performance, it might improve the performance on the less frequent peptides, because the model is not able to overfit on their sequences.

The result show that for many of the less frequent peptides and peptides that have less bias in germlines, including the energy gives a better performance (figure 21). For some peptides using the energy terms alone even performs best. However, the increase in performance when including energy terms is small in most of the cases.

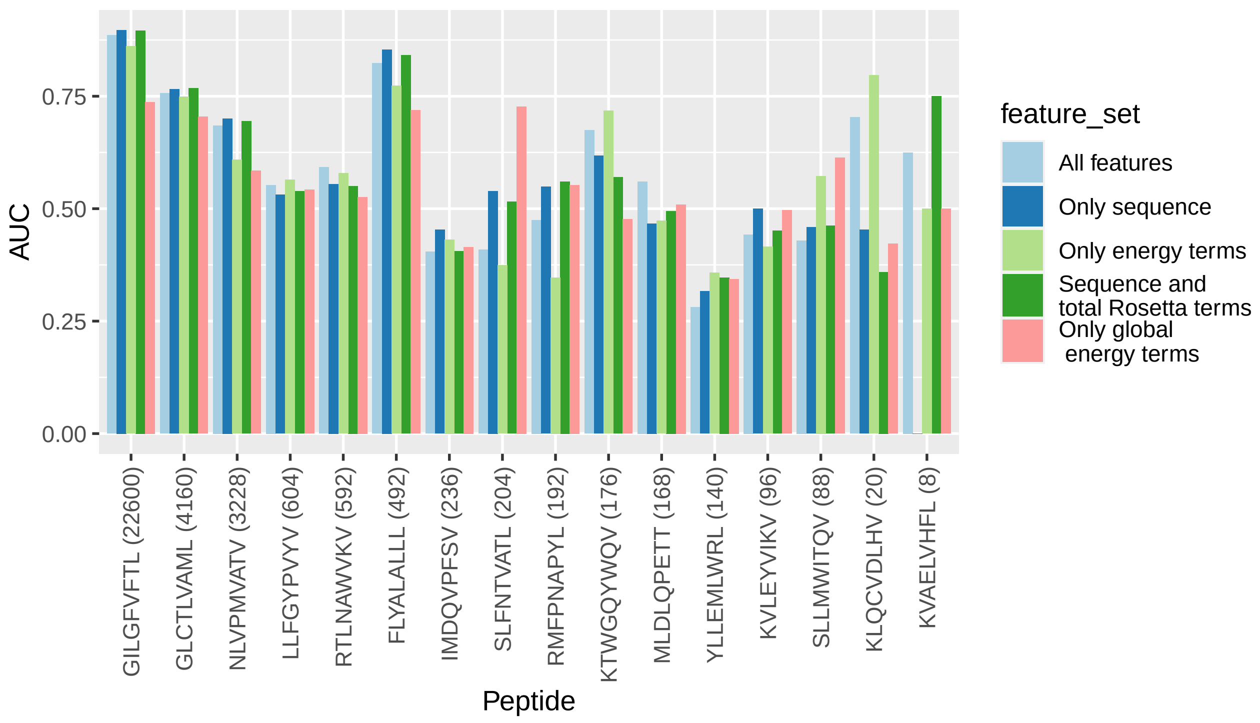


Figure 21: Per-peptide performance with subsets of features. For many of the less frequent peptides including the energy terms improves the performance. The MHC was excluded from the training, and the swapped negatives were excluded from the test set.

We also ran the experiment of leaving out parts of the sequence as above, but this time evaluating the performance per-peptide. These results show that for most of the peptides, the model performs best when excluding the MHC from the training (figure 22). For some of the less frequent peptides, this is not the case, but it might be due to the high uncertainties when there are very few samples.

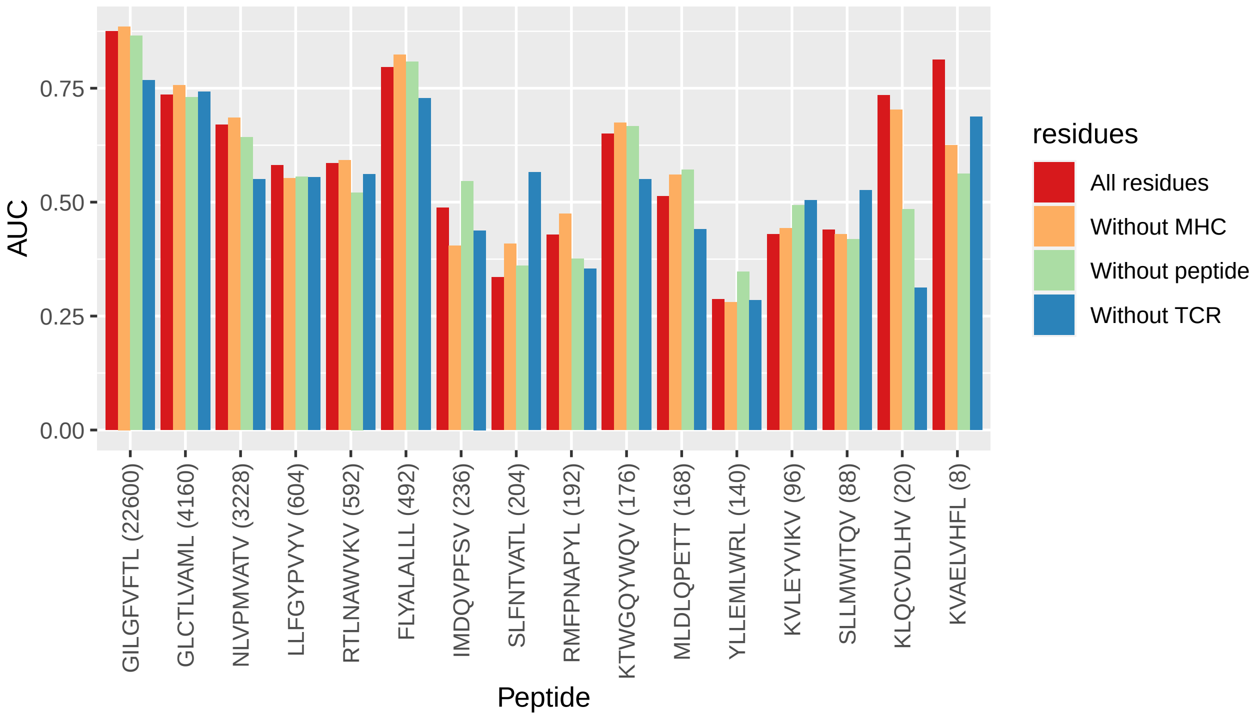


Figure 22: Per-peptide performance when excluding MHC, peptide and TCR one-by-one. Excluding MHC gives the best performance for most of the peptides. The swapped negatives were excluded from the test set.

