Technical University of Denmark

Deep learning using protein structural modeling for prediction of T-cell receptor and peptide MHC binding

MSc student Magnus Haraldson Høie

Supervisors:
Associate Professor Paolo Marcatili
Principal Investigator Han Dali

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

1st of June 2019

This report is the written result of the MSc thesis 'Deep learning using protein structural modeling for prediction of T-cell receptor and peptide MHC binding', attended at the Technical University of Denmark in June 2019.

Abbreviations

MHC	Major histocompatibility complex
TCR	T-cell receptor
TCR-p-MHC	TCR-peptide-MHC protein complex
Template	Homology modelling template
Pytorch	Python machine learning library
fast.ai	Deep learning library built on Pytorch
CNN	Convolutional neural network
Random forest	Decision tree machine learning method
Learning rate	Model hyperparameter for updating weights
1-cycle policy	Variable learning rate training method
RP	Gaussian noise matrix used to multiply tensor
Accuracy	Ratio of correct over all predictions
AUC-ROC	Area-under-curve Receiver-operating-characteristics
MCC	Matthews correlation coefficient
F1-score	Harmonic mean of precision and accuracy
TPR	True positive rate
Recall	Same as TPR
Precision	Ratio of true positives over all predicted positives
TNR	True negative rate
Specificity	Same as TNR

Summary

The aim of the project was to build a tool to predict the binding between T-cell receptors (TCRs) and peptide-bound major histocompatibility complexes (MHCs) using deep learning. Previous work by Olsen showed improved performance of using deep learning versus simpler feed forward networks when predicting on TCR-p-MHC BLOSUM-alignment scores together with protein global energy terms extracted using FoldX.

A comprehensive data visualization and random forest feature analysis on the original dataset provided by Olsen showed that using BLOSUM-scores as training features provided a strong bias for the negative complexes, and overfitted on negative BLOSUM-scores in the peptide region. The original dataset BLOSUM-scores were replaced with TCR-p-MHC amino-acid sequences and a combination of extracted protein structural features and a subset of available FoldX global energy terms. LOGO-plots and a simple feed forward network showed a significant reduction in bias for the amino-sequence data versus the BLOSUM-scores.

A new convolutional neural architecture was coded from scratch in Pytorch based on a published state of the art protein-protein interaction deep-learning model. The dataset was split into 5 different partitions of less than 80% similarity. 600 different models were independently trained on different combinations of the features.

Combining all features in the training set outperformed the amino-sequences only in 4 out of 5 partitions, reaching top ranking performance in 3 partitions. The energy terms by themselves performed exceptionally well, reaching top performance in 2 partitions and were narrowly outperformed by all features in a third. It was observed that increases in the all-features performance versus amino-sequences only was dependent on a strong performance of the energy terms by themselves. Similar findings have been reported in the literature and are discussed further in the thesis discussion.

Analysis of the enrichment in prediction scores for positive complexes showed that combining all features significantly increased the recall of positive complexes. Both complexes that had significant recall by the amino-sequences and those which had low or no recall by the amino-sequences had significantly increased chances of being correctly predicted as positives.

Further analysis should confirm these findings on an independent test set, and further investigate which local structural and global protein structures can provide significant predictive power by themselves and in combination with TCR-p-MHC sequences.

Acknowledgements

I want to thank professor Paolo Marcatili for his support and enabling me to have autonomy in my approach and direction of this incredible learning experience. Thank you for your creative and clever ideas on tackling roadblocks we faced in the project, and for always pointing me in the direction of the best next steps!

I wish to express gratitude to professor Han Dali for warmly welcoming me to his lab at the Beijing Institute of Genomics Beijing, a truly interesting experience and a stay which enabled me to learn much of the bioinformatic skills this project was dependent on.

I want to give a big thank you and eternal gratitude to Tobias Hegelund Olsen for taking significant time out of his work to both guide me and actively discuss progress during this project. Without your continued support and active sharing of your knowledge this thesis´ swift progress would have been grudgingly slow and painful.

Thank you to Sophie Bögemann and Andreas Fønss Møller for providing me feedback and much needed fresh eyes on the writing of this thesis.

Contents

1 Introduction **1**

1.1 The immune system . . . 1

1.1.1 The innate immune system . . . 2

1.1.2 The adaptive immune system . . . 2

1.1.3 T-cell receptors . . . 3

1.1.4 Major histocompatibility complexes . . . 4

1.1.5 Antigen processing . . . 5

1.1.6 Binding assays . . . 5

1.2 Immunoinformatics background . . . 6

1.2.1 Protein modelling . . . 6

1.2.1.1 Protein energy terms and structural features . . . 6

1.2.1.2 BLOSUM-scores . . . 6

1.2.2 Machine learning . . . 7

1.2.2.1 Feed forward networks . . . 7

1.2.2.2 Convolutional neural networks . . . 8

1.2.2.3 Random forests . . . 9

1.2.2.4 Cross-validation . . . 10

1.2.2.5 Performance measures . . . 10

1.3 TCR-p-MHC deep learning model by Olsen . . . 12

2 Methods **14**

2.1 Random forest analysis of provided dataset . . . 14

2.2 Choice of structural features . . . 14

2.3 Simple feed forward networks . . . 17

2.4 Design of new CNN architecture . . . 17

2.5 Model performance measures . . . 18

2.6 Data visualization . . . 19

3 Results **20**

3.1 Original dataset summary and performances . . . 20

3.2 Dataset visualization . . . 21

3.3 Random forest analysis on BLOSUM-data . . . 27

3.4 LOGO plots of peptide sequences . . . 30
3.5 Setting baseline with simple feed-forward network . . . 31
3.6 New CNN architecture . . . 31
3.7 CNN amino-sequence performance . . . 33
3.8 CNN combined features performance . . . 35
3.9 Visualizing model predictions . . . 37
3.10 Prediction enrichment graphs . . . 39
3.11 Effects of model quality . . . 40

4 Discussion **41**

5 Conclusion **43**

6 Future perspectives **44**

7 References **45**

8 Appendix **48**

8.1 Randomized parititon heatmaps . . . 48
8.2 PDB data [28] . . . 52
8.3 Feature importance [28] . . . 53

1 Introduction

The aim of the project was to build a tool to predict the binding between T-cell receptors (TCRs) and peptide-bound major histocompatibility complexes (MHCs) using deep learning. While several machine learning models already exist for the prediction of which peptides can be presented by MHCs, the specificity of TCRs to those peptides is much less well studied. Previous work by Olsen showed improved performance of using deep learning versus simpler feed forward networks when predicting on BLOSUM-alignment scores together with protein global energy features. This project develops further on this idea by addition of new protein structural features and integrating the complex amino-acid sequences. The new dataset is then trained on a previously published, state of the art protein-protein interaction deep learning network.

A few key questions answered in this project are:

1. To what extent can deep learning networks (CNNs) improve prediction performance over simpler machine learning models such as random forests and feed forward networks?
2. What predictive power can be obtained from training on the amino-acid sequences of the TCR-peptide-MHC complexes by themselves?
3. Does training on additional structural or global protein energy terms in combination with the amino-acid sequence of TCR-peptide-MHC complexes increase predictive power?
4. How predictive are structural features and energy terms by themselves?
5. How does training on these features combined change the recall of known positive and negative complexes?

Before introducing the dataset and previous work by Olsen, a brief introduction of the immune system, immunoinformatics, protein modelling and deep learning is given.

The following is a brief introduction to the immune system, with a special focus on T-cell receptors, major histocompatibility complexes (MHCs) and antigen processing. The summary is largely based on the 9th edition of Janeway´s immunobiology, by Kenneth Murphy [1].

1.1 The immune system

The immune system is an organism s crucial line of defense against disease, infection and foreign particles. It faces the challenging task of distinguishing its own healthy cells, tissue and microbiome from foreign ones, including viruses, bacteria, foreign particles and its own diseased or dysfunctioning cells. This recognition of “self” from “non-self” is done through recognition of antigens - molecules which elicit an immune response and are associated with pathogens or foreign particles.

It is crucial for the organism s immune system to be able to distinguish normal, healthy tissue (self) from dysfunctional or infected tissue or foreign pathogens. An overactive immune system responding to self-antigens can lead to allergies and auto-immune diseases, while an under-active immune system leaves the organism vulnerable to infections, harmful substances and diseased or cancerous cells.

The immune system is traditionally split into two main parts, the innate immune system, responsible for a quick and general response, and the adaptive immune system which has a more tailored but delayed response.

1.1.1 The innate immune system

The innate immune system is the first line of defense against disease and pathogens, and forms a primary and non-specific response to acute infections. It consists of both physical and chemical barriers including the skin and mucous membranes, and gastric acid and flushing in the cases of tears and saliva. Both the innate and the adaptive immune system can be further divided into cell-mediated and humoral parts.

Immune cells known as phagocytes as part of the cell-mediated response, and pro-inflammatory soluble factors which are responsible for stimulating an inflammatory response (humoral response). Together they drive recruitment of immune cells towards sites of infection and provide degradation and removal of foreign particles.

The phagocytes include macrophages which engulf and digest foreign particles, and dendritic cells which process antigens and present them on the cell surface for recognition by T-cells. They are all leukocytes (white blood cells) and derived from differentiating hematopoietic stem cells.

Processing and presentation of antigens is a highly important link between the innate and adaptive immune system, which learns to recognize and memorizes antigens. This allows for a specialized or adaptive response to future infections, providing immunity for the organism.

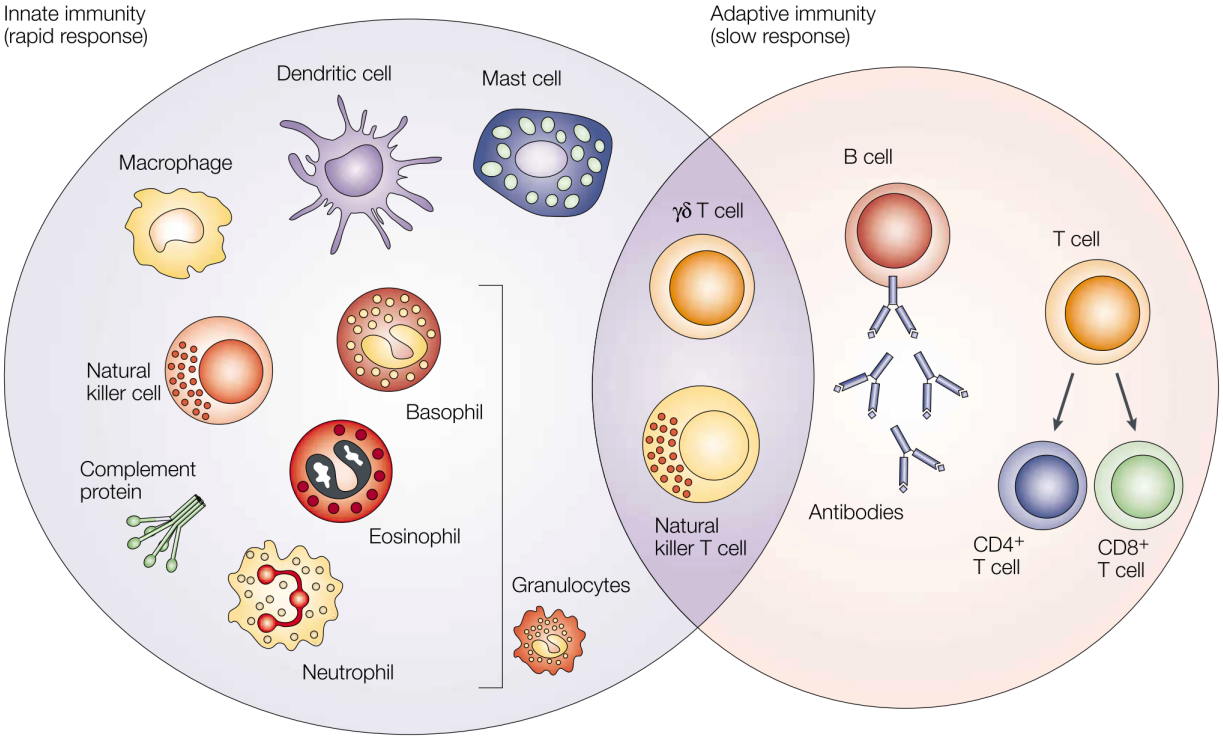


Figure 1: An overview of the immune system divided into two parts, the innate immunity and the adaptive immunity. Adapted from [2].

1.1.2 The adaptive immune system

While the innate immune system provides a generalized response towards all pathogens, the adaptive immune system provides a long-term and pathogen or antigen specific response. The adaptive immune system works as an immunological memory of past infections, providing an increased response to pathogens that are encountered again, providing

the basis for immunity towards specific pathogens through recognition of their antigens. This also forms the basis for vaccinations.

The cell-mediated part of the adaptive immune system consists of two types of cells, "memory" B-cells and T-cells, which are also kinds of lymphocytes or white blood cells. B-cells are part of the humoral immunity protecting extracellular space, and are responsible for producing soluble, antigen recognizing receptors known as antibodies. Antibodies recognize and stick to pathogens, both directly inhibiting their function and stimulating an immune response. B cells also act as professional antigen presenting cells which are able to activate the T-helper cells.

T-cells can differentiate into two different types of cells. Cytotoxic T-cells (CD8+ T cells) and T-helper cells (CD4+ T cells) scan cells for immunogenic antigens, and kill diseased or malfunctioning cells. T-helper cells are responsible for producing pro-inflammatory cytokines when they detect immunogenic antigens. T-cells have highly adaptable T-cell receptors (TCR) which are able to recognize specific antigens or peptide fragments presented on cell-surface proteins known as major histocompatibility complexes (MHCs).

1.1.3 T-cell receptors

Most T-cell receptors consist of an alpha and beta chain, which are responsible for recognizing antigens (Figure 2). Some TCRs have less well studied gamma and delta chains which will not be covered here . These chains must be able to specifically recognize an extremely diverse array of possible peptides, but are derived from a limited number of genes. In order to increase their genetic diversity, they employ several strategies.

Somatic variation is achieved through random genetic rearrangement through a process known as V(D)J recombination. This generates unique T-cell receptors through random combination of the variable (V), diversity (D) and (J) joining genes, allowing multiple segments of V, D and J for beta chain, and V and J segments for the alpha-chain. Diversity of TCRs in the population is further achieved through a high degree of genetic polymorphism, with a large degree of single nucleotide variations maintained in the genetic pool [3]. TCR binding has been shown to be specific enough that a single amino-acid substitution of the TCR sequence can deactivate response to a specific antigen [4].

The alpha and beta chains are again divided into a variable region involved in binding the peptide-MHC and a constant region which binds the TCR to the T-cell. The variable region has three genetically hypervariable domains known as complementarity determining regions (CDRs), CDR1, CDR2 and CDR3. CDR3 is thought to be primarily responsible for the antigen binding, partially together with CDR1, and CDR2 is thought to recognize the MHC.

Due to recent advancement in high throughput sequencing methods, there has been significant interest in performing machine learning predictions directly on the CDR regions. The NetTCR web-server predicts binding directly on TCR CDR3 sequence for the most common MHCI HLA-A*02:01 allele, using a convolutional neural network[5]. This has the advantage of providing far more data points for training prediction models, but has the disadvantage of not capturing information in the rest of the sequence which likely affects binding.

Altogether, these combinations allows for diverse generation of up to 10 to 100 million unique TCR combinations in humans [3]. The high binding specificity of TCRs likely limits T-cells to only binding 1 of 100,000 peptide-MHCs [4].

T-cells go through a process of maturation in the thymus. Negative selection destroys T-cells which bind too strongly to self-MHC-peptides and would lead to an undesirable auto-immune response. Positive selection destroys T-cells which do not have a sufficient response to presented peptide-MHCs and fail to develop a binding TCR. In this way, only T-cells which are specific enough to bind to peptide-MHCs and which do not cross-react too strongly to any “self-antigens” are left. Up to 98 % of thymic T-cells are filtered out, leaving a selection of around 2 % which go on to become functional T-cells [6].

Recognition of a specific MHC-peptide by a TCR then leads to activation of the T-cell. This causes a reaction which ultimately leads to induced apoptosis (programmed cell death) of the diseased cell. While immunological databases such as VDJdb (mentioned later) show clear examples of multiple TCR sequences binding to the same peptide, the extent of TCR cross-reactivity with multiple antigens is not well understood. Numerical models suggest some degree of cross-reactivity is a necessity to allow for sufficient recognition of the vast array of potential pathogen antigens [7], with recent studies investigating the extent of cross-reactivity using libraries of barcoded peptide-MHCs [8].