3.4.4 Leave-one-out performance

There are several things that indicate that the model is overfitting. Since we do not have an external dataset to assess overfitting, we ran a leave-one-out experiment. In this setup, the model was trained on all peptides except one, and then the performance was evaluated on that peptide.

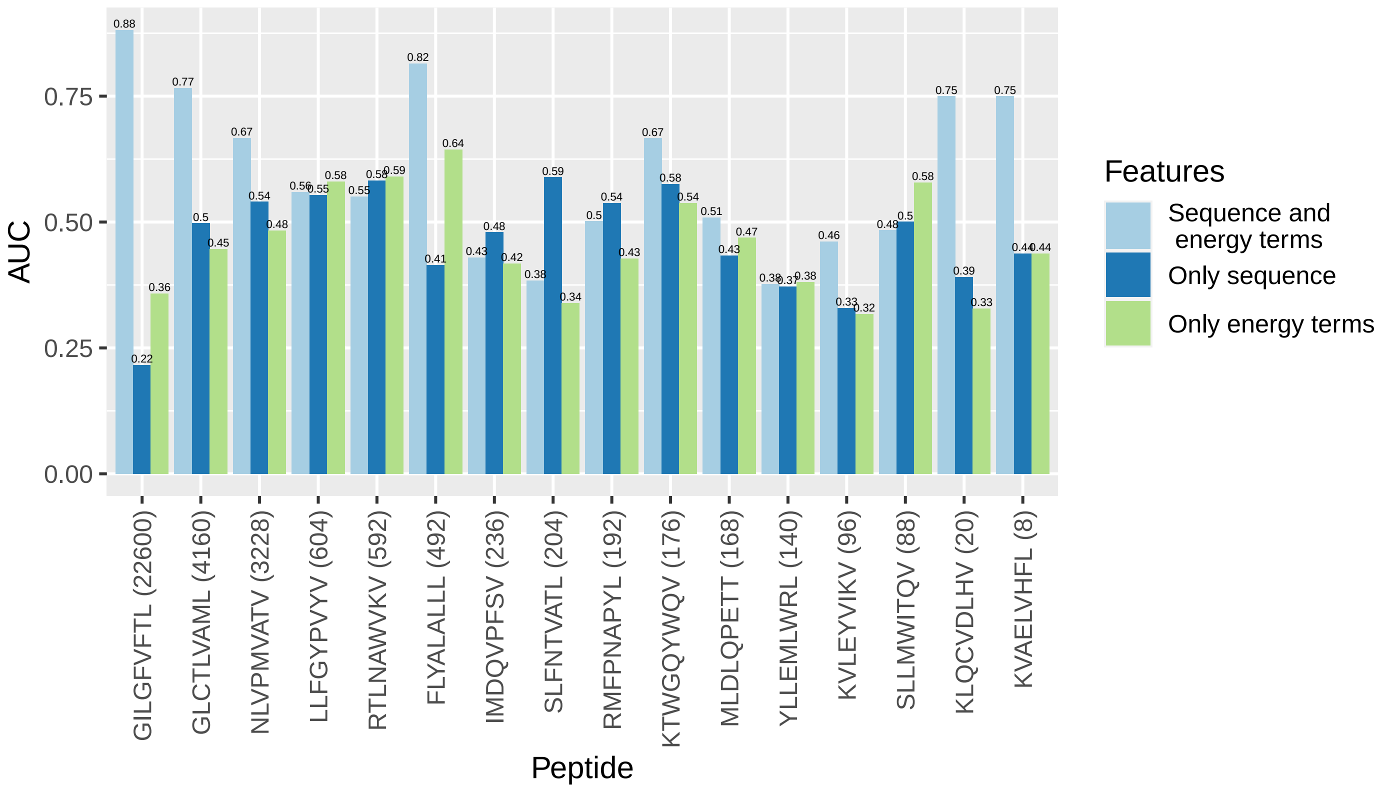
The results from this setup show an enormous drop in performance when training on the sequence only (figure 23). For ”GILGFVFTL” the AUC is 0.22 when training on sequence only while it was 0.90 in the normal training setup. For almost all the peptides, the predictions are no better than random when training on the sequence only. This is a clear proof of our hypothesis that the model was overfitting on the sequence. When the model is not trained on a given peptide, it is not able to make predictions based on only the sequence.

On the other hand, the model turns out to perform much better when including both sequence and energy terms. For example, it achieves an AUC of 0.88 for ”GILGFVTFL” (figure 23a). The MCC for ”GILGFVFTL” when using all features is 0.72, while it is -0.11 when training on sequence only (figure 23b). This shows that adding the energy terms makes the model much better at generalizing.

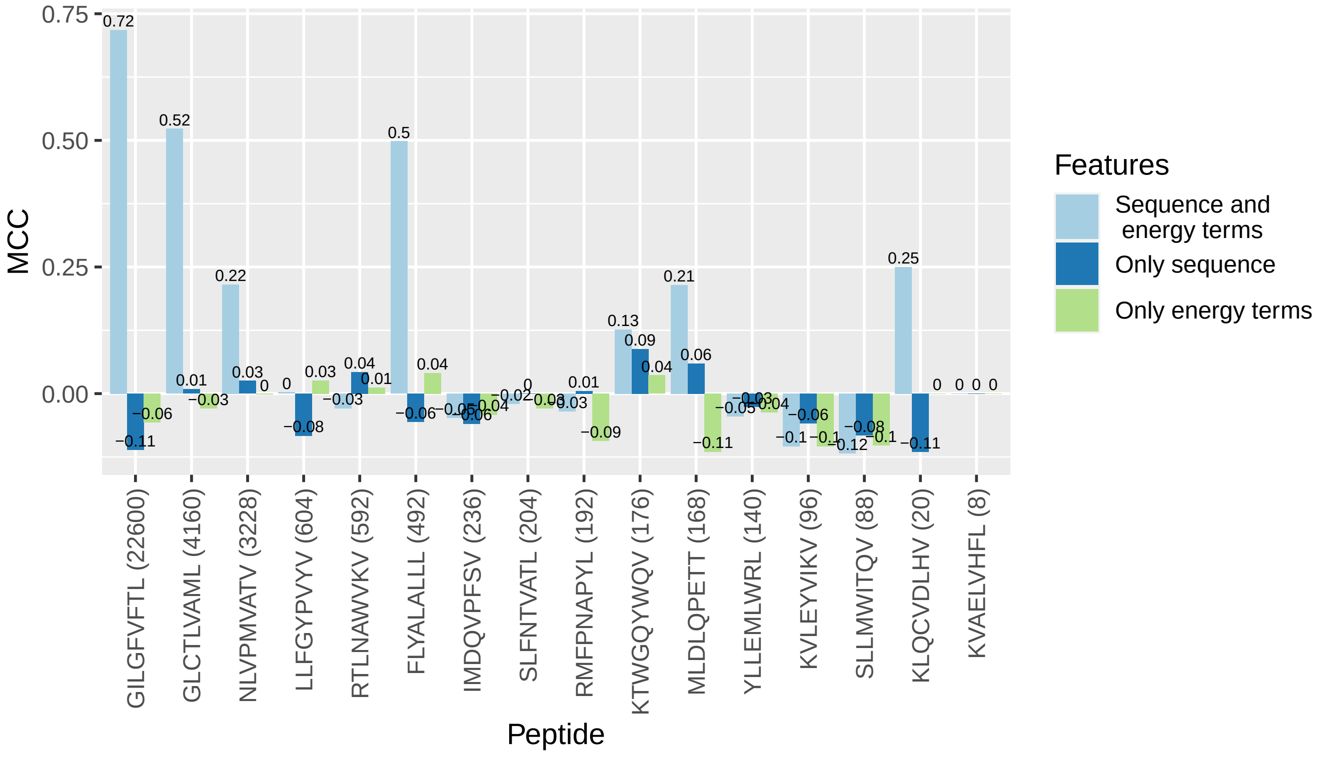
The performance is still best on ”GILGFVFTL”, which is surprising, because the training set in this setup is much smaller. In general the performance is best on the three most frequent peptides and

”FLYALALLL”. This is probably because the bias in germlines is similar for these peptides, thus the model is able to overfit on the germlines even though the relevant peptide is not present in the training set.

To sum up, using a leave-one-out setup, gives a huge drop in performance for sequence only, proving that the sequence-only model was greatly overfitting in the normal training setup. However, including the energy terms makes it much less prone to overfitting.



(a)



(b)

Figure 23: Performance in the leave-one-out setup. For each peptide, the model was trained excluding that peptide from the training set, and the model was then evaluated on that peptide. a) AUC. b) MCC. For most of the peptides, the model performs best when including both the sequence and the energy terms. The swapped negatives were excluded from the test set.

4 Discussion

As we have seen, the results show that including the force field energy terms substantially improves the model's ability to generalize compared to using sequence alone. The sequence-only model showed a very good overall performance, but the performance dropped dramatically in the leave-one-out setup. On the contrary, the performance was stable when including the energy terms in the model. This shows that the energy terms make a big difference for the stability of the model and its ability to generalize.

Comparing our results with the results achieved in [34] on the same dataset, we see similar performances. Their baseline model, which predicts binding based on the highest similarity score to the positives, shows a performance on ”GILGFVFTL” of 0.85 when using CDR3β only and 0.89 when using both α and β. Our model gave a performance on ”GILGFVFTL” of 0.88, thus, better than the baseline model on β only, but slightly worse than when using α and β. However, it is reasonable to expect that our model would perform better on new data, like we saw in the leave-one-out experiment.

The Rosetta per-residue energy terms showed to improve the performance compared to using only global energy terms. The individual Rosetta terms, on the other hand, seemed not to improve performance compared to using only the total energy (further experiments are needed to confirm this).

Our dataset showed to have a strong bias in the germline distributions, which caused the model to overfit. A future experiment could be to try to reduce the bias in the datasat by either downsampling or by generating more swapped negatives. We saw that ”FLYALALLL” had a very large bias in germlines, resulting in a very good performance for this peptide. An option would be to remove some of the ”FLYALALLL” positives from the dataset or exclude ”FLYALALLL” entirely. Including more swapped negatives could be done in order to have a more equal distribution of germlines between the positives and negatives.

However, we expect that more data will become available in future, and that this will have a major impact on the model's ability to generalize.

In general the CNN-biLSTM architecture was shown to be able to learn the patterns in the dataset, but there is still room for improvements. One thing that could be changed is to use a separate CNN for the MHC, peptide, TCRα and TCRβ, similar to the architecture of NetTCR and the Master's thesis by Olsen [36, 33, 34]. In this way the filters will more easily learn to find specific motifs for each subunit. Another thing that could maybe improve the model is to include an attention mechanism after the biLSTM as suggested in [24].

We saw that the model was more stable and less overfitting when we included dropouts and L2 regularization (data not shown). However, we used a dropout probability of only 10% and it is very likely that a higher dropout probability would make the model less prone to overfitting. In future work, it would be beneficial to do hyperparameter optimization of the dropout probability and the weight decay.

Another factor influencing the overfitting is how the partitioning is done. In our dataset, the partitions are made solely based on CDR3 similarity, while all peptides and germlines are present in all the partitions. This makes the model very prone to overfitting. In this dataset, however, we cannot avoid having the same peptides in different partitions, since ”GILGFVFTL” constitutes ∼60% of the entries, but when more data becomes available this would be beneficial. It would also be interesting to investigate how different partitioning thresholds would affect the performance.

Finally, it would be beneficial to do some assessment of the quality of the molecular models. The TCRpMHCmodels paper showed that there is a correlation between template identity and model quality [30]. It would be interesting to correlate the template identity to the performance to see if there is a correlation. Further, one could include the template identity for each complex as a feature in the deep learning model.

5 Conclusion

In this project we have created a benchmarking dataset for TCR-pMHC predictions. We have modeled $\sim$10,000 binding and non-binding TCR-pMHC complexes and calculated their energy terms using the FoldX and Rosetta force fields. We have developed a deep learning model for prediction of binding, consisting of a CNN followed by a biLSTM.