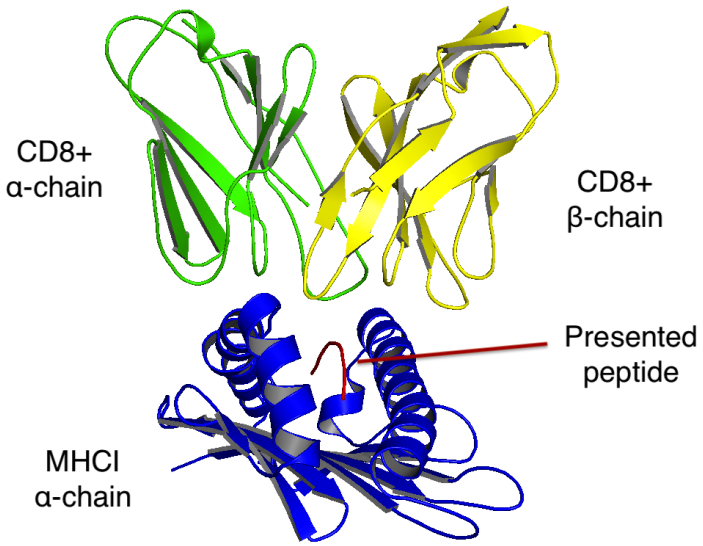


Figure 2: Sample structure of the MHC-peptide TCR complexes taken from the training models. Shown above is a positive binding of MHC class I HLA-02, presenting peptide ILAKFLHWL in a complex with a cognate CD8+ T-cell receptor (PDB structure [9]). The MHC binding pocket is shown inside its alpha-chain in the lower part of the image, with a bound peptide. The TCR alpha and beta chains are shown in green and yellow.

1.1.4 Major histocompatibility complexes

MHCs are cell surface proteins responsible for presenting antigens for recognition by CD4+ or CD8+ T-cells. These antigens can either be presented as peptide fragments originating from within a cell. MHC class I (MHCI) presents peptide fragments from inside the cell, and can be recognized by CD8+ T-cells. MHC class II (MHCII) presents peptide fragments derived from engulfed pathogens, and shown on the cell surface of antigen-presenting cells. These are recognized by CD4+ T-cells.

MHCs must be able to not only present an extremely diverse array of potential antigens, but also elude pathogens developing resistance towards presentation of their antigens. This is achieved by several genetic strategies. MHC diversity is reached through them being polygenic (constructed from multiple, interacting genes), polymorphic (a set of genes can produce many versions) and co-dominant (both alleles of a gene contribute contribute to the MHC phenotype. [1].

MHCI consist of two polypeptide chains, an alpha chain (shown in figure 2) encoded by the HLA-gene and a beta-microglobulin chain encoded by the beta2-microglobulin gene. The alpha-chain can be divided into 3 domains, 1, 2, and 3. Together, the 1 and 2 domains form a groove that allows binding of the peptide, and provides a binding site for the TCR while scanning the bound peptide. The thesis focuses on binding between TCRs and peptide bound MHC class I, and will therefore focus on MHCI. After the introduction, this will only be referred to as MHC.

1.1.5 Antigen processing

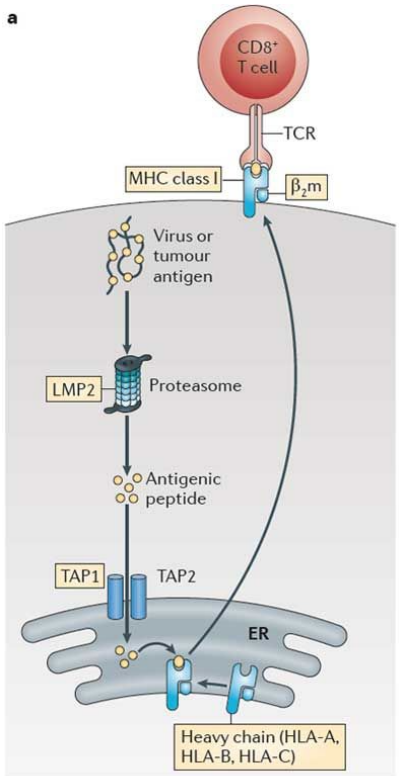


Figure 3: breakdown of proteins in the proteasome leads to generation of peptide fragments which bind to MHCs. Adapted from [10]

MHCI presents peptides mainly derived internally from the cell from protein degradation in the proteasome. The process of extracting for later presentation of the produced peptide fragments is known as the antigen presenting pathway. The proteasome mainly functions to provide amino-acid building blocks for synthesis of new proteins, but some peptide fragments are transported away by TAP (transporter associated with antigen processing). These are brought to the endoplasmic reticulum.

The peptides are then loaded onto inherently unstable MHC molecules with the assistance of several other molecules. MHCs also go through a maturation process, where MHCs binding and presenting self-antigens are degraded. MHCs which fail to bind to a peptide or bind too poorly are also degraded. Only peptide-MHC complexes which are stable and have strong enough binding to antigens are made ready for transportation to the cell surface for recognition by passing CD8+ T cells. It should be noted that only a minor fraction, around 2 % [11] of peptides can be presented this way.

Computational models for prediction of protein peptide fragments which are likely to bind to different MHC types have had some success, with the DTU NetMHCpan web server providing the basis for generation of artificial negative complexes in the data provided for this project[12].

The presented peptides are generally 8-10 amino acids long, due to space constraints in the binding pocket, but longer peptides up to 14 amino acids long have been observed to produce a T-cell response [13].

1.1.6 Binding assays

While computational prediction models of peptide MHC binding (especially of MHCI) have achieved significant success, the success of models predicting binding of TCRs to their cognate peptide-MHC targets has been limited. This is in large part due to a lack of data, however there have been recent developments in high-throughput sequencing using DNA barcodes and multiplexing [8].

VDJdb is a curated online database of TCR sequences and antigen specifics with over 10 000 TCR sequences, 100 antigens and 40 MHC alleles [14, 15]. The database provide antigen specific TCR sequences from peptide-MHC multimer-labeled T-cell populations and in vitro assays of T-cell binding to specific peptide-MHC expressing targets. Most of the sequences are however limited to the general MHC type and the CDR regions of the TCRs.

The full peptide and MHC alpha chain, and TCR alpha and beta-chain data which formed the basis for this projects dataset, was downloaded from Immune Epitope Database (IEDB) [16, 17]. IEDB provides data related to T cell epitopes, with a special focus on pathogenic epitopes. It also contains MHC-binding data.

1.2 Immunoinformatics background

Below is a brief introduction to the data and tools used in protein modelling, the principles used in random forests, feed forward and convolutional neural network machine learning models, examples of their use for prediction on biological data, and the evaluation of model performances.

1.2.1 Protein modelling

Protein sequences can be identified relatively easily using high throughput sequencing tools such as Next Generation Sequencing or more precise Sanger sequencing. Identifying protein 3-dimensional structures is however far more costly and challenging, requiring careful use of X-ray crystallography or NMR as well as suitable processing of the purified protein first. this makes physical determination of protein structures costly. Also, predicting protein structures from their sequences alone, for example based on their energy landscape, has traditionally proved challenging and unreliable for anything but short protein sequences. However significant conservation of protein structure has been observed in evolutionarily related (homologous) proteins.

This can be exploited for modelling proteins by taking known structures of proteins or protein domains and using them to model sequences which do not have a known structure. The higher the similarity of the template protein is to the sequence to be modelled, the higher structural similarity it is likely to have. This means that the ability to model unknown sequences greatly increases the larger the database of available structures becomes.

Tools such as MODELLER [**modeller**] are widely used in protein modelling, and exploits conservation of protein structure along with algorithms for taking into account spatial restraint, in order to generate artificial structures for given sequences.

1.2.1.1 Protein energy terms and structural features It has been previously shown in work by Lanzarotti et al that TCR-p-MHC binding can be predicted with significant accuracy using only protein energy terms [18]. It is generally accepted that protein structures fold into their lowest potential energy states. As a corollary, calculation of a protein structure s potential energy can be used as an indirect measure of the model s quality or similarity to the real structure.

Tools such as Rosetta [19], and FoldX [20] based on empirical measurements of protein force fields, use algorithms for calculating the free energy based on a protein structure. Due to this, FoldX is used a protein modelling tool, where mutation of residues and changing of orientation and bonding angles allows optimization of a model (energy minimization) to increase its quality and therefore accuracy in imitating a real world structure.

The FoldX algorithms allow for calculation of energy terms such as the folding Gibbs free energy, electrostatics, van der Waals forces, covalent bonds, loop entropies and more (see appendix). Crucially, this information can encode for the likelihood of a structure being likely to exist biologically, or an unlikely one. This can therefore aid a model in making predictions.

The protein structure encodes information such as the secondary structure and conformation of residues, the relative accessibility and their carbon contact number. Given the necessity of having a protein structure for calculating its energy terms, these structural features provide additional information which can potentially aid a protein-protein interaction prediction model.

1.2.1.2 BLOSUM-scores BLOSUM-scores are assigned values for estimating the similarity between two residues or two sequences in a sequence alignment. Higher scores indicate that the residues are identical or share a high physico-chemical similarity, while lower scores indicate that they are very different from each other.

BLOSUM-scores are assigned based on a BLOSUM matrix, which is an empirically calculated table with values between 11 and -4 for the substitution of any of the 20 amino acids with any of the other 20 amino acids. The original matrix was calculated based on the frequency of amino acids in very conserved groups of proteins, and their amino-acid substitution probabilities [21].

1.2.2 Machine learning

Machine learning describes of mathematical models used for predicting some target output based on an input of training data. Machine learning models are first train on the provided input data, and initially generate a random output. A function is applied to give the algorithm feedback on how wrong its output was, and this feedback is applied to change the parameters of the model to increase the accuracy of the output. This is described as "learning".

Machine learning models can largely be divided into three categories; supervised, unsupervised and reinforcement learning. Only supervised learning will be discussed here. This project aims at designing a binary classification predictor, namely the prediction of either positive or non-positive binding of a TCR with peptide-MHC. As the target output values are known, this describes a supervised learning problem. Below three different types of supervised machine learning models will be described.

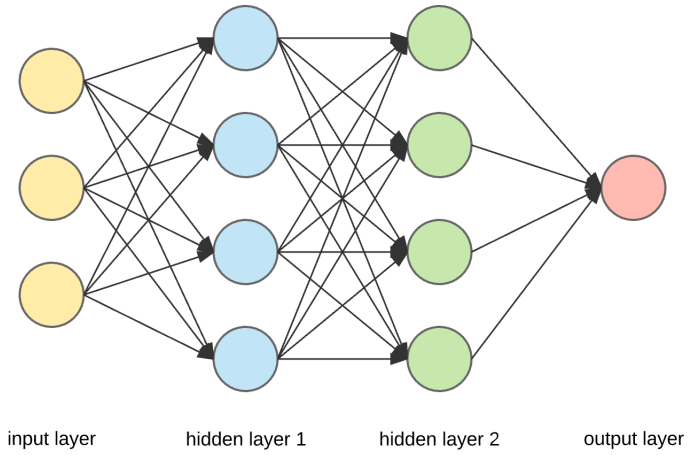


Figure 4: A fully connected, feed forward network. Adapted from [22]

1.2.2.1 Feed forward networks Feed forward networks are largely inspired by the biological brain. Feed forward networks are divided into layers of nodes, where each node is connected to all nodes in the layer before it. These nodes are often referred to as "neurons", given the similar architecture as the brain's neurons connect to each other. The network is said to be fully connected in that the each layer´s nodes is connected to all previous and succeeding layers´ nodes.

Information passes through the network as numerical values, where each node sees the sum of outputs from all its connected nodes from the previous layer. At each node, this sum, along with a constant bias value and a given weight or factor that the sum is multiplied with, passes through an activation function. This activation function calculates a given mathematical function, and produces an output value which is then passed deeper into the network. The end layer must reduce its input into the final output, which is the target value that should be reached. By manipulating the model´s parameters, its weights and biases, an input (e.g. a protein sequence) can be reduced to its target output (positive or negative prediction).

The first pass throughout the network produces a random prediction, as the model s parameter hav only been randomly initialized and not further adjusted. After each batch or pass of training samples, the model calculates the difference between the target value it should have predicted, and its actual output. This is done using a loss function. The output value of the loss function is is then used to update all of the network s parameter, aiming to reduce the loss for the next pass. When this process is repeated many times, the network gradually learns to produce an accurate output matching the real target values.

An example of a feed forward network used in immunoinformatics is the NetMHCpan web-server, which is based on peptide and MHC binding affinity data [23]. It also formed the basis for generation of negative complexes in the provided dataset for this project.

1.2.2.2 Convolutional neural networks

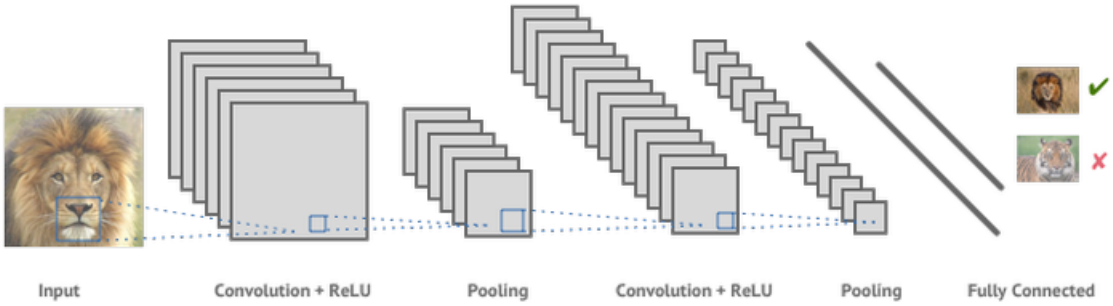


Figure 5: An overview of the use of convolutional filters and pooling layers in a convolutional neural network. An input image is through a series of convolutions and pooling layers. These finally reach one or several fully connected layers, which reduced the image to a single output. Shown is a binary classification model predicting all images as either tiger or lion. Figure adapted from [24].

Convolutional neural networks have shown impressive, state of the art performance in a range of problems, such as image classification and natural language processing for translation or voice recognition. Within biology, they have successfully been applied towards protein folding, prediction of protein binding, prediction of contact maps and protein-protein interactions [25].

Convolutional neural networks do not use layers which are fully connected to each other, and instead make use of moving filters (convolutions) which pass over the input. This has the advantage of allowing the model to scan over its input, e.g. a sequence, and learn to focus on specific features it can extract, such as sequence motifs.

Taking the example of an input image, these filters of a given height and width sequentially pass over the image, stepping a given number of pixels at a time. At each stop, the filter reduces its input (all pixels within the filter area) into a single value, which is then appended to a feature map. Once the entire image has been passed over, a full feature map is produced. This process can be repeated many times, focusin on separate features each time to produce different feature maps.

These feature maps can then be down-sampled through a process known as pooling. This also uses a filter which passes over the input, but reduces the input dimensions while aiming to preserve the most important information of the feature map by averaging or taking its maximum values. This makes further convolutions of the down-sampled layers easier for the next convolutional layers, while also enabling them to capture features further away from each other.

Convolutional neural networks always have a feed forward network at the end, which takes the feature maps and reduces them to the target outputs. Training of the model parameters is done in the same way that the parameters (weights and biases) of a feed forward network are updated.

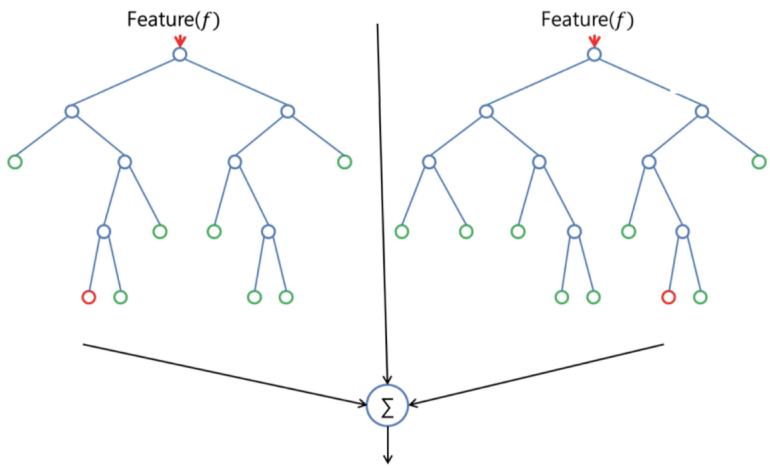


Figure 6: A random forest model, here consisting of two decision trees. Each tree produces an output prediction (red or green) based on its input of features. The output of the two trees are then averaged to produce the final prediction. Figure adapted from [26]

1.2.2.3 Random forests The inner-workings of convolutional neural networks are traditionally considered challenging to interpret. Random forests however have a large benefit of providing powerful tools such as feature importance and dependence graphs in order to justify which features it considers and how its final predictions are calculated based on those decisions. Another benefit is that they are easy to apply and often do not require much tuning of the model´s hyperparameters (model parameters set before training begins) to get great results.