The results show that including energy terms substantially improves the model's ability to generalize compared to using the sequence only. While using sequence only showed a very good overall performance, it gave almost random predictions in a leave-one-out setup, thus proving that it is overfitting. On the contrary, we observed that when including the energy terms, the model showed very good performance even in the leave-one-out setup. Thus, including energy terms makes the model much more stable and better at predicting peptides that it has not been trained on.

We also saw, that including Rosetta per-residue energies increased the performance compared to using global energy terms only. Leaving out MHC, peptide and TCR one-by-one showed that the best performance was achieved when training on only peptide and TCR.

Our dataset was very biased in terms of peptide frequencies and germline distributions, which made the model prone to overfitting. We saw that the model performed best on the most frequent peptides in the dataset and on the peptides that had a clear bias in the germline distribution. However, we expect that as more data becomes available, the model will become better at generalizing.

Bibliography

[1] A. K. Abbas, A. H. Lichtman, and S. Pillai, *Basic Immunology: Functions and Disorders of the Immune System*. Elsevier, fifth ed., 2016.

[2] K. S. Kobayashi and P. J. Van Den Elsen, "NLRC5: A key regulator of MHC class I-dependent immune responses," *Nature Reviews Immunology*, vol. 12, no. 12, pp. 813–820, 2012.

[3] W. K. Wong, J. Leem, and J. Leem, "Comparative Analysis of the CDR Loops of Antigen Receptors," *Frontiers in Immunology*, vol. 10, pp. 1–17, 2019.

[4] B. Al-Lazikani, A. M. Lesk, and C. Chothia, "Canonical structures for the hypervariable regions of T cell αβ receptors," *Journal of Molecular Biology*, vol. 295, no. 4, pp. 979–995, 1987.

[5] B. Al-Lazikani, A. M. Lesk, and C. Chothia, "Standard conformations for the canonical structures of immunoglobulins," *Journal of Molecular Biology*, vol. 273, no. 4, pp. 927–948, 1997.

[6] R. Andersen, M. Donia, E. Ellebaek, T. H. Borch, P. Kongsted, T. Z. Iversen, L. R. Hölmich, H. W. Hendel, Met, M. H. Andersen, P. T. Straten, and I. M. Svane, "Long-Lasting complete responses in patients with metastatic melanoma after adoptive cell therapy with tumor-infiltrating lymphocytes and an attenuated il2 regimen," *Clinical Cancer Research*, vol. 22, no. 15, pp. 3734–3745, 2016.

[7] M. H. Kershaw, J. A. Westwood, and P. K. Darcy, "Gene-engineered T cells for cancer therapy," *Nature Reviews Cancer*, vol. 13, no. 8, pp. 525–541, 2013.

[8] M. H. Kershaw, J. A. Westwood, C. Y. Slaney, and P. K. Darcy, "Clinical application of genetically modified T cells in cancer therapy," *Clinical and Translational Immunology*, vol. 3, no. 5, pp. 1–7, 2014.

[9] T. Kumai, A. Fan, Y. Harabuchi, and E. Celis, "Cancer immunotherapy: moving forward with peptide T cell vaccines," *Current Opinion in Immunology*, vol. 47, pp. 57–63, 2017.

[10] M. A. Curotto de Lafaille and J. J. Lafaille, "CD4+ regulatory T cells in autoimmunity and allergy," *Current Opinion in Immunology*, vol. 14, no. 6, pp. 771–778, 2002.

[11] A. A. Vandenbark, E. Morgan, R. Bartholomew, D. Bourdette, R. Whitham, D. Carlo, D. Gold, G. Hashim, and H. Offner, "TCR peptide therapy in human autoimmune diseases," *Neurochemical Research*, vol. 26, no. 6, pp. 713–730, 2001.

[12] H. M. Berman, T. Battistuz, T. N. Bhat, W. F. Bluhm, P. E. Bourne, K. Burkhardt, Z. Feng, G. L. Gilliland, L. Iype, S. Jain, P. Fagan, J. Marvin, D. Padilla, V. Ravichandran, B. Schneider, N. Thanki, H. Weissig, J. D. Westbrook, and C. Zardecki, "The protein data bank," *Acta Crystallographica Section D: Biological Crystallography*, vol. 58, no. 6 I, pp. 899–907, 2002.

[13] D. A. Benson, K. Clark, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and E. W. Sayers, "GenBank," *Nucleic Acids Research*, vol. 43, no. D1, pp. D30–D35, 2015.

[14] Y. Yang, J. Gao, J. Wang, R. Heffernan, J. Hanson, K. Paliwal, and Y. Zhou, "Sixty-five years of the long march in protein secondary structure prediction: The final stretch?," *Briefings in Bioinformatics*, vol. 19, no. 3, pp. 482–494, 2018.

[15] A. Fiser and A. Šali, "MODELLER: Generation and Refinement of Homology-Based Protein Structure Models," *Methods in Enzymology*, vol. 374, pp. 461–491, 2003.

[16] R. Kaushik, A. Singh, and B. Jayaram, "Ab initio protein structure prediction," *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics*, vol. 1-3, pp. 62–76, 2018.

[17] R. Guerois, J. E. Nielsen, and L. Serrano, "Predicting changes in the stability of proteins and protein complexes: A study of more than 1000 mutations," *Journal of Molecular Biology*, vol. 320, no. 2, pp. 369–387, 2002.

[18] J. Schymkowitz, J. Borg, F. Stricher, R. Nys, F. Rousseau, and L. Serrano, "The FoldX web server: An online force field," *Nucleic Acids Research*, vol. 33, no. SUPPL. 2, pp. 382–388, 2005.

[19] S. Wu and Y. Zhang, "Protein structure prediction," *Bioinformatics: Tools and Applications*, vol. 383, no. 2003, pp. 225–242, 2007.

[20] R. F. Alford, A. Leaver-Fay, J. R. Jeliazkov, M. J. O'Meara, F. P. DiMaio, H. Park, M. V. Shapovalov, P. D. Renfrew, V. K. Mulligan, K. Kappel, J. W. Labonte, M. S. Pacella, R. Bonneau, P. Bradley, R. L. Dunbrack, R. Das, D. Baker, B. Kuhlman, T. Kortemme, and J. J. Gray, "The Rosetta All-Atom Energy Function for Macromolecular Modeling and Design," *Journal of Chemical Theory and Computation*, vol. 13, no. 6, pp. 3031–3048, 2017.

[21] O. Buß, J. Rudat, and K. Ochsenreither, "FoldX as Protein Engineering Tool: Better Than Random Based Approaches?," *Computational and Structural Biotechnology Journal*, vol. 16, pp. 25–33, 2018.

[22] E. Fenoy, J. M. Izarzugaza, V. Jurtz, S. Brunak, and M. Nielsen, "A generic deep convolutional neural network framework for prediction of receptor-ligand interactions-NetPhosPan: Application to kinase phosphorylation prediction," *Bioinformatics*, vol. 35, no. 7, pp. 1098–1107, 2019.

[23] J. J. Almagro Armenteros, C. K. Sønderby, S. K. Sønderby, H. Nielsen, and O. Winther, "DeepLoc: prediction of protein subcellular localization using deep learning," *Bioinformatics (Oxford, England)*, vol. 33, no. 21, pp. 3387–3395, 2017.

[24] V. I. Jurtz, A. R. Johansen, M. Nielsen, J. J. Almagro Armenteros, H. Nielsen, C. K. Sønderby, O. Winther, and S. K. Sønderby, "An introduction to deep learning on biological sequence data: Examples and solutions," *Bioinformatics*, vol. 33, no. 22, pp. 3685–3690, 2017.

[25] T. Liu, H. Xu, M. Ragulskis, M. Cao, and W. Ostachowicz, "A data-driven damage identification framework based on transmissibility function datasets and one-dimensional convolutional neural networks: Verification on a structural health monitoring benchmark structure," *Sensors (Switzerland)*, vol. 20, no. 4, pp. 1–25, 2020.

[26] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[27] A. Sircar, E. T. Kim, and J. J. Gray, "RosettaAntibody: Antibody variable region homology modeling server," *Nucleic Acids Research*, vol. 37, no. SUPPL. 2, pp. 474–479, 2009.

[28] M. A. Messih, R. Lepore, P. Marcatili, and A. Tramontano, "Improving the accuracy of the structure prediction of the third hypervariable loop of the heavy chains of antibodies," *Bioinformatics*, vol. 30, no. 19, pp. 2733–2740, 2014.

[29] M. S. Klausen, M. V. Anderson, M. C. Jespersen, M. Nielsen, and P. Marcatili, "LYRA, a webserver for lymphocyte receptor structural modeling," *Nucleic Acids Research*, vol. 43, no. W1, pp. W349–W355, 2015.

[30] K. K. Jensen, V. Rantos, E. C. Jappe, T. H. Olsen, M. C. Jespersen, V. Jurtz, L. E. Jessen, E. Lanzarotti, S. Mahajan, B. Peters, M. Nielsen, and P. Marcatili, "TCRpMHCmodels: Structural modelling of TCR-pMHC class I complexes," *Scientific Reports*, vol. 9, no. 1, pp. 1–12, 2019.

[31] I. Hoof, B. Peters, J. Sidney, L. E. Pedersen, A. Sette, O. Lund, S. Buus, and M. Nielsen, "NetMHCpan, a method for MHC class i binding prediction beyond humans," *Immunogenetics*, vol. 61, no. 1, pp. 1–13, 2009.

[32] B. Reynisson, B. Alvarez, S. Paul, B. Peters, and M. Nielsen, "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," *Nucleic Acids Research*, vol. 48, no. W1, pp. W449–W454, 2021.

[33] V. I. Jurtz, L. E. Jessen, A. K. Bentzen, M. C. Jespersen, S. Mahajan, R. Vita, K. K. Jensen, P. Marcatili, S. R. Hadrup, B. Peters, and M. Nielsen, "NetTCR: Sequence-based prediction of TCR binding to peptide-MHC complexes using convolutional neural networks," *bioRxiv*, no. 5, 2018.

[34] A. Montemurro, V. Schuster, H. R. Povlsen, and A. K. Bentzen, "NetTCR-2 . 0 : Accurate Prediction of TCR-Peptide Binding is Contingent on Paired TCRα and β Sequence Data," *Submitted*.

[35] E. Lanzarotti, P. Marcatili, and M. Nielsen, "Identification of the cognate peptide-MHC target of T cell receptors using molecular modeling and force field scoring," *Molecular Immunology*, vol. 94, no. September 2017, pp. 91–97, 2018.

[36] T. H. Olsen, "Combining deep learning and structural modeling to predict T-cell receptor specificity," no. July, 2018.

[37] H. Høie, "Deep learning using protein structural modeling for prediction of T-cell receptor and peptide MHC binding," no. June, 2019.

[38] S. Hashemifar, B. Neyshabur, A. A. Khan, and J. Xu, "Predicting protein-protein interactions through sequence-based deep learning," *Bioinformatics*, vol. 34, no. 17, pp. i802–i810, 2018.

[39] R. Vita, J. A. Overton, J. A. Greenbaum, J. Ponomarenko, J. D. Clark, J. R. Cantrell, D. K. Wheeler, J. L. Gabbard, D. Hix, A. Sette, and B. Peters, "The immune epitope database (IEDB) 3.0," *Nucleic Acids Research*, vol. 43, no. D1, pp. D405–D412, 2015.

[40] D. V. Bagaev, R. M. Vroomans, J. Samir, U. Stervbo, C. Rius, G. Dolton, A. Greenshields-Watson, M. Attaf, E. S. Egorov, I. V. Zvyagin, N. Babel, D. K. Cole, A. J. Godkin, A. K. Sewell, C. Kesmir, D. M. Chudakov, F. Luciani, and M. Shugay, "VDJdb in 2019: Database extension, new analysis infrastructure and a T-cell receptor motif compendium," *Nucleic Acids Research*, vol. 48, no. D1, pp. D1057–D1062, 2020.

[41] 10X Genomics, "LIT047_A New Way of Exploring Immunity - Linking Highly Multiplexed Antigen Recognition to Immune Repertoire and Phenotype," no. D, pp. 1–13, 2019.

[42] J. Robinson, D. J. Barker, X. Georgiou, M. A. Cooper, P. Flicek, and S. G. Marsh, "IPD-IMGT/HLA Database," *Nucleic Acids Research*, vol. 48, no. D1, pp. D948–D955, 2020.