Random forests are based on decision trees. An input dataset of numerical features is provided. At each branching in the tree, the model calculates which feature at which threshold will best be able to separate the data into two parts, by trying all possible features at all possible thresholds. The best separation is achieved when the averaged values for each divided part has the lowest absolute mean difference from the target value. This process is repeated as many times as is necessary until the dataset cannot be divided further. Higher accuracy can be achieved by using multiple trees trained in different ways, and averaging their final predicted outputs.

When predicting, the input passes through the tree and lands at one of the final nodes and is assigned an output value or classification. An understanding of how each feature contributes to the predictions can be calculated based on the rank of which decisions (features) the networks finds the most important for separating the data into its classes or values.

The main disadvantage of using random forests is their inability to extrapolate outside of its training data to predicting targets it has never seen before. They also lack the ability to scan over and interpret feature maps in the way that CNNs can, and have a powerful ability to overfit on their data. While a convolutional neural network can be fed data of any shape, a random forest only accepts data in the form of a 2-dimensional matrix.

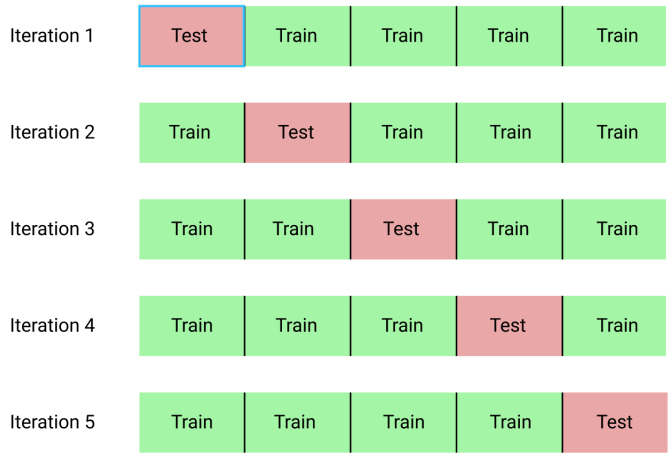


Figure 7: Training and testing on the entire dataset using 5-fold cross-validation. Figure adapted from [27]

1.2.2.4 Cross-validation The main goal of training machine learning models is to have them train on a training dataset and then generalize well on data the model has never seen before. To test its ability to predict on new data, a validation or test set which the model has never seen before is used, in order to calculate the model´s predictive performance.

Cross-validation describes a process of dividing the dataset into a given number of folds, then systematically allowing each fold to be the test set while training on the remaining folds as the training set. This has the benefit of allowing both training and validation on all parts of the dataset, allowing use of all the data and also removing the need to introduce a biased decision of which part of the dataset to leave out for the test set.

It is particularly important to ensure that the folds are not too similar to each other, by clustering of similar samples in the same fold. This helps prevent over-fitting, which is when the model learns to memorize the training data, preventing it from learning more general patterns and therefore likely performing poorly on new data.

1.2.2.5 Performance measures In order to understand how well a model is performing, it is essential to understand the most often used performance measure measures used to evaluate a model. Below is a brief overview over the performance measures used for evaluating machine learning models.

Matthews Correlation Coefficient (MCC If the dataset is unbalanced so that 5 out of 6 samples are negative, both MCC and F1-score will punish the prediction of "easy" true negatives and highly reward prediction of true positives. This is highly useful for getting an idea of how well models performs independent of how unbalanced the validation or test sets happens to be.

MCC is highly useful for evaluating performance on unbalanced datasets, and tries to reduce the confusion matrix of true positive, negative and false positives and negatives to a single number. A score of 1.0 or -1.0 is a perfectly positive or negative predictor, while a score of 0 has no predictive power. MCC indicates how well the model is recalling true positives and negatives out of all true positives and negatives.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN))}} \tag{1}$$

F1-score F1-score measures a test's accuracy, and balances the recall of positives and negatives with the precision. It indicates how much you can "trust" the model's predictions, as oppposed to how many of the true labels it is able

"catch" or recall. E.g. a model recalling 5 out of 10 positives will have a recall of 50 %, but if the model also predicts 20 negatives as false positives, the model´s precision is only $\frac{5}{25}$ 20 %.

F1-score takes both of these numbers into account. A score of 1 is a perfect predictor while a score of 0 indicates no predictive power.

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

Other performance measures

Accuracy is the ratio of correct predictions over all predictions.

$$Accuracy = \frac{TP + FN}{P + N} \tag{3}$$

Recall is the capture of positives out of all the positives in the dataset. Also known as the True Positive Rate.

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

True Negative Rate is the ratio of correct predictions over all predictions.

$$TNR = \frac{TN}{N} \tag{5}$$

Precision is the ratio of true, predicted positives out of all predicted positives.

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

1.3 TCR-p-MHC deep learning model by Olsen

This project is a continuation of the work of Olsen in "Combining deep learning and structural modeling to predict T cell receptor specificity" [28]. Olsen successfully developed a deep learning model for prediction of TCR-p-MHC binding based on the complexes BLOSUM-alignment scores compared to their homology templates and FoldX extracted energy terms. The convolutional neural network architecture coded in TensorFlow was adapted based on the design used for the NetTCR web-server ("NetTCR: sequence-based prediction of TCR binding to peptide-MHC complexes using convolutional neural networks")[5].

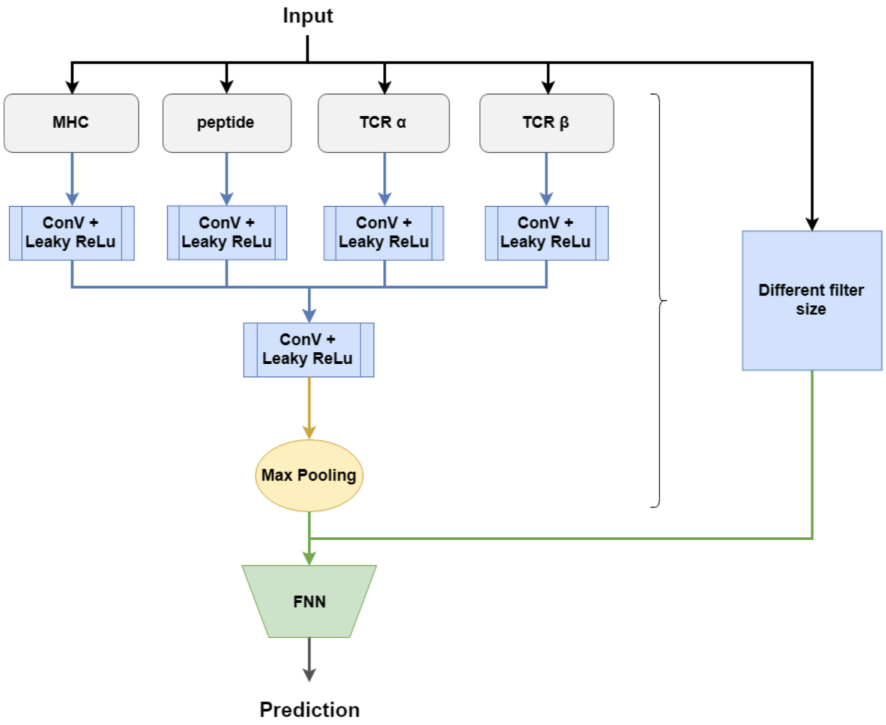


Figure 8: Overview of original neural network architecture by Olsen. The model first separates the complex MHC, peptide, TCR-alpha and TCR-beta parts into 4 convolutional pipelines followed by Leaky ReLU. All convolutional layers are setup with sequential filter sizes increasing from 1, 3, 5, 7 to 9. Finally the outputs from the layers are concatenated, run through another series of 5 convolutional layers with the same filter sizes. The output is then maxpooled and finally fed to a linear layer (feed forward network) for prediction of either positive or negative complex. Figure adapted from [28]

The network architecture uses a series of separate convolutional pipelines which work independently on the MHC, peptide, TCR-alpha and beta parts of the complex. This has the benefit of unambiguously splitting the full complex into its individual parts, and prioritizes allowing the model to understand the individual parts first before they are concatenated and convolutions are run on the full complex. The model was run as a nested cross-validation on 5 separate partitions, with an average MCC and AUC of 0.167 and 76 %.

Table 1: Original training data features as used in the results of Olsen in "Combining deep learning and structural modeling to predict T-cell receptor specificity.

Feature	Channels	Note
Per position BLOSUM62 substitution scores	3	Calculated from BLOSUM62 substitution matrix based on per-position amino-acid and their homology template models. 3 separate channels for substitution based on entire complex template, and only peptide-MHC template and only TCR template (with padding of score 1 for missing positions).
Global template identity	1	0 - 100 %, ratio of identical amino acids between models and modelling templates.
FoldX Energy terms	9	Top 9 rank, as ordered by random forest feature importance analysis by Olsen. Based on global protein interaction energies by FoldX algorithm [25,26]
Total:	**13**	Input size: 1464 **X** 468 **X** 13

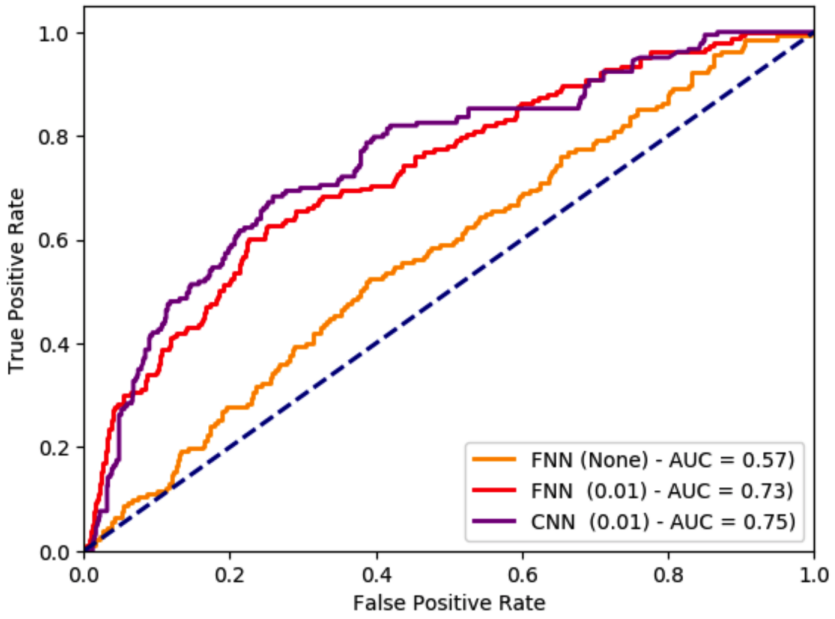


Figure 9: Original CNN model performance scores by Olsen as AUC-ROC curve. True positive rate: 7.6 %, True Negative rate 99 %, MCC 0.167, average precision 0.053. Also shown as the performances of the network when only predicting using the last feed-forward part. Figure adapted from [28]

The dataset provided for this project was generated by an in-house pipeline by Olsen. All models were generated based on the full complex sequences and homology modelling using MODELLER (see appendix Table 8.2 for an overview of models). Thresholds of up to 80, 90, 95 and 99 % template identity was allowed for modelling. Calculation of energy terms was first done after minimizing the structure energies in FoldX using the RepairPDB command [28].

2 Methods

This section will describe the dataset features extracted, the methods used for training the models and the producing the graphs shown in the results.

2.1 Random forest analysis of provided dataset

Random forest analysis was done using Scikit-learn´s *RandomForestRegressor* [29] with settings n_estimators = 150, max_features = 0.5.

Input was provided as a randomized 80 % of the available rows in the training data, using *fastai.structured set_rf_samples* [30]. Hyperparameters were determined experimentally. The random forest training data was split into four groups:

1. Complex BLOSUM62 scores, the BLOSUM62 substitution scores for the full complex sequence compared to the full complex homology modelling template.
2. Peptide BLOSUM62 scores, by isolation of peptide region
3. Only FoldX energy terms, training on 24 available energy terms
4. Complex BLOSUM62 + energy terms

A single run was done on each of the original partitions using each of the feature datasets, and performance scores averaged across all partitions. No features were removed before training to increase random forest performance.

Dependence plots

Dependence plots were constructed based on the trained random forest model on the original dataset. First the highest importance features were identified from the trained sklearn *RandomForestRegressor* model, and visualized using fast-ai´s structured.py rf_feat_importance [30]. The dependence plot for the top 2 feature importance (position 186 and 185) were graphed, with the average feature values vs the average y_true label.

```
imp = str(round(float((fi[fi.loc[:,"cols"].str.contains("185")].imp)), 4))
(ggplot(x_all, aes("185", 'y_true'))
+stat_smooth(se=True, method='loess')
+ ggtitle("Peptide position 185 (" + imp + ") value contribution")
+ xlab("Position BLOSUM62-score")
+ ylab("y_pred contribution"))
```

2.2 Choice of structural features

A large bias inherent in the original dataset BLOSUM62 substitution scores (see results) necessitated the use of new features. The original amino acid sequences were used, which had already been aligned and padded to a total sequence length of 468 amino acids.

In the amino-acid training data, each complex is encoded as numpy arrays of size 468×20, with the first dimension representing the sequence position, and the second dimension 20 channels; one for each amino acid (as a 1-hot encoded sparse matrix). This per-position way of representing features means that new per-position channels can very easily be added to the training data without changing the architecture of the network.

In order to easily extract per-position structural features, a github available PDB file processing tool called PDB_Tool by Sheng Wang [31]. PDB_Tool automatically extracts structural features as from the structural encoding in .PDB files (in a similar way that Pymol would read a .PDB file). It can extract features including [31]:

1. *AMI - amino acid sequence (20 states)*
2. *CLE - conformational letter sequence in 17-states*
3. *ACC - relative solvent accessibility value in 3-states*
4. *Carbon contact number with carbon-α and carbon-β; and co-ordinate of carbon-α and carbon-β*

Table 2: Overview of all available training data features as channels. The total size of the training data was [number of complexes × sequence length × feature channels], or a total size of [1464 × 468 × 122]. It was later found that optimal model performance was reached using a combination of all features with only the first top 24 FoldX energy terms (80 feature channels).

Feature	Channels	Note
Amino-acids	21	20 residues +1 channel for missing residue due to padding one-hot encoded
Conformational letter sequence	17	One-hot encoded (PDB_Tool)
Secondary structure element	8	One-hot encoded (PDB_Tool)
Relative solvent accessibility	3	One-hot encoded (PDB_Tool)
Percent solvent accessibility	1	Value from 0 - 100 (PDB_Tool)
Carbon-a and b contact number	2	Value from 0 - ∞ (PDB_Tool)
TCR:p:MHC protein type	3	Specifies whether residue is a part of the TCR, peptide or MHC
Residue sequence position	1	Encoded as $\frac{position + 1}{sequence\ length + 2}$
FoldX energy terms	66	Protein interaction energies based on FoldX protein design algorithm values (see intro)
Total:	**122**	1464 **X** 468 **X** 122

All added features were L2-normalized using *sklearn.preprocessing.normalize* across the dataset, so that all values were between 0 and 1. While neural networks are still able to learn when the data is not properly normalized, this can severely slow down training and result in convergence to a higher training loss and therefore less predictive model.

```
from sklearn import preprocessing
X_normalized = preprocessing.normalize(X, norm='l2')
```

In addition, the training made use of a variable learning rate and fast.ai´s implementation of the 1-cycle policy [32]. This has been shown to both lead to faster training and acts a regularization method, increasing generalization performance between the training and validation data [33].

The learning rate was determined using fast.ai´s learning rate finder and set to alternating cycles of 1e-02 to 1e-01 and 5e-03 to 5e-02. Loss was calculated using Pytorch´s *nn.CrossEntropyloss* [34], with torch.optim.Adam as optimizer.

Early-stopping was used to save the model at the points of highest performance, measured by MCC-score, while continuing training until 20 epochs had passed.

Training was done on a GPU cluster server provided by DTU Compute, on a Nvidia Titan X GPU, 2 x Xeon E5-2620 CPUs and 128 GB host RAM.

2.3 Simple feed forward networks

Two simple feed-forward networks were constructed to test the validation set performances for the single channel BLOSUM62 dataset and the 21 channel amino-acid dataset. A single layer of fully connected neurons was used for both networks. Input features were flattened to a single vector of size 468 $\times$ 1 for the BLOSUM62 data and 468 $\times$ 21 for the amino-acid data. All data was normalized as part of the pre-processing and batch-normalized.

Training was done using 20 epochs with early stopping and the 1-cycle policy.