Appendices

A Distribution of peptides per partition

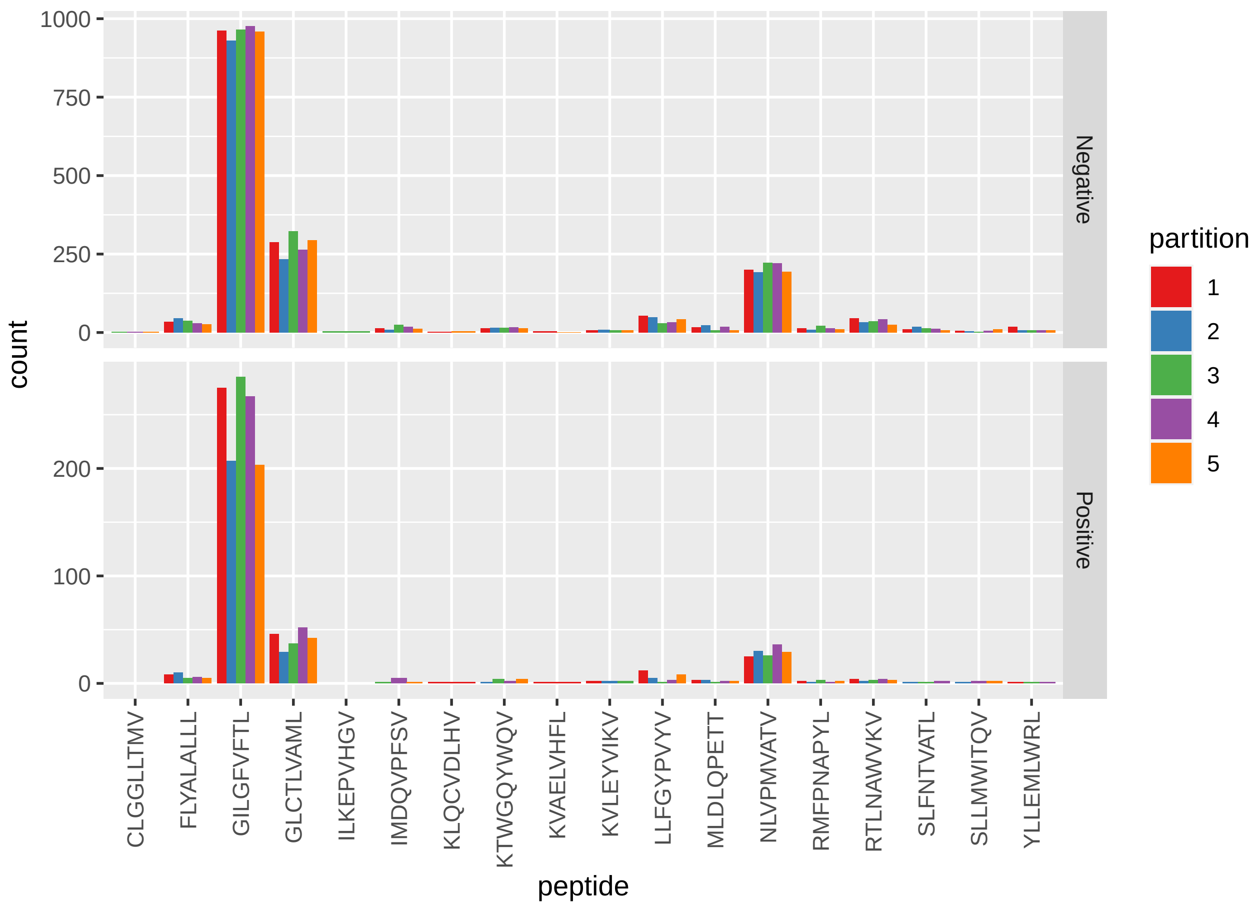
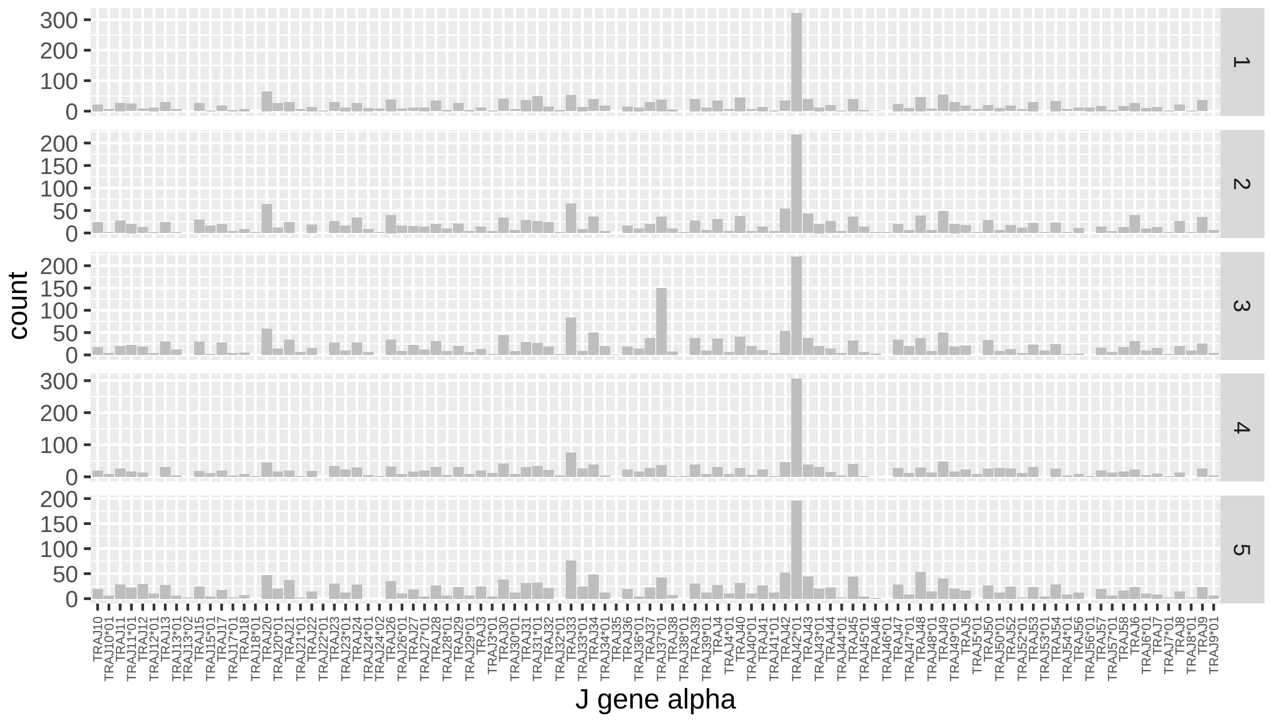
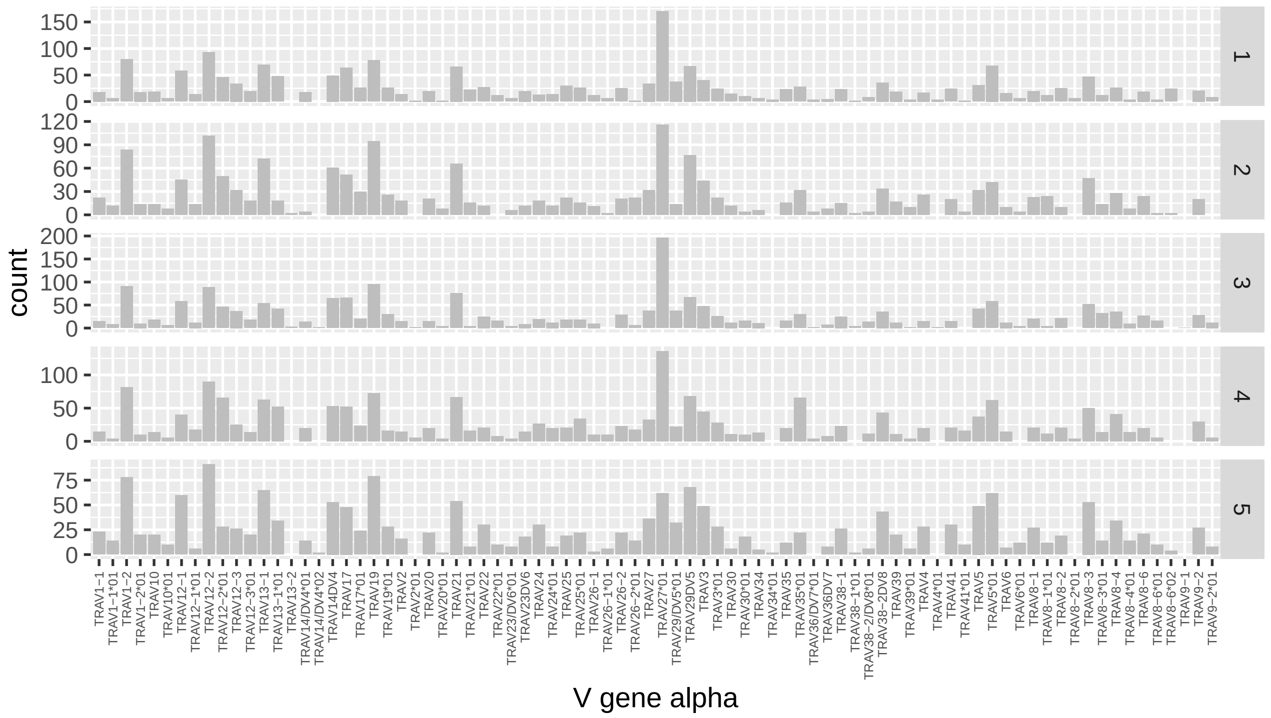