2.4 Design of new CNN architecture

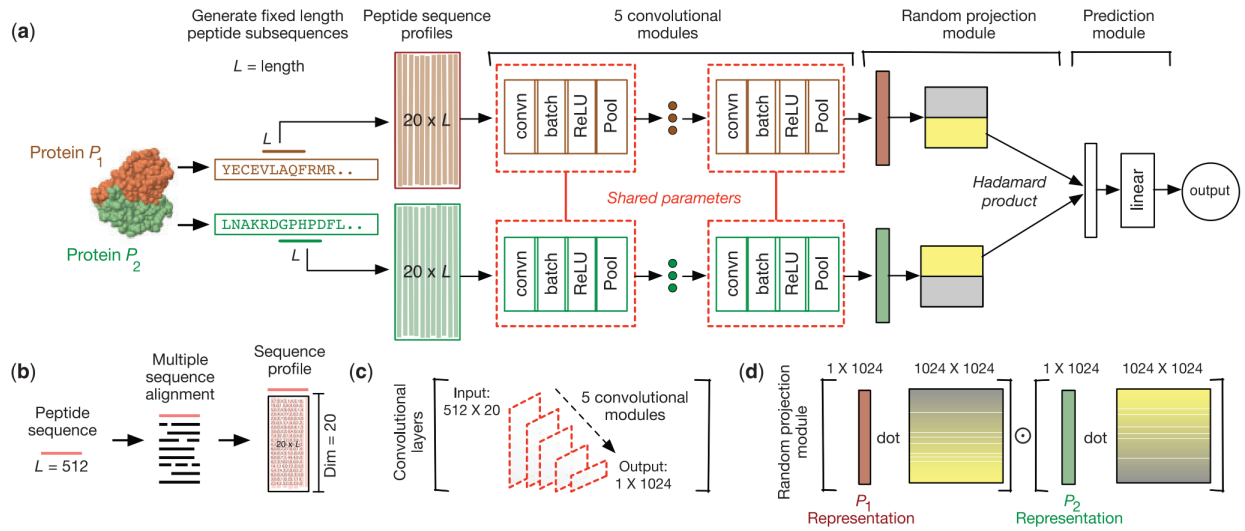


Figure 10: Basis of design of new architecture, from Hashemifar et al 2018, "Predicting protein-protein interactions through sequence based deep-learning". Neural network design based on a "Siamese-like convolutional neural network", splitting two proteins into two identical sub-networks on which a series of convolutions on the amino-acid sequence is performed before the resulting outputs are multiplied together. Figure adapted from [35]

Each of the 5 convolutional modules follows a series of convolution, batch normalization, ReLU and average pooling. The final pooling layer causes a halving of the input tensor along the sequence-length dimension (e.g. 512 $\times$ 20 to 256 $\times$ 20). An input of size 512 $\times$ 20 finally becomes a tensor of size 16 $\times$s 20. This tensor is then multiplied with a "random projection" weight matrix, which simulates a constant gaussian noise the protein vector must overcome to successfully decrease the model´s loss (and increase performance).

The random projection (RP) module consists of two gaussian noise matrices which are "inverted" in one pipeline compared with the other (see figure 10). This is achieved by first generating two gaussian noise sub-matrices, numbered 1 and 2. These are then stacked on top of each other in the order of 1, 2 for the first RP matrix and 2, 1 for the second RP matrix. As each protein representation sees inverted versions of the two RP matrices, by overcoming their noise in inverse ways they also simulate an interaction between them, e.g. between the MHC:peptide and TCR. This has been shown to significantly increase performance in a state of the art protein-protein interaction model [35].

Finally the two protein vector representations are multiplied together element-wise (Hadamard product), before the final vector is fed to a feed forward layer and predicted as either a positive or negative.

The design of the pipeline was borrowed and coded from scratch in PyTorch. The pipeline was primarily modified in six ways:

1. The use of per-position structural features and global FoldX energy terms in addition to just the amino-acid sequences, added as additional channels along the sequence.
2. The two pipelines were separated so they could be trained independently and did not share parameters. This allows specialization pipelines to their respective parts, the top part always receiving the MHC:peptide features and the bottom the TCR features.
3. Residual addition in the first four modules. Addition is done of the previous module´s output to the next, before average pooling. This follows the idea of ResNet [36], where residuals increase performance and allows training of deeper networks, by alleviating the problem of vanishing gradients [37]
4. The use of drop-out layers before average-pooling in each convolutional module (dp = 0.2) to increase generalization performance.
5. The random projection module was found to empirically lower the model´s recall score and slightly lower the MCC-score. It was therefore decided to not use it.
6. A descending kernel-size for each module, starting at 11 and decreasing by 2 for each module, until a minimum of 3.

2.5 Model performance measures

All model predictions, targets and performance scores were recorded in .CSV files for later retrieval and statistical analysis. All analysis was facilitated using Numpy [38] and Pandas [39] in a Jupyter Lab Notebook [40].

Mann-Whitney U-testing was using the Scipy package [41] *scipy.stats.mannwhitneyu*. Default settings were used. The two groups to be compared had all values flattened into a single vector before calculation.

AUC, MCC, F1-score, Average precision, TPR and TNR were all calculated using sklearn metrics functions and ran using default settings [29].

2.6 Data visualization

Data visualization was performed on the predictions and recorded performance scores in the logged .CSV files. Visualization was done using PlotNine [42] in Python.

Heatmaps

Heatmaps of the TCR, peptide and MHCs were constructed by first generating a per sequence comparison numpy matrix of pairwise alignment scores using Biopython´s pairwise2 [43]. Each part (p1 and p2) was calculated with the following scores for a match, mismatch, opening and extension:

```
import pairwise2
pairwise2.align.globalms(p1, p2, 1, 0, -1, 0)
```

Only taking the dataset sequence into account, this resulted in a 366 $\times$ 366 numpy matrix with alignment scores between 0 and 1. Clustering such that high scoring sequences were placed close to each other was done by individually sorting rows and columns in descending order per partition. The resulting sorted matrix was then visualized in PlotNine using *geom_tile()* and *coord_equal()*.

LOGO plots

Logo plots were constructed using mpl_logoplot and default settings [44]. Peptide regions from positions 181-192 were extracted from the complexes and for each dataset, for each of the 11 available positions the frequency of each residue was counted in an 11 $\times$ 21 numpy array. The array was then normalized and plotted:

```
seq = array
seq = seq / np.sum(seq, axis=1, keepdims=True)
seq = np.array(list(map(lambda x: np.where(x == 0, 0.01, x), seq))
entropy = np.array([1.]*21)
fig, ax = plt.subplots()
fig.set_size_inches(12, 3)
mpl_logoplot.logoplot(ax, seq, 'ARNDCQEGHILKMFPSTWYVX',
height=entropy, color='protein')
ax.set_xlabel('Position')
ax.set_ylabel('Entropy [Bits]')
fig.tight_layout()
```

Model prediction graphs

The prediction graphs were generated by first training 20 independent models per validation for every feature dataset. The count of all positive predictions was extracted and then matched and added to a sorted dataframe with an organized order of all target complexes. In this way the counts for the positive, negative, random and switched groups could easily be visualized.

Enrichment graphs were made by subtracting the comparison dataframe with the other, and then plotted in the same way.

3 Results

This section will present the results from the project, and start by summarizing the the dataset provided by Olsen.

The full dataset was initially constructed based on 61 known positive matches between TCR-p-MHCs complexes with known structure. These were identified and downloaded using IEDB (Immune Epitope Database) [17].

As most of the data provided for the project was based on the previous work by Olsen, significant effort was spent on visualizing and understanding the data provided. Below is a summary of the positive and negative complexes, their MHC, peptide and TCR parts and their template homology modelled structural models.

3.1 Original dataset summary and performances

The full dataset consisted of:

- PDB file structure and sequence of 1464 MHC, peptide and T-cell receptor protein complexes, divided into 244 sets of 6 complexes each.
- Each set consisted of:
  - 1 positive complex from a known MHC-p-TCR match with available structure (downloaded from IEDB)
  - 5 negative complexes made from an in-house pipeline:
    - 3 **High Rank%** negatives (labelled N1, N2 and N3) with changed peptide estimated to have a high binding affinity by by NetMHC
    - 1 **Low Rank%**, "random" negative (labelled R1) with switched peptide estimated to have a low binding affinity by NetMHC
    - 1 **Switched TCR** (S1), randomly switched peptide taken from the other positive complexes.
- For each complex, four structural models were generated, based on homology modelling templates allowed a threshold of up to 99, 95, 90 or 80 % threshold of template identity.
- The 468 sequence long complexes were aligned and padded, consisting of the MHC alpha chain (α-1 and α-2, positions 0-181), peptide (181-192) and the TCR variable domains, alpha (192-330) and beta chain (330-468) (Figure 2.
- In total, 1464 models were available with their full sequences and structures in .PDB file format.

Further analysis showed that of these there was a total of:

- 166 unique peptide sequences
- 49 unique T-cell receptor sequences (alpha and beta variable domains)
- 15 unique MHC sequences (alpha-1 and 2 interacting domains)

3.2 Dataset visualization

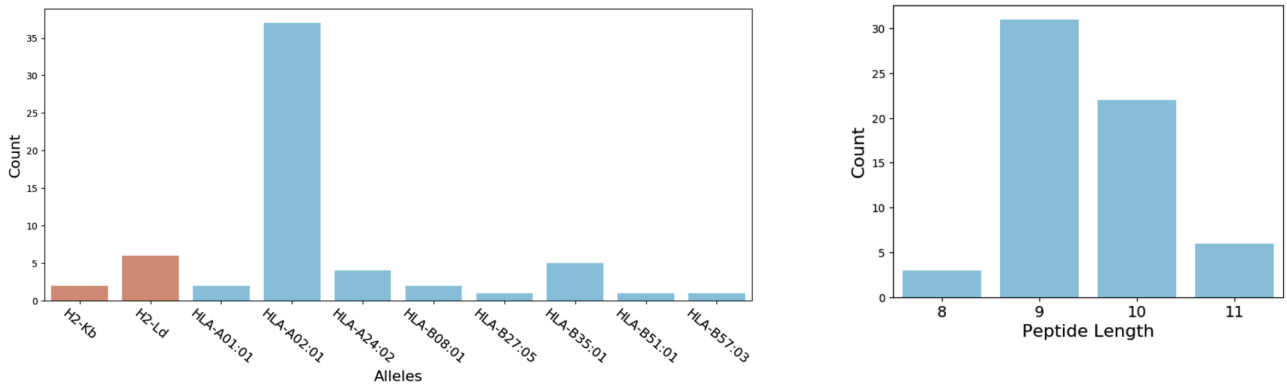


Figure 11: Overview of MHC types (a) and peptide lengths (b). Figure adapted from Olsen. [28]

The 61 sets of complexes were partitioned into 5 partitions. This was achieved by first grouping the positive complexes by 70 % sequence similarity (giving $\frac{1}{3}$ weight to the TCR, peptide and MHC), producing 20 groups. Of these 20 groups, the 5 partitions were made by equally assigning 4 groups to each partition.

As there was a limited number of peptides available for the 366 combinations of TCR-p-MHC complexes, the distribution of peptides within each partition were visualized.

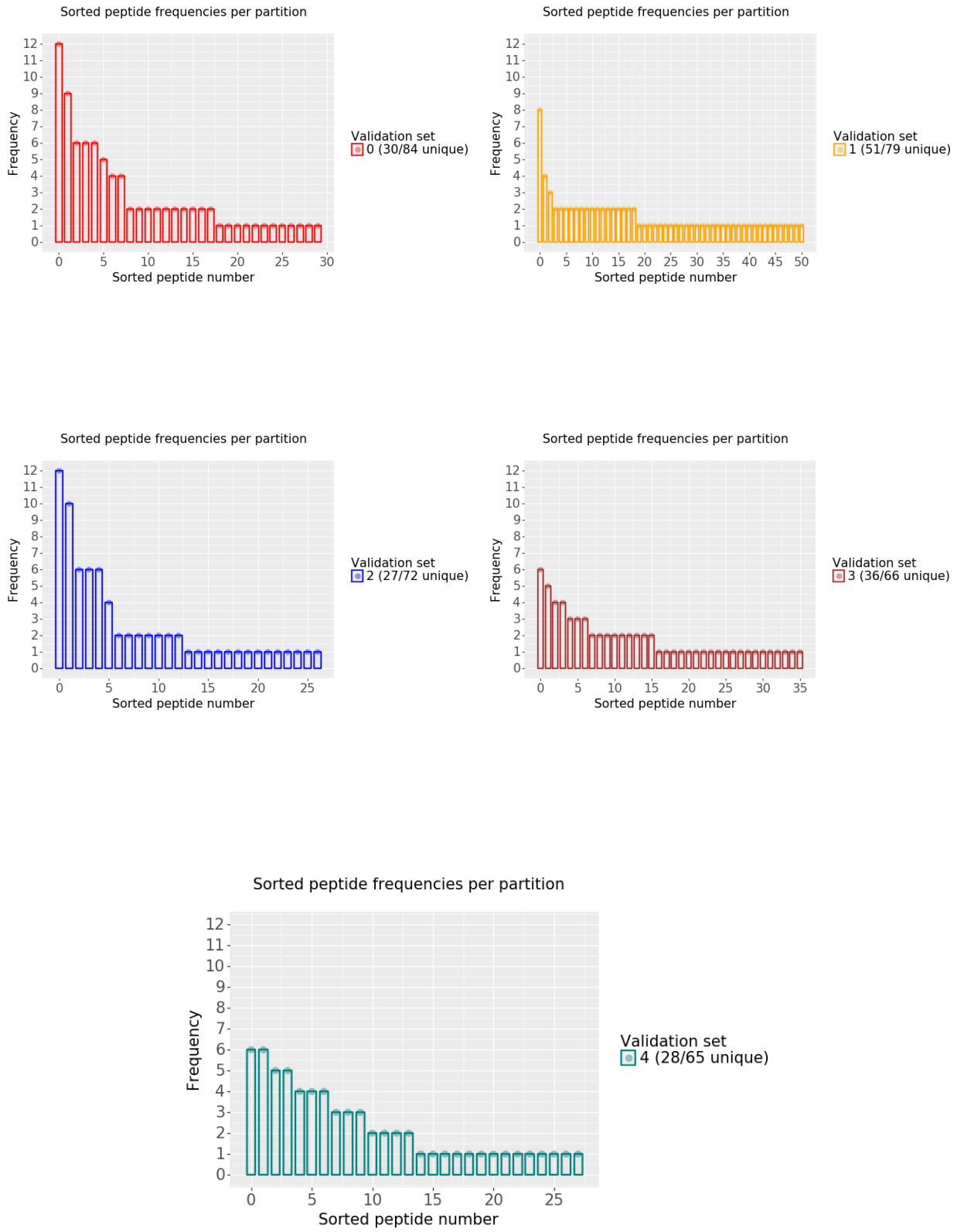


Figure 12: Graph showing the count per peptide ID in each partition. Peptides were ordered in descending order with the highest count (duplicates of the same peptide) first. The peptide IDs are not shared between partitions, although the actual peptides can be. The count of individual or unique peptide sequences out of the total set of peptide sequences including duplicates is shown in the legend.

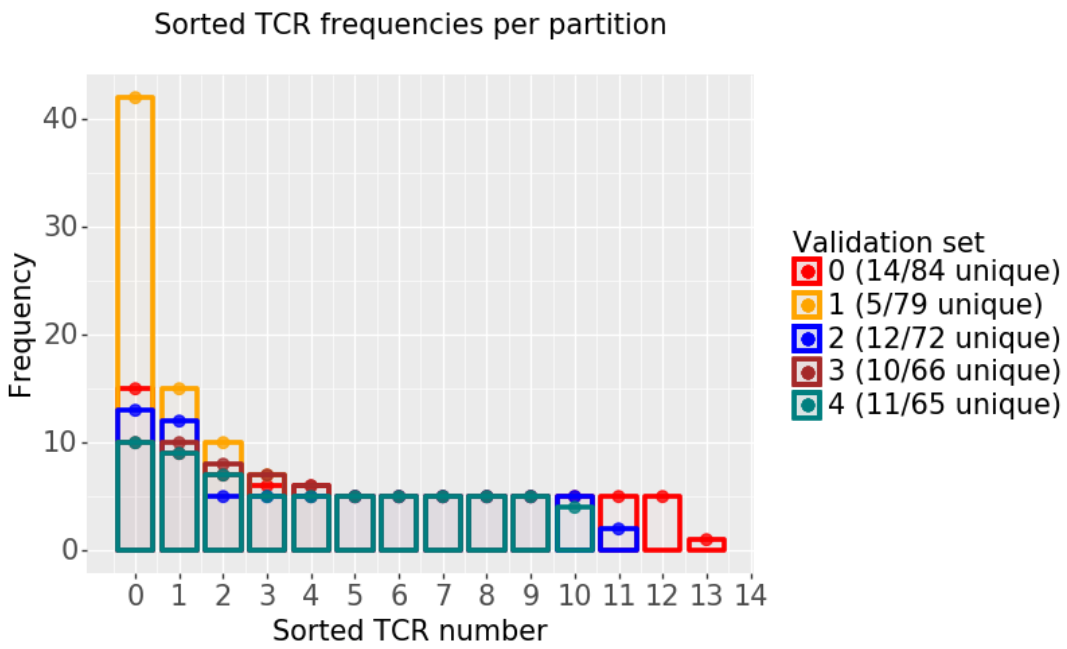


Figure 13: Graph showing the count per TCR ID in each partition. The sequences were ordered in descending order with the highest count (duplicates of the same sequence) first. The sequence IDs are not shared between partitions, although the actual sequences can be. The unique count out of the total set is shown in the legend.

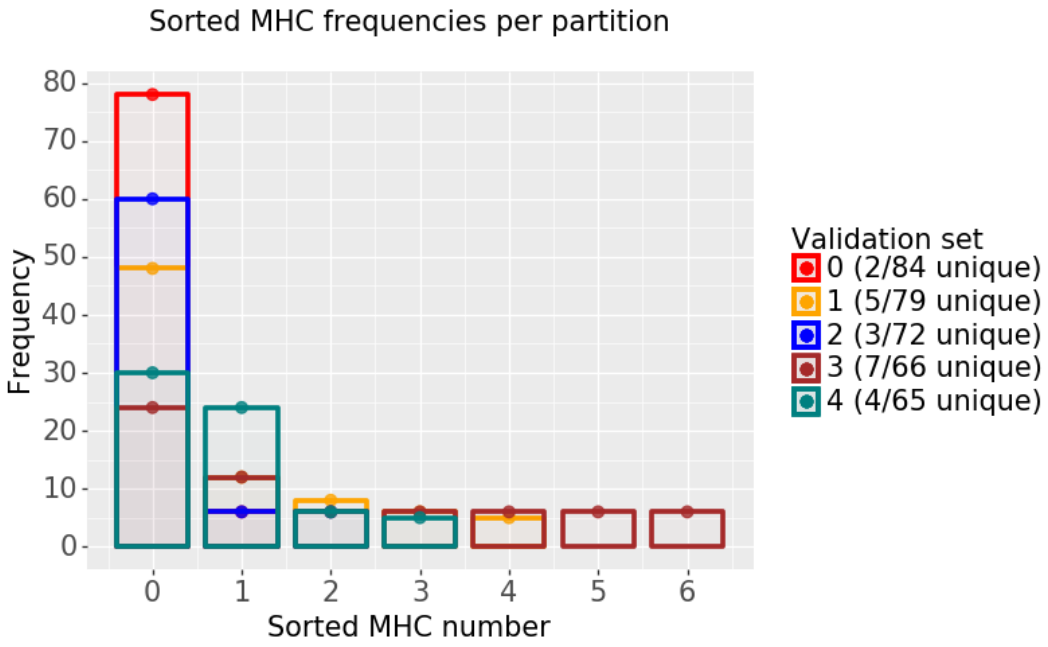


Figure 14: Graph showing the count per MHC ID in each partition. The sequences were ordered in descending order with the highest count (duplicates of the same sequence) first. The sequence IDs are not shared between partitions, although the actual sequences can be. The unique count out of the total set is shown in the legend.

It is important to note that the re-use of peptides across complexes and partitions is likely to allow for the model to overfit on the peptide sequence, by learning which peptides are often associated with negatives or positives. Therefore a heatmap was constructed to visualize the distribution of peptides across all partitions:

Figure 15: Heatmaps of the peptide, distributions across all partitions, with clustering of high % identity score sequences within each partition. Each sequence was compared with all other sequences for the group using pairwise alignment. The identity score was plotted as a heatmap. Clustering of high % identity scores was done per partition. The partitions are ordered from 0 - 4 along the x and y-axis, separated by white borders. The diagonal boxes represent sequences within a partition compared to itself, with the red squares within each box representing a cluster of the same or high identity percentage sequences. % identity score calculated from pairwise-alignment using Bioconda [43]

Figure 16: Heatmaps of the TCR and MHC distributions across all partitions, with clustering of high % identity score sequences within each partition. Each sequence was compared with all other sequences for the group using pairwise alignment. The identity score was plotted as a heatmap. Clustering of high % identity scores was done per partition. The partitions are ordered from 0 - 4 along the x and y-axis, separated by white borders. The diagonal boxes represent sequences within a partition compared to itself, with the red squares within each box representing a cluster of the same or high identity percentage sequences. % identity score calculated from pairwise-alignment using Bioconda [43]

As seen in the whole complex pairwise alignment heatmap (figure 17), there is some overlap of different partitions sharing the same or very high % identity peptides, but this represents a minority of the data. This is good, as failing to cluster the same peptides within a partition could allow the model to effectively "memorize" which peptides are associated with positive or negatives. This could result in a model which struggles to generalize its results to peptides it has never seen before.

It was seen during initial training of the models that the model performances were significantly higher when training on randomized partitions (not shown), where the partitions were constructed by randomly assigning complexes to each partition. This caused the partitions to have a far higher similarity between them. A heatmap of the randomized partitions is shown in the appendix.

We observe much more overlap in the T-cell receptor sequences and especially the MHC sequences. While there are not many unique MHC sequences in the dataset, the specific T-cell receptor binding to the peptide-MHC also plays a major role in determining positive or negative binding for the complex. A bias here could also allow the model to easily overfit on the models. In order to observe the total distribution of each complex, a heatmap was generated by averaging the % identity scores for MHC, peptide and TCR sequences, giving $\frac{1}{3}$ weight to each.

Figure 17: Heatmap of the TCR-p-MHC complexes pairwise alignment identity similarities. The sequences for each of the complex parts, the TCR, peptide and MHC, were extracted and pairwise alignment calculated. These were then averaged giving $\frac{1}{3}$ weight to each to give the final complex identity scores.

Despite significant overlap in the use of the limited number of MHC sequences, and partial overlap in the TCR and peptide sequences, the overall complex heatmap shows that the vast majority of complexes share less than 80 % combined identity.

3.3 Random forest analysis on BLOSUM-data

Understanding the inner mechanism or reason for predictions that come out of CNNs is challenging. One of the challenges with understanding the results from the original CNN model by Olsen is that beyond the performance scores there is little understanding for which features are important for the model predictions, how those features affect the predictions and whether there is any inherent bias in the features.

In order to answer these questions, it was decided to perform a random forest analysis. It was decided to split the original training data into four groups to study their effects on the model individually and in combination (see methods). All training was done using the original partitions and all or subsets of the original training data.

Table 3 shows that the isolated BLOSUM-scores when isolated by themselves strongly outperform the original CNN performance with an AUC 90%, MCC 0.591 vs AUC 75%, MCC 0.167. It also shows that when training on the peptide dataset, such that only the peptide region BLOSUM scores are available, the model still performs with an AUC of over 85 % and MCC of 0.37.

This indicates a large BLOSUM bias in the peptide region for the positive binding and negative non-binding groups. The global energy terms perform surprisingly well with an average MCC of 0.436.

A feature importance analysis of the top 20 features was generated. It can be seen in figure 18 that the peptide positions 183, 186 and 185 dominate the feature importance, corresponding to peptide positions 3, 6 and 5. With the exception of the energy terms P_total energy and PA_electrostatic kon and TCR position 300 (TCR alpha position 108), the top 8 positions were all part of the peptide.

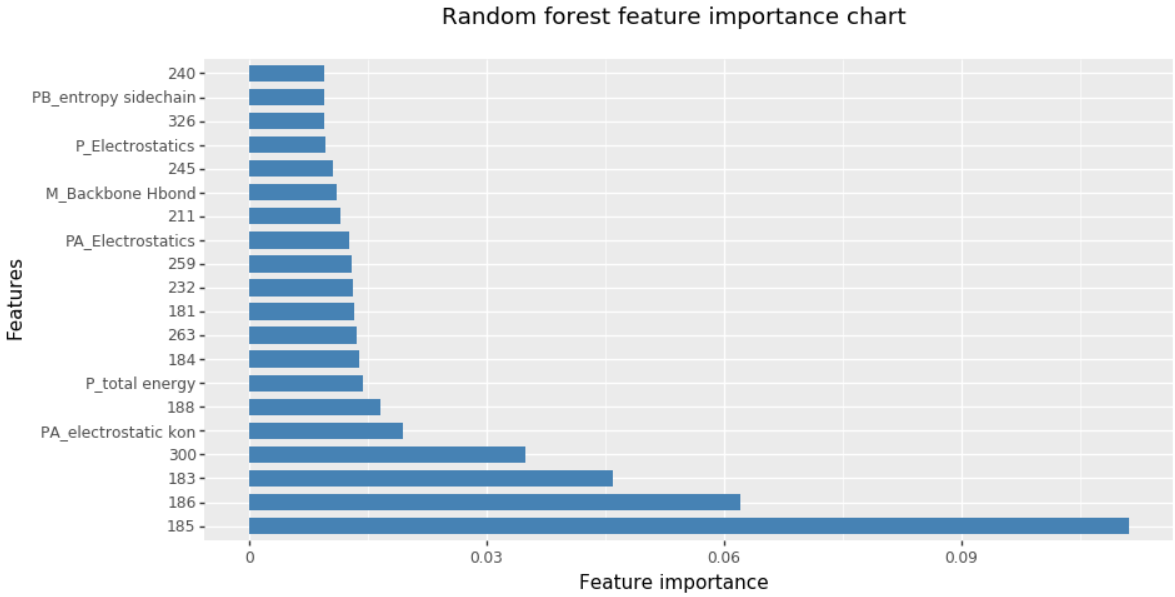


Figure 18: Feature importance chart of trained random forest. Feature importance is shown along the x-axis and the BLOSUM sequence positions or global energy terms are shown along the y-axis.

Observing the very strong performance and model dependence of the peptide region BLOSUM scores, it was decided to visualize the distribution of BLOSUM-scores between the peptides in the positive binding group and the negative non-binding groups (N, R and S).

Table 3: Random forest performance scores of the original dataset features, averaged per partition.

Feature	Accuracy	AUC	MCC	F1-score
Peptide only BLOSUM62	0.844	0.854	0.371	0.424
Full complex BLOSUM62 scores	0.897	0.899	0.591	0.616
BLOSUM-62 with Energy terms	0.872	0.804	0.436	0.438
Energy terms	0.827	0.610	0.020	0.03

Table 4: Mann-Whitney U-test p-values for the BLOSUM-score peptide distribution as seen in figure 19.

Comparison	p-value
Positives vs Negatives	2.21e-230
Positives vs High Rank% Negatives	4.42e-226
Positives vs Low Rank% Negatives	3.90e-125
Positives vs Switched negatives	0.33e-1

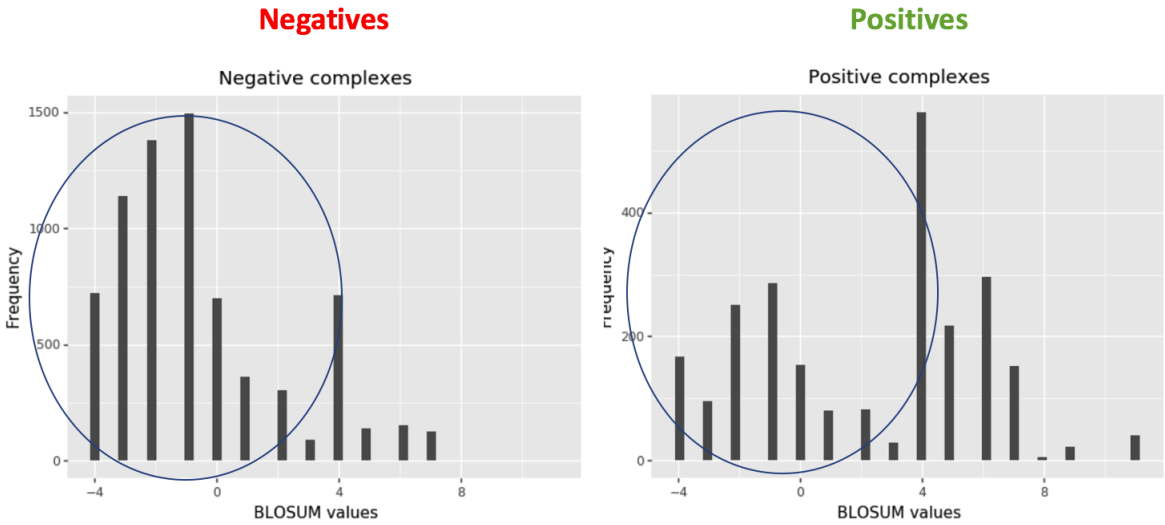


Figure 19: BLOSUM-score distribution for peptide positions 181-192 of positive and negative complexes excluding the Switched group. A significant bias towards negative BLOSUM-scores between the negative and positive group is seen, and encircled in both graphs.

A clear bias was seen favoring a higher frequency of negative BLOSUM-scores for the Negative (High and Low Rank %) group, and positive scores for the Positive group. The Switched group showed an identical distribution to the positive (not shown). This is expected as the Switched group only switches the TCR sequence, while the peptide remains the same. A Mann-Whitney U-test of the groups confirms the significant bias in the negative group BLOSUM scores 4.

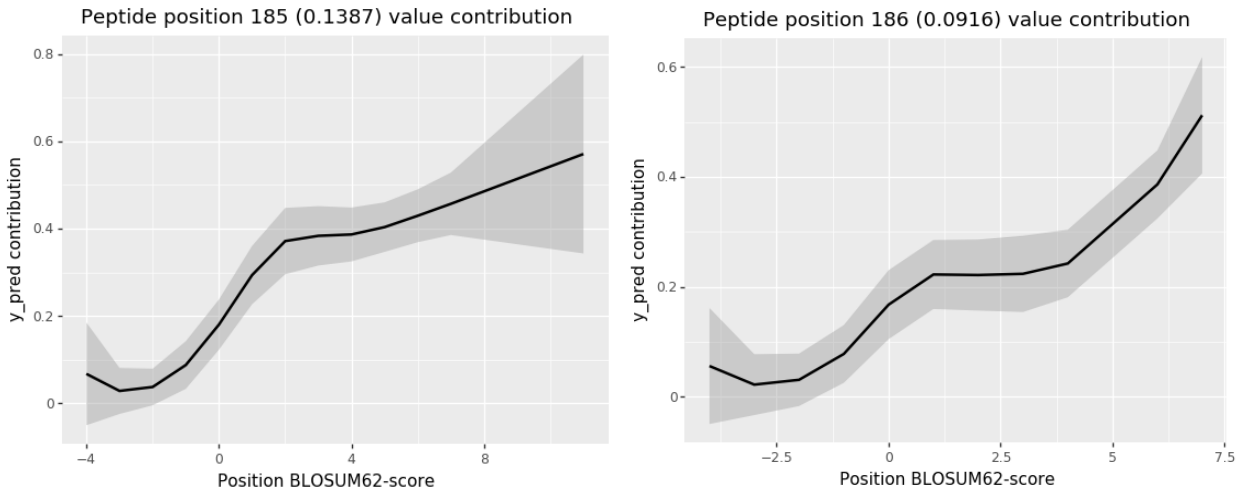


Figure 20: Relationship between peptide positions 185 and 186 BLOSUM-scores and their contribution to the random forest prediction score (y_pred). Made by extracting 500 random samples from the training set and plotting the average y_true values (0 for negatives, 1 for positives) for increasing BLOSUM-scores (see methods).

The BLOSUM-scores of the top 2 features, peptide position 185 and 186, and their indirect contribution to the prediction scores, can be visualized as a dependence graph (see methods).

Figure 19 shows how BLOSUM-scores for only these two positions strongly affect the final prediction. It can be observed that a BLOSUM-score at roughly 4 points is associated with a rise in the average true (y_true) label of 0.4 and 0.2 points respectively. A predicted value of over 0.5 points would be interpreted as a positive.

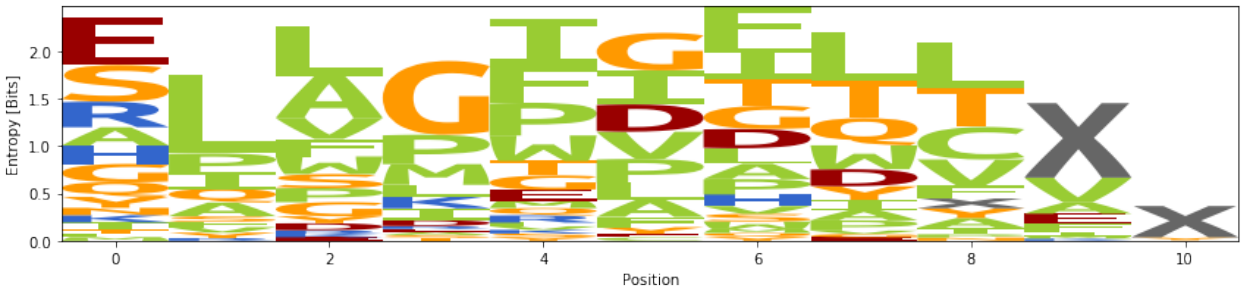
With these findings in mind, it was decided that the BLOSUM-scores displayed too much bias to be a meaningful replacement for the full amino-sequences. There is a clear difference in the BLOSUM scores for the positive and negative groups, and it is likely that the machine learning models exploit this bias from the dataset generation to drive predictions, as opposed to learning other patterns in the data.