Figure 24: Distribution of peptides per partition. The peptides are present with more or less equal frequencies in each of the partitions.

B Distribution of germ lines per partition

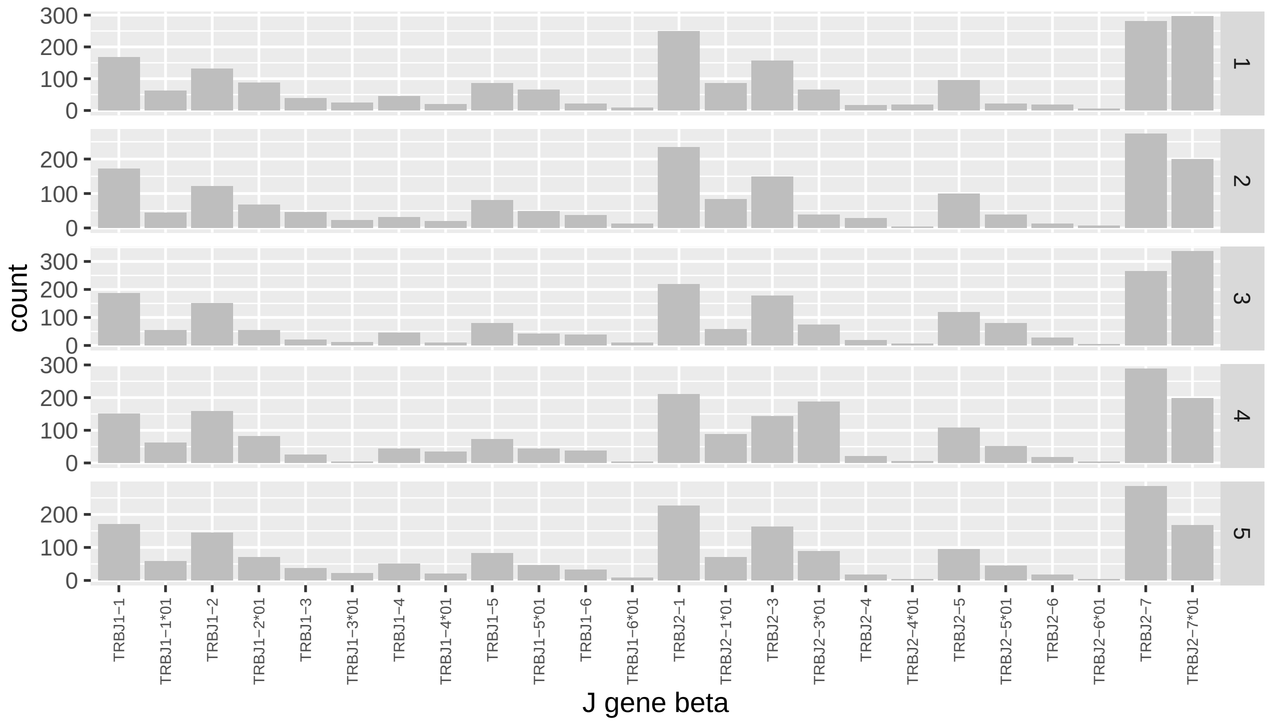


(a)