Due to the strong bias seen in the BLOSUM-score data, it was decided to remove the per-position BLOSUM scores and replace them with the original amino-acid sequences in the complexes.

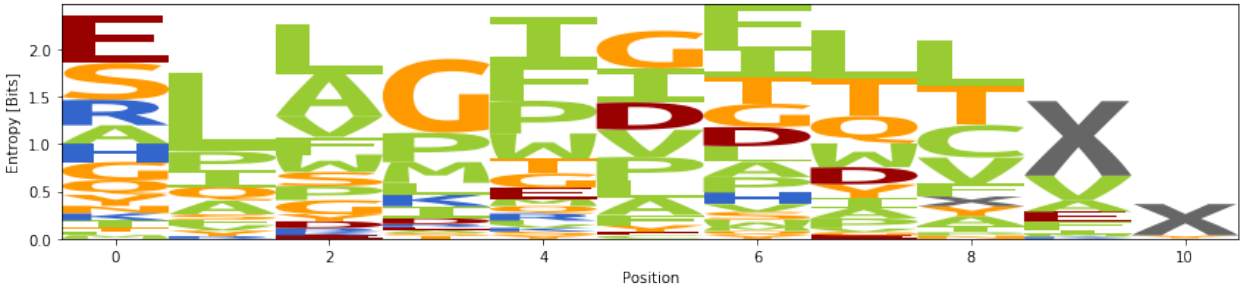
3.4 LOGO plots of peptide sequences

Due to the previously observed bias in the BLOSUM-scores, it was important to observe whether there was any clear bias in the peptides´ amino acid motifs. Logo plots were constructed from the positive, switched and negative group complexes to visualize the distribution.

LOGO-plot of positive complexes across all partitions:



LOGO-plot of switched complexes:



LOGO-plot of negative complexes:

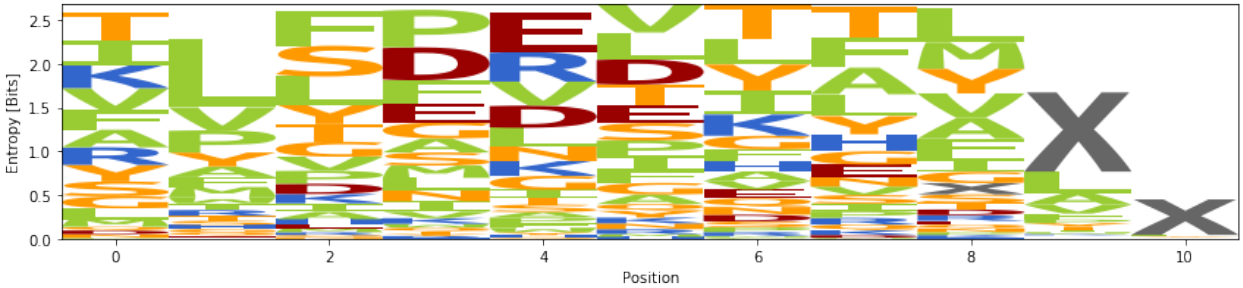


Figure 21: Overview of peptide sequence (positions 181-192) LOGO-plot motifs across all partitions. Entropy in bits is shown along the y-axis and the aligned peptide position with budding at positions 9 and 10 is shown along the x-axis

Table 5: Feed forward network performance scores of the original dataset features, averaged per partition.

Feature	Accuracy	AUC	MCC	F1-score
BLOSUM scores	0.743	0.764	0.386	0.475
Amino-sequences	0.750	0.620	0.112	0.252

Figure 21 shows that the LOGO plots motifs are seen to have significant overlap between them. The positive and switched groups share the same peptide motifs, as expected. The negative groups have a larger mix of residues which are not seen to the same extent in the positive group. However the distributions between the groups seems to largely overlap and no clear signal is seen for separating the positive and negative peptides.

3.5 Setting baseline with simple feed-forward network

Given the already mentioned bias in the BLOSUM scores, it is important to establish that training on the amino-sequences does not retain the same bias. It is also important to establish a baseline from a simple model on the new amino sequence data, from which the performance of a more complex CNN can be compared against.

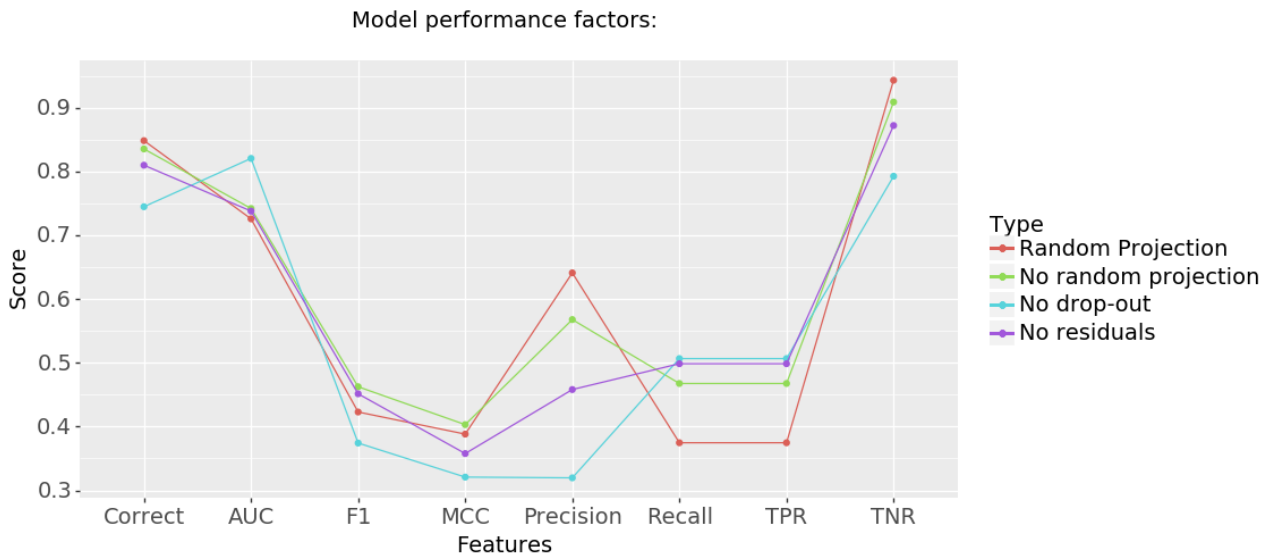
In order to achieve this, two simple feed forward network consisting of a fully connected layer connected to all input features along the entire sequence, one for the amino-sequence dataset and one for the original BLOSUM scores (see methods). Each feature dataset was trained for 20 epochs using early stopping and the 1-cycle policy, and the performances per validation set averaged.

The BLOSUM-scores were as expected seen to perform moderately well, with an AUC of 76 % and MCC of 0.39. Training on the amino-sequences did not present any strong performance scores, with a very poor AUC of 62 % and MCC of 0.11. This suggested that the amino-acid sequences at the very least presented significantly less bias the BLOSUM scores.

3.6 New CNN architecture

As the aim of the project included the design of a new and improved deep learning models, a new CNN architecture needed to be designed. A new architecture based on a published state of the art protein-protein interaction deep learning model was used. The architecture splits the protein complexes into two parts sent into independent convolutional pipelines for the TCR and MHC-peptide (see methods).

A series of test runs were done on the data to determine optimal architecture and hyperparameter settings. It was determined that the random projection unit (see methods) decreased the model s recall with a slight decrease in MCC-score. Testing of the use of drop-out and residuals confirmed that these improved the performance significantly. Testing of the specific combination of convolutional modules and residuals was not done due to time-constraints, and further testing would likely lead to a further optimized model.



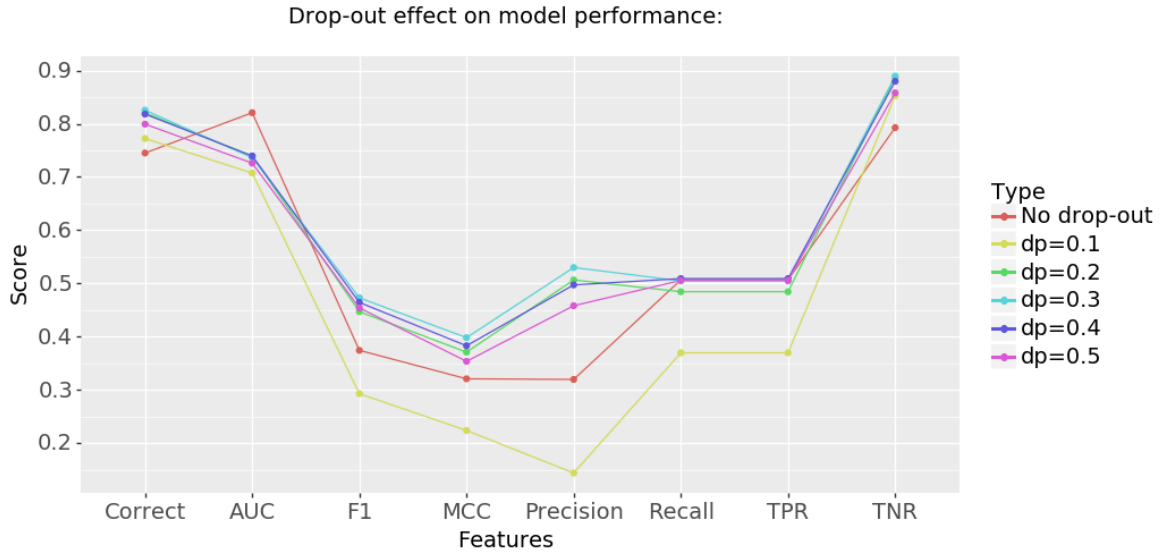


Figure 22: Selection of network architecture and hyperparameter decisions and effect on model performance. All runs without random projection except for the RP data points. Scores were averaged across 5 partitions, with 20 runs per partition.

The aim of the thesis was to demonstrate that extracting and training on additional, structural features in the protein structure can increase the model s performance in predicting positive models. In order to determine this, a baseline performance training only on the amino acid sequence was measured.

3.7 CNN amino-sequence performance

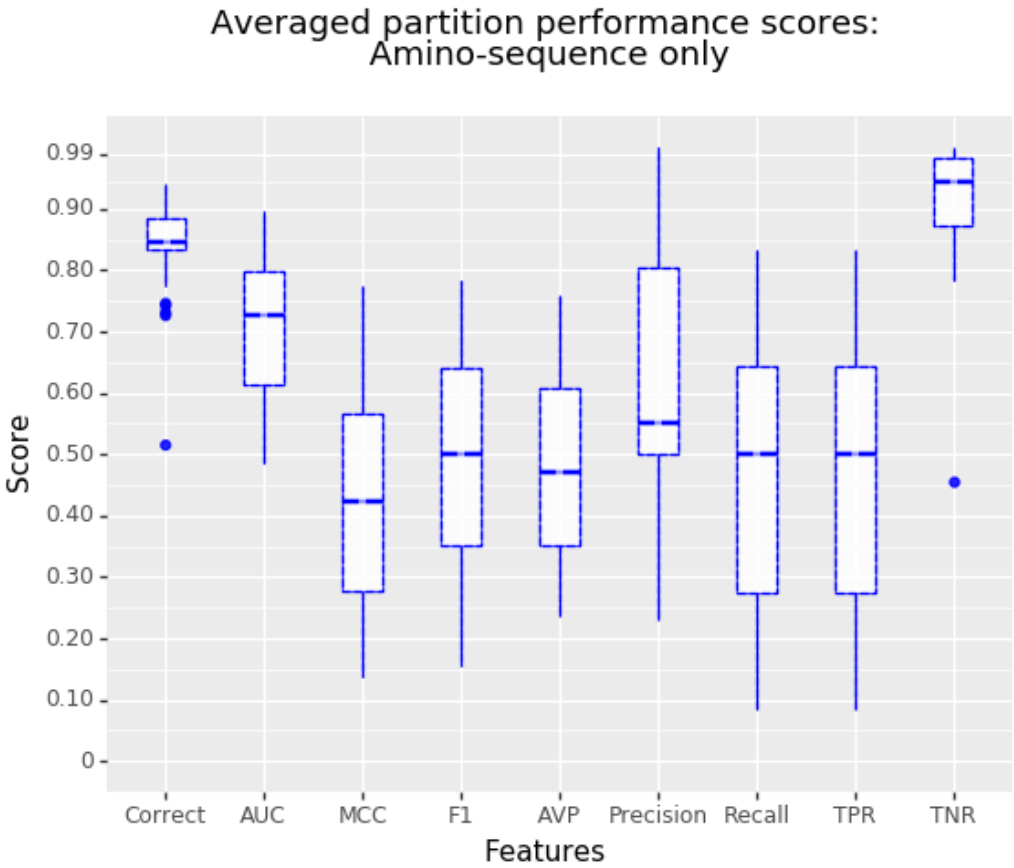


Figure 23: Averaged validation set performance of the Amino-acid sequence only. Scores were collected from 20 runs per validation set and averaged (100 runs total). Boxes show the 25-75th quartile and median values. Scores include AUC at default 50% cut-off, Matthews Correlation Coefficient, F1-score, Average Precision, True Positive and True Negative rate.

In the amino-acid training data, each complex was encoded as a 1-hot encoded sparse matrix, with each position having 20 feature channels representing the amino-acid residues. A series of 20 runs per validation set training only on the amino-acid sequence data was done.

An average accuracy of 85 % was noted (see Figure 23). The True Negative Rate was generally quite high at median 95 %, while the True Positive Rate was much lower, varying from 10-80 %. This was combined with a very large variation in the Matthew Correlation Coefficient (MCC) and F1-scores, at 0.15 - 0.76 and 0.15 to 0.77 respectively. As a large deviation in the average scores was observed, the individual partition performances contributing to that average were visualized.

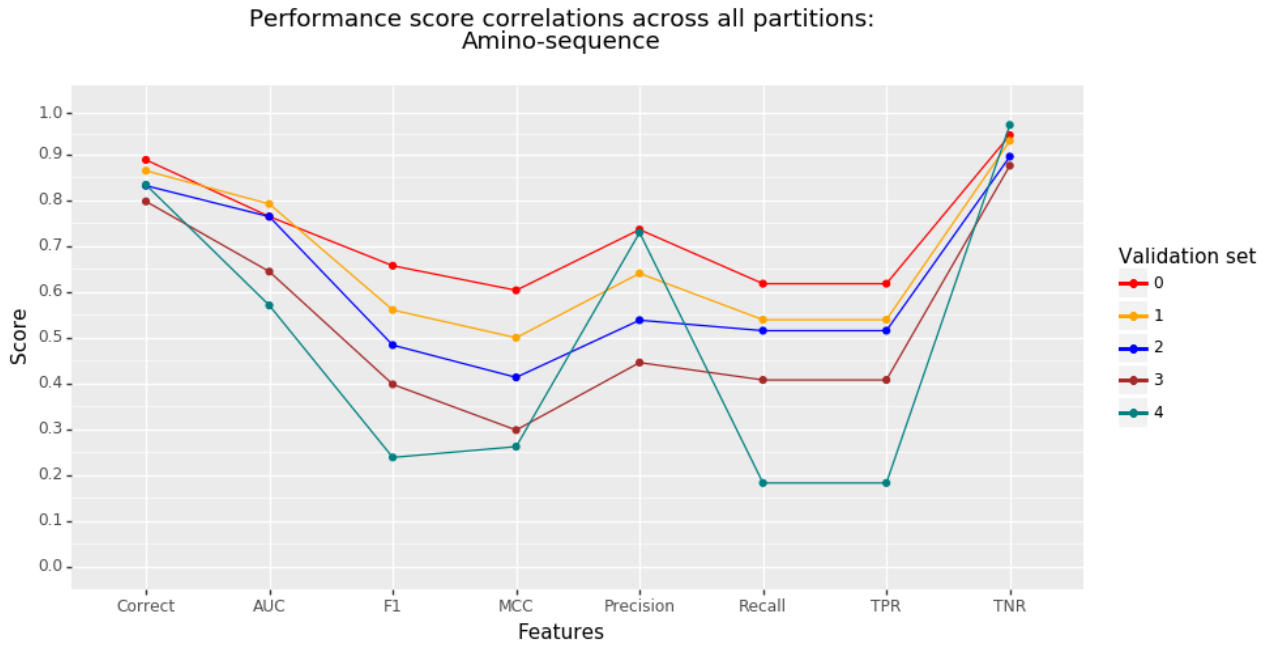


Figure 24: Performance of the Amino-acid sequence dataset, averaged per partition validation-set performance. Scores were collected from 20 runs per validation set (100 runs total).