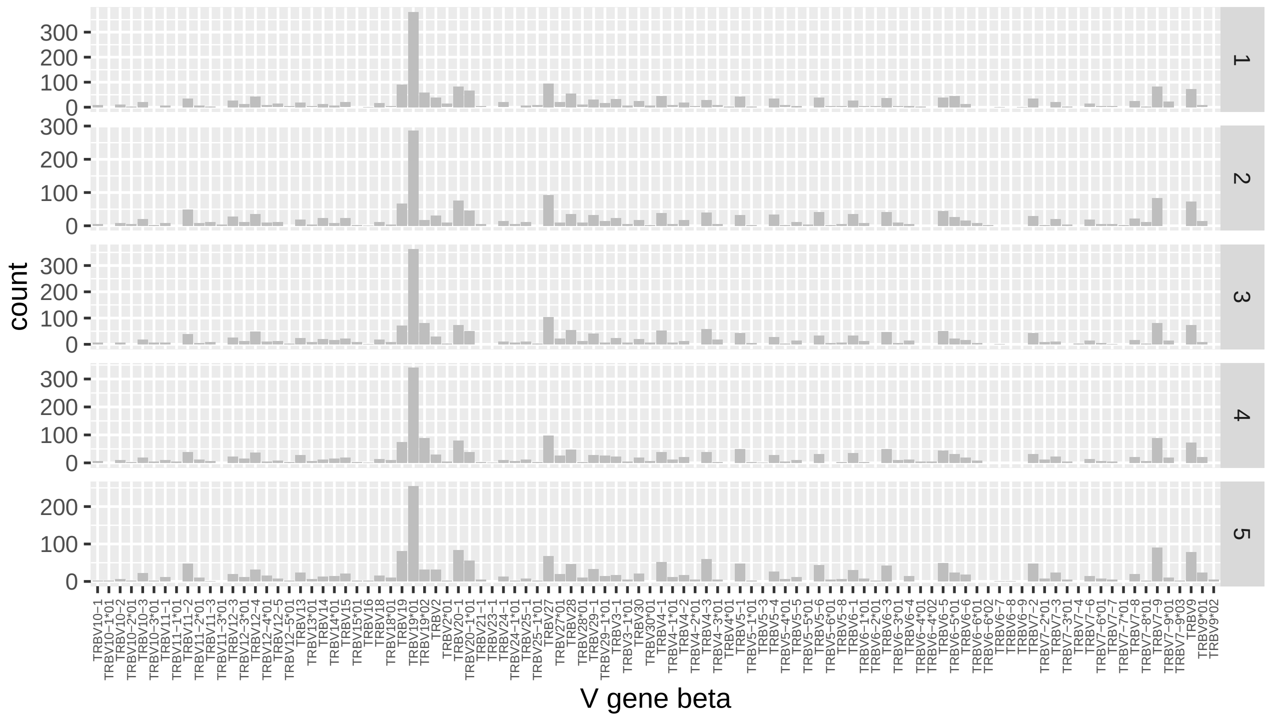


(b)

Figure 25: Distribution of TCRα germlines per partition. a) J gene for TCRα. b) V gene for TCRα. The germlines are present with similar distributions in the partitions.



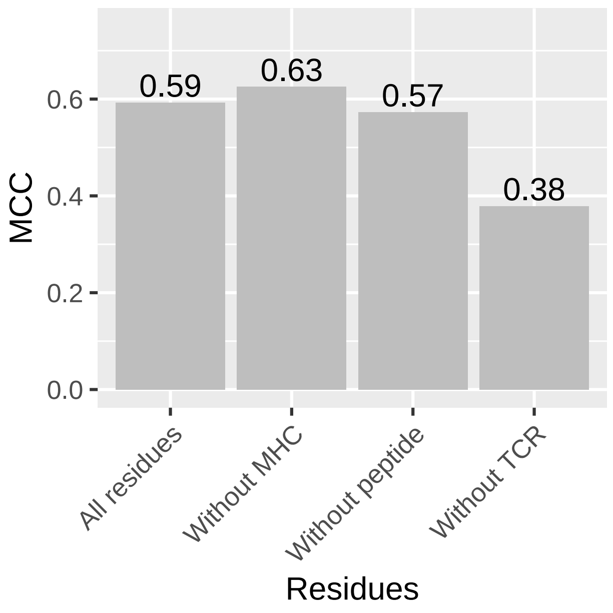
(a)



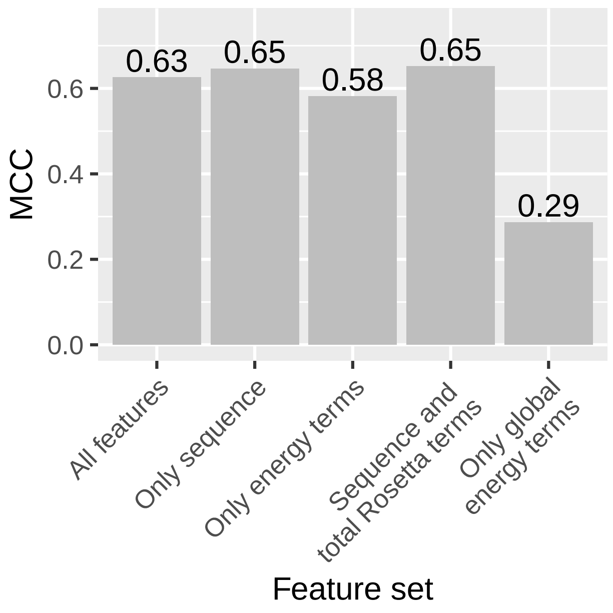
(b)

Figure 26: Distribution of TCRβ germlines per partition. a) J gene for TCRβ. b) V gene for TCRβ. The germlines are present with similar distributions in the partitions.

C MCC for overall performance



(a)



(b)

Figure 27: MCC for overall performance. a) Performance when excluding parts of sequence. The best performance is achieved when excluding MHC. b) Performance on subsets of features. The best performance is achieved when training only on sequence or on sequence and total Rosetta energy terms.