Figure 24 shows differences in the individual partition performances, averaged across 20 runs per partition. It can be observed that there are large differences between the top and bottom scoring partitions, and a general correlation of most performance scores. It is noted that:

- The True Negative Rate is very high for partitions, with a mean of 87 - 96 % for all partitions.
- The True Positive Rate is generally much lower and varies much more, down to an average of 18 % for one partition and up to 62 % for the highest partition.
- For two of the partitions, 4 and 3, the general performance is very poor (MCC 0.26 and 0.30, AUC 57 - 64 %)

These findings seem to suggest that while the amino-acid feature dataset can produce good performance scores in AUC and MCC, it really struggles to generalize for some partitions and generally performs very well on the true negatives and more poorly on the positive complexes.

In order to try to raise the predictive power of the model and especially increase the recall of positive complexes (true positive rate), it was decided to integrate additional structural and energy term features.

3.8 CNN combined features performance

Each TCR-p-MHC complex has 4 structural models, modelled on templates with up to 99, 95, 90 and 80 % template identity respectively. The complexes are stored in .PDB files, meaning any structural features that can be extracted from the protein structure format with tools such as Bioconda can be used for the training data.

A PDB file processor named PDB_Tool was used to extract several per-position structural, along with some custom-generated features such as the protein type and sequence position (see methods). In addition to this, 66 ranked FoldX interaction energy terms, detailing the global interaction energies of the protein complexes, were made available from the previous work by Olsen extracted from the structural models using FoldX [45] (see methods). These were already ranked according to their feature importance as determined by random forest analysis, from highest impact on predictions to lowest (see appendix, table 8.3).

The full dataset was preprocessed and normalized, and split into the original 5 partitions. In order to gain a better understanding of how the amino-acids, structural and energy features each and in part contributed to the model performance, it was decided to split the dataset into 8 subsets (see methods):

1. Amino-sequence excluding Peptide
2. Structure terms only
3. Top 5 energy terms
4. Top 10 rank energy terms
5. Structure terms top 10 energy terms
6. Amino-acid sequence
7. All features combined (Amino-sequence, structural terms top 24 energy terms)

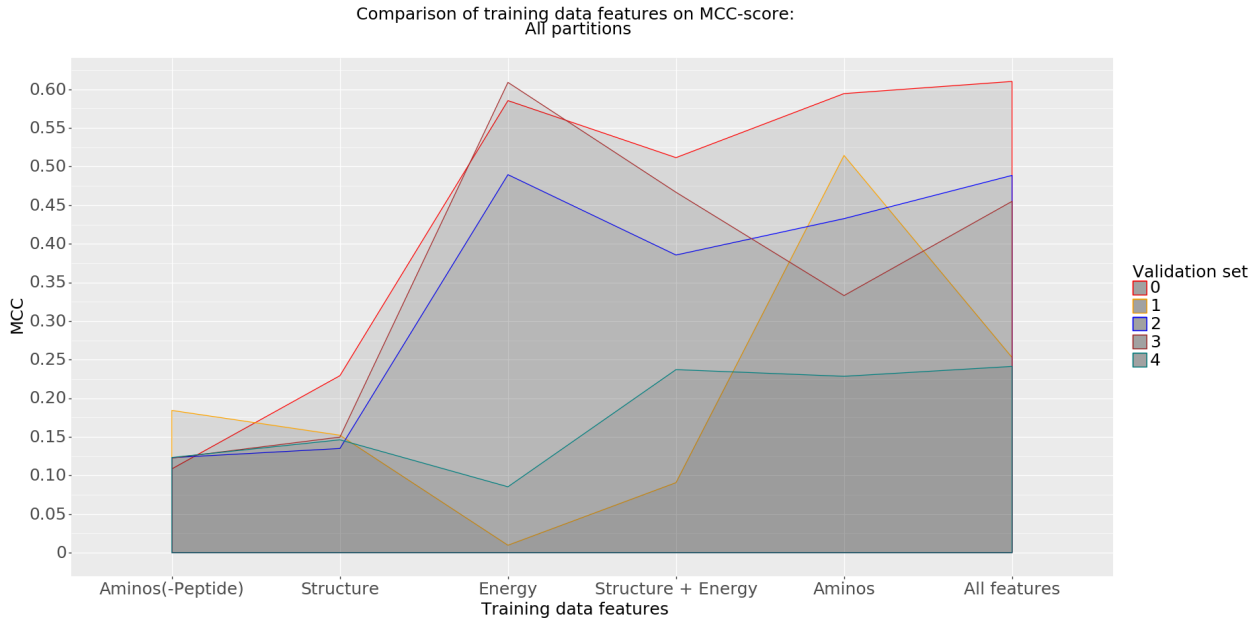


Figure 25: Comparison of all training features on model MCC-score performance. Each feature set was trained independently 20 times per partition (800 total), and the average MCC was calculated.

Figure 25 shows the results of training the 8 different subsets a total of 20 times per partition and averaging the partition MCC-scores. The amino-acids were observed to reach top performance only a single partition (validation set 1). Combining all features is seen to increase the MCC-score in 4 out of 5 partitions by 0.05 to 0.15 points, while strongly decreasing it in one partition by an average of 0.20 points.

The structural dataset performs about equally well in each partition at a poor 0.15 MCC. The energy terms by themselves are observed to have a surprisingly high performance, reaching top or near top performance in 3 partitions (MCC 0.48 to 0.62), but having poor performance in the remaining two partitions 4 and 1 (MCC MCC 0.09 and 0.01).

As the aim of integrating the new structural and energy features was to see an increase not only in MCC but particularly of the recall of positive complexes or true positive rate, the individual partition performances were averaged across all partitions and all performance scores visualized for the datasets.

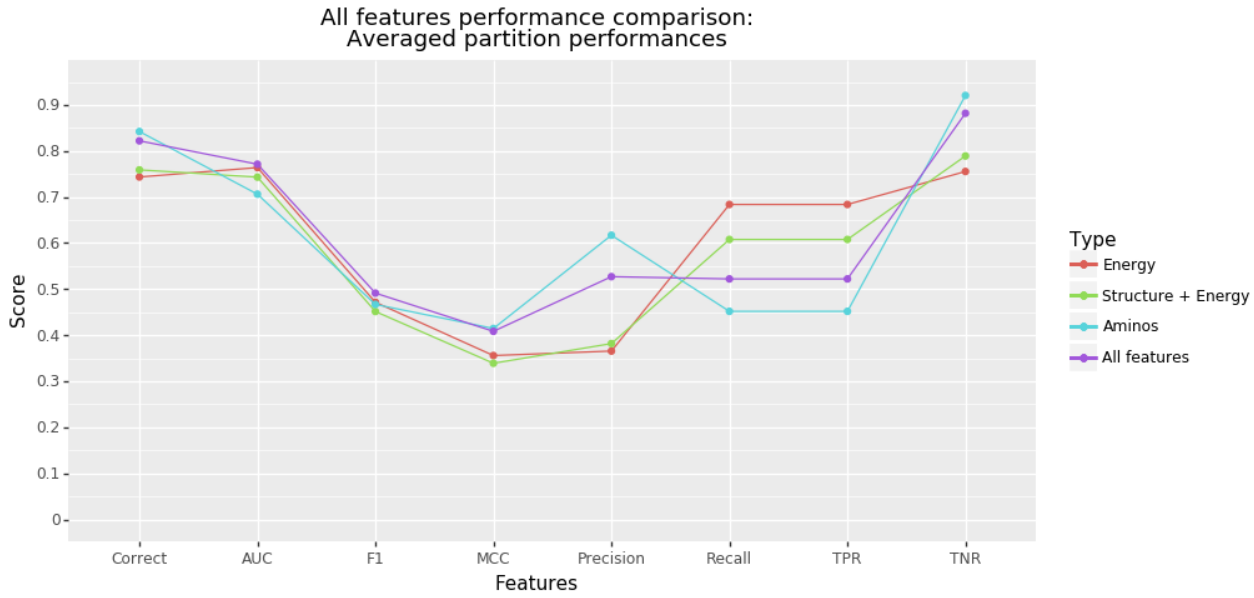


Figure 26: Comparison of all performance scores across feature datasets. Partition performances scores were averaged across 20 runs per partition per feature dataset (800 total).

Figure 26 shows all averaged performance scores for the same feature datasets. With the averaged scores, it is seen that the MCC is almost exactly the same for all features and the amino-sequence datasets (0.42 vs 0.43), with a slightly higher F1-score for all features (0.49 vs 0.46). The amino-sequence dataset is seen to have the lowest True Positive Rate and the highest True Negative Rate. All features has a slightly higher true positive rate (increase of 8 %) and slightly lower true negative rate (decrease of 3 %). The structure + energy terms and especially the energy terms by themselves are observed to have the highest true positive rates (69 and 61 %) while only having moderately lower MCC-scores (0.42 to 0.36 and 0.34 respectively).

While the model performance scores give an indication of the overall performance, they do not visualize or explain what is happening with the predictions of the individual complexes in the validation set. In order to gain a better understanding of how well the models capture positives, prediction graph visualizing the frequency of positive predictions per complex were made.

3.9 Visualizing model predictions

A series of graphs were made visualizing the counts of positive predictions across 20 individually trained models (see methods). In order to understand these graphs, first the frequency of predictions for validation set 2 is shown, and then the combined graphs for all validation sets (see appendix for prediction graphs for the other partitions).

Figure 27 shows the frequency of predictions for validation set 2 and indicate stability of those predictions.

A few observations can be made from these graphs:

- The amino-acid dataset generally favors negative predictions and has poor signal for most positive complexes (TPR 0.51, TNR 0.89, MCC 0.41).
- All features combined increases the recall of positive complexes significantly, both of those predictions caught by the amino-sequence dataset, and new ones (TPR 0.72, TNR 0.84, MCC 0.50).
- The energy data seems to be able to identify the highest ratio of positive complexes but also has a higher degree of false positives (TPR 0.82, TNR 0.78, MCC 0.49)
- The structural data produces far more noise and it is not clear which predictions it favors (TPR 0.47, TNR 0.69, MCC 0.13).

Overall most positive complexes are missed except for the structural data predictions. However combining all features together seems to boost the number of true positive complexes significantly.

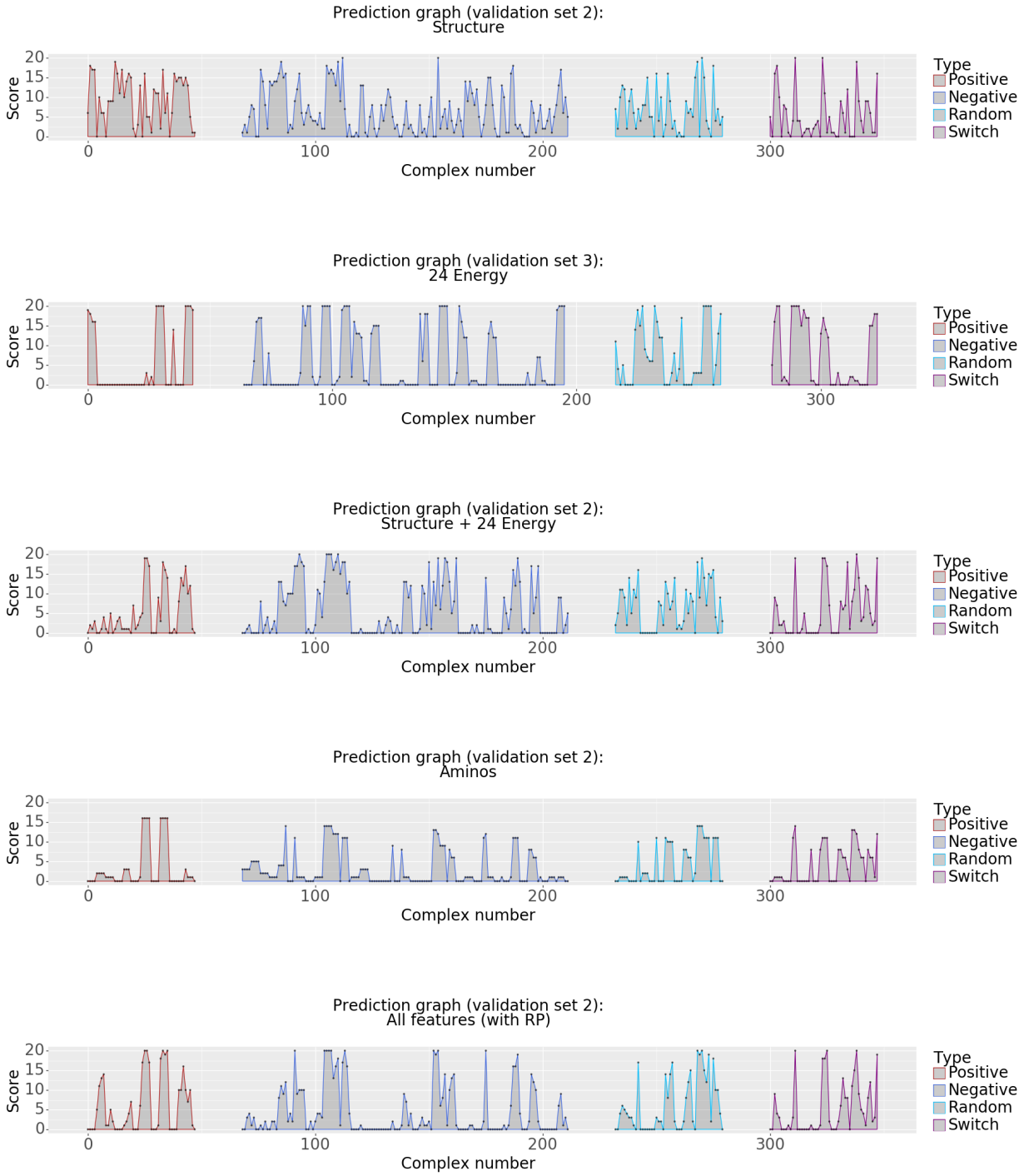


Figure 27: Graph of model predictions per feature dataset across validation set 2. Out of 20 trained models, every positive prediction gains a score of +1. Complexes with +20 score were always predicted as positive, and 0 never predicted as positive. Datasets: 1. Amino-sequence with peptide sequence excluded, 2. Structural features (36) only, 3. Energy terms (24) only, 4. Amino-acid sequence only 5. All features combined.

Next, the combined prediction graphs for all partitions were made, in order to observe whether similar observations can be generalized across the entire dataset. Similar findings can also be seen when looking at all partition performances combined:

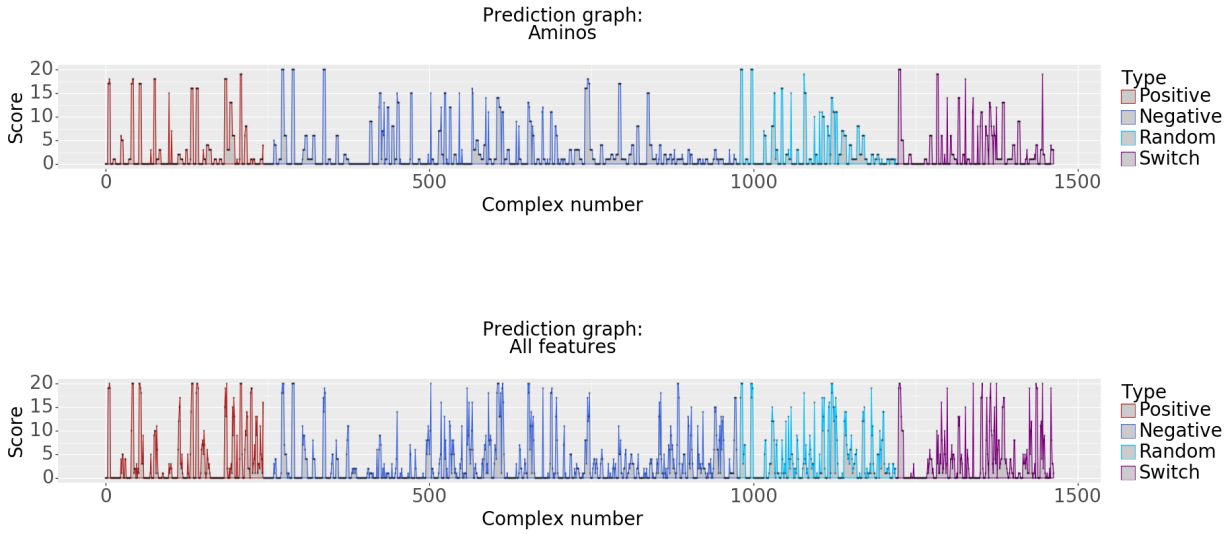


Figure 28: Global prediction scores for number of positive predictions over 20 trained models across all validation sets. All validation sets have been concatenated together to represent the entire dataset of all complexes, for the amino-sequence and all features combined datasets.

It seems that combining all features together significantly increase the number of positively predicted complexes and their count. In order to better visualize the difference between the graphs, an enrichment graph visualizing only their differences was made.

3.10 Prediction enrichment graphs

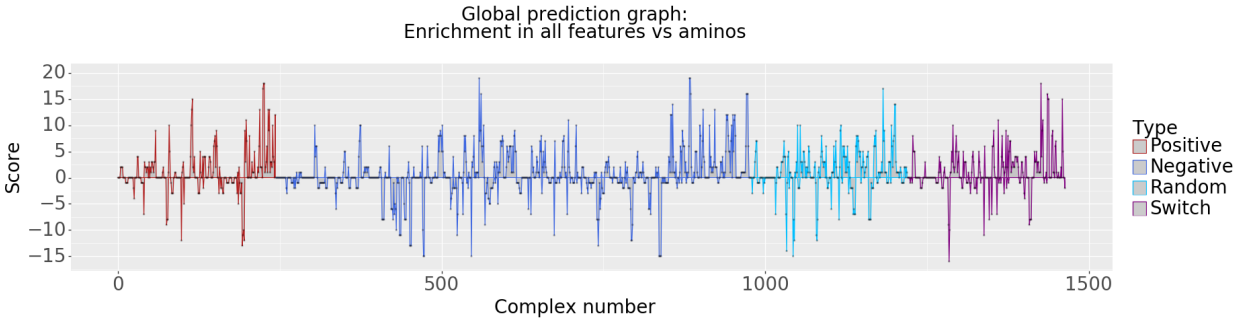


Figure 29: Global enrichment of positive prediction chance in all features vs amino-sequence features only, comprising all complexes. Positive scores indicate an increase in the number of predicted positives for the all-features data, while negative scores indicate a decrease.

It can be observed in figure 29 that all-features generally has a strong increase of many positive complexes, with a more mixed set of increases and decreases for the non-binding groups. Out of 244 positive complexes, all-features leads to an increase in true positives for 76 of them (31 %) by an average score of 5.40. Meanwhile there is a decrease in 28 complexes (11 %) by an average score of -5.29 ($p = 0.036$).

Considering the negative groups the increase in model performance is less clear: out of 732 negative complexes, all-features leads to an increase in false positives for 177 negative complexes (24 %), by an average of 5.86, and a decrease in false positives for 113 negative complexes (15 %) by an average score of -5.6 ($p = 0.0003$).

3.11 Effects of model quality

Until now there has been no consideration of how the model s predictions is likely to change based on the quality of the homology templates they were modelled on. Each model is allowed a template threshold of up to 99, 95, 90 or 80 %. In order to observe any potential impact of the model quality on the predictions made, the predictions of the grouped and averaged 95 and 99 % threshold models were compared with the grouped and averaged 80 and 90 % threshold models (see methods), visualizing the enrichment between them (figure 30).

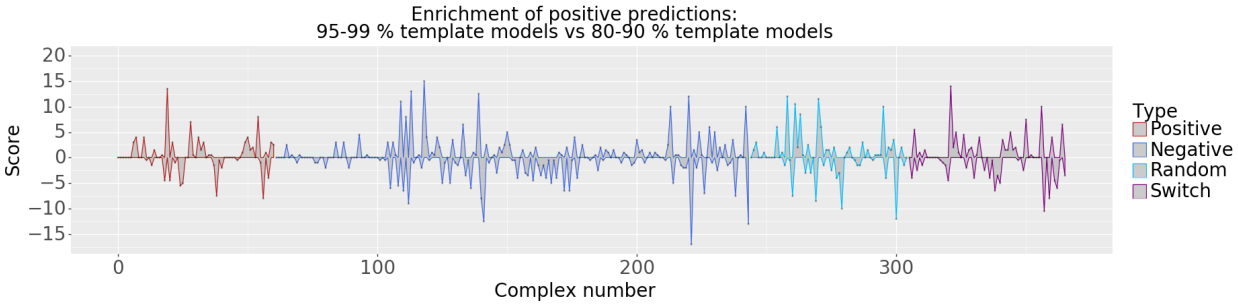


Figure 30: Global enrichment of positive predictions in the 95-99 % vs 80-90 % template threshold models, for the all features dataset. The two groups were averaged, and the 80-90 % scores subtracted from the 95-99 % scores.

While there seems to be a weak favoring of the higher quality models (95 and 99 %) in increase true positive scores, the difference is not statistically significant (p = 0.25). 17 complexes (28 %) have an increased score by an average of 4.0 points, while 10 complexes (16 %) have a decrease score by an average of -4.4 points. In the negative groups, there is an equal increase and decrease of ca 41 and 42 complexes by 4.64 and -4.73 points respectively (p = 0.28).

4 Discussion

The final results show that when comparing all features combined versus only the amino acid sequence, there is not sufficient evidence to prove a general increase in the MCC-scores. The sequence data by itself performs quite strongly with an MCC of 0.42, which is not surprising given the publication of models such as NetTCR, which has success in predicting MHC-p-TCR binding on the most common MHCI allele using TCR CDR3 sequences only [5].

A general pattern can be observed where poor energy term scores are seen to either significantly pull the All-features performance down or prevent them from increasing vs amino-acids only. When the energy terms perform well there is a significant increase in their performance vs amino-acids. The top performing validation sets for the energy dataset are also the top performing ones for the all-features dataset. These show a clear increase in performance over the amino-sequence set. However when the energy data is seen to be performing poorly, the all-features set performance either significantly decreases or is not able to improve. This is not surprising, given that previous results by Lanzarotti et al showed that protein energy terms extracted from FoldX and Rosetta by themselves had significant predictive power towards TCR-p-MHC binding [18]. However their model´s performance was reliant on the structural modelling quality, and accurate prediction of MHC-peptide binding.

The amino-sequence dataset had the lowest recall (% of positive cases caught) while having a high precision and very high true negative rate. The model was observed to mostly predict all complexes as negative with a strong signal for a limited number of positive complexes (figure 28). It is interesting to note that the combined energy and structural data only show weakly to moderately decreased AUC, F1 and MCC scores while still capturing far more positives, as seen in their top ranking recall scores (figure 26).

The prediction enrichment graphs showed that combining all features significantly increased the recall of complexes which already had positive prediction signals in the amino-sequence set, and lead to recall of more complexes which were otherwise missed. Both the structural data and energy data by themselves were seen to be both somewhat and very predictive, with the structural data having a stable and lower MCC of 0.15 - 0.23. The energy terms show a combination of both surprisingly high MCC at 0.48 to 0.62 and low scores of 0.09 and 0.01 depending on the validation dataset. As the FoldX energy term data is extracted independently of the amino-sequence and has no direct relationship with the PDB_tool residue features, this gives further evidence that potentially strong signals for protein-interaction prediction can be extracted from global protein features such as energy terms. It is likely that a larger training dataset is required for the structural and energy data to generalize better.

These results indicate that careful selection of predictive features plays a key role in improving performance beyond predicting on amino-sequences. Further efforts should be spent on establishing which structural and energy features extracted from protein structural models are predictive of binding by themselves, and are capable of improving predictor performance in combination with the amino-sequences.

Model quality

The graph of enrichment of positive predictions in the higher quality templates (figure 30) did not indicate that the all-features model had any clear favoring of higher quality models when it had been trained on models of varying quality of 80-99% thresholds. This is surprising, given previous results by Lanzarotti et al observing that the model quality was of high importance in providing good predictions. This could suggest that training on energy features in combination with amino-sequences and structural data could alleviate the limitations of predicting on the energies by themselves. Or more likely, the training dataset is too small, or there is too strong signal in the amino-sequence data which would not generalize well with new data. This should be investigated further with a new test set.

Dataset bias

It was seen in the original BLOSUM62 dataset that a relatively simple random forest model was able to predict extremely well based on the per position BLOSUM62 substitution scores, both across the entire complex and when the peptide was isolated by itself. This was confirmed in a large negative score bias for the negative peptides.

Generation of the negative dataset is a crucial step, as the machine learning model is only trained on distinguishing the difference between the two sets. If the method for creating the negatives has an inherent bias, the model can easily learn to pick up on the introduced bias rather than learning more complex but desirable patterns in the data. It is likely that the bias arises as an artefact of the homology modelling process, whereby higher identity templates are likely to be available for real, existing structures, and less readily available for artificial negative ones. In effect this would let the model indirectly know which complexes are likely to be real simply by their BLOSUM-scores, a pattern seen in the random forest dependence plots (figure 20). This describes a phenomenon known as "data leakage", a big problem in developing predictive models whereby data from outside the training set "leaks" into the dataset and can still be used by the model.

Replacing the BLOSUM scores with the sequence amino acids was seen to significantly reduce the bias in a simple feed forward network. While no clear motif signal was seen in the peptide logo plots, with a large overlap between the groups, it is likely crucial that the dataset includes a peptide switched set which punishes the model for memorizing peptide motifs in the positives.

5 Conclusion

Random forest analysis of the BLOSUM62 features showed a very strong positive bias towards the positive complexes, which was present even when predicting only in the peptide region. The scores were instead replaced with their original TCR-p-MHC amino-acid sequences. A simple, single-layer feed forward network comparing two datasets of the original BLOSUM-scores and the new amino-sequences showed a significant reduction in prediction performance.

Global FoldX energy terms and energy terms combined with structural features encoded along the sequence length was seen to have significant predictive power by themselves, with average MCC-scores of 0.36 and 0.34 respectively. The Amino-sequence features by themselves had an average MCC of 0.42. Including structural and energy features with the amino-sequence did not increase the average MCC, showing an average MCC of 0.41. However, increased MCC-score performance was seen in 4 out of 5 partitions by 0.05 to 0.15 MCC points, with a large decrease of 0.26 points in the last partition. Average AUC was increased from 71 % to 78 %.

The average true positive rate was seen to rise significantly by 15 % points (45 to 60 %), with a 5 % point decrease in the true negative rate (93 to 88%). The global enrichment in prediction scores showed that 31 % of the positive complexes saw an increase in recall by an average of 27 % points, while 11 % of the positive complexes saw a decrease by an average of 25 % points, favoring an increase in prediction power not seen in the amino-sequence data.

The all-features model showed no statistically significant relationship between the homology template thresholds for the structural models and their increase or decrease in prediction scores. No clear relationship favoring prediction of the high or low rank% peptide or TCR-switched groups was seen.

6 Future perspectives

The project results have provided exciting findings that combination of protein structural features and global energy terms can provide significant power for prediction of TCR-p-MHC binding, both by themselves and especially in combination with the complex sequences. Given the challenges of predicting on sequences only, and the increasing availability of homology modelling templates as more proteins have their structures elucidated, integration of features hidden in the protein structure could provide a significant increase in predictive model performances.

A systematic search elucidating which protein structural features or global energy terms most reliably increase performance, by themselves and in combination with the complex sequences, should be performed. Additionally these findings should be confirmed on independent test sets. It should also be investigated further whether integration of these features with the sequences can weaken predictive performance if the model qualities are poor. A strategy of integrating the model qualities into the predictor using model quality indicators such as protein-docking DockQ [46] scores or local residue quality assessments using ProQ3D [47] should be investigated. However such an approach must take care to not introduce data leakage into the model, such as was introduced using BLOSUM substitution scores in this project. Unintentionally introducing biases of which models are likely to be real world structures and therefore positives, and which are artificially created, would risk artificially elevating performance scores and lead to a model with poor generalization performance when predicting new data.

7 References

[1] Charles A Janeway Jr et al. "The major histocompatibility complex and its functions". In: *Immunobiology: The Immune System in Health and Disease. 5th edition*. Garland Science, 2001.

[2] Glenn Dranoff. "Cytokines in cancer pathogenesis and cancer therapy". en. In: *Nat. Rev. Cancer* 4.1 (Jan. 2004), p. 11.

[3] Et al Arstila TP. *A direct estimate of the human alphabeta T cell receptor diversity. - PubMed - NCBI*. https://www.ncbi.nlm.nih.gov/pubmed/10542151. Accessed: 2019-5-28.

[4] Andrew J T George, Jaroslav Stark, and Cliburn Chan. *Understanding specificity and sensitivity of T-cell recognition*. 2005.

[5] Vanessa Isabell Jurtz et al. *NetTCR: sequence-based prediction of TCR binding to peptide-MHC complexes using convolutional neural networks*.

[6] N Pozzesi et al. "Role of caspase-8 in thymus function". en. In: *Cell Death Differ.* 21.2 (Feb. 2014), p. 226.

[7] Stephanie Gras et al. "Allelic polymorphism in the T cell receptor and its impact on immune responses". en. In: *J. Exp. Med.* 207.7 (July 2010), p. 1555.

[8] Et al Bentzen AK. *Large-scale detection of antigen-specific T cells using peptide-MHC-I multimers labeled with DNA barcodes. - PubMed - NCBI*. https://www.ncbi.nlm.nih.gov/pubmed/27571370. Accessed: 2019-5-28.

[9] Protein Data Bank. *PDB 5men structure summary ‹ Protein Data Bank in Europe (PDBe) ‹ EMBL-EBI*. https://www.ebi.ac.uk/pdbe/entry/pdb/5MEN. Accessed: 2019-5-28. Dec. 2016.

[10] Koichi S Kobayashi and Peter J van den Elsen. "NLRC5: a key regulator of MHC class I-dependent immune responses". en. In: *Nat. Rev. Immunol.* 12.12 (Nov. 2012), p. 813.

[11] Andrew J Bordner. "Structure-Based Prediction of Major Histocompatibility Complex (MHC) Epitopes". en. In: *Immunoproteomics*. Humana Press, Totowa, NJ, 2013, pp. 323–343.

[12] *NetMHCpan 4.0 Server*. http://www.cbs.dtu.dk/services/NetMHCpan/. Accessed: 2019-5-30.

[13] Et al Burrows SR. *Have we cut ourselves too short in mapping CTL epitopes? - PubMed - NCBI*. https://www.ncbi.nlm.nih.gov/pubmed/16297661. Accessed: 2019-5-28.

[14] Mikhail Shugay et al. "VDJdb: a curated database of T-cell receptor sequences with known antigen specificity". en. In: *Nucleic Acids Res.* 46.Database issue (Jan. 2018), p. D419.

[15] *VDJdb*. https://vdjdb.cdr3.net/. Accessed: 2019-5-28.

[16] Et al Vita R. *The immune epitope database (IEDB) 3.0. - PubMed - NCBI*. https://www.ncbi.nlm.nih.gov/pubmed/25300482. Accessed: 2019-5-30.

[17] *IEDB.org: Free epitope database and prediction resource*. http://www.iedb.org. Accessed: 2019-5-29.

[18] Nielsen M3 Lanzarotti E1 Marcatili P2. "Identification of the cognate peptide-MHC target of T cell receptors using molecular modeling and force field scoring". In: *Mol. Immunol.* 94 (Feb. 2018), pp. 91–97.

[19] Yifan Song et al. "High-resolution comparative modeling with RosettaCM". eng. In: *Structure (London, England : 1993)* 21.10 (Oct. 2013), pp. 1735–1742. ISSN: 1878-4186. DOI: 10.1016/j.str.2013.08.005. URL: https://www.ncbi.nlm.nih.gov/pubmed/24035711%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3811137/.

[20] Joost Schymkowitz et al. "The FoldX web server: an online force field". en. In: *Nucleic Acids Res.* 33.Web Server issue (July 2005), W382.

[21] Sean R Eddy. "Where did the BLOSUM62 alignment score matrix come from?" en. In: *Nat. Biotechnol.* 22.8 (Aug. 2004), pp. 1035–1036.

[22] Prince Canuma. *Understanding Neural Networks*. https://towardsdatascience.com/understanding-neural-networks-22b29755abd9. Accessed: 2019-5-31. Feb. 2019.

[23] *NetMHCpan 4.0 Server*. http://www.cbs.dtu.dk/services/NetMHCpan/. Accessed: 2019-5-30.

[24] Pedro Ney Stroski. *What are convolutional neural networks? - Electrical e-Library.com*. http://www.electricalelibrary.com/en/2018/11/20/what-are-convolutional-neural-networks/. Accessed: 2019-5-30. Nov. 2018.

[25] Et al Min S. *Deep learning in bioinformatics. - PubMed - NCBI*. https://www.ncbi.nlm.nih.gov/pubmed/27473064. Accessed: 2019-5-30.

[26] Niklas Donges. *The Random Forest Algorithm*. https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd. Accessed: 2019-5-30. Feb. 2018.

[27] Raheel Shaikh. *Cross Validation Explained: Evaluating estimator performance*. https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85. Accessed: 2019-5-30. Nov. 2018.

[28] Olsen TH. *Combining deep learning and structural modeling to predict T cell receptor specificity - DTU Findit*. https://findit.dtu.dk/en/catalog/2438676047. Accessed: 2019-5-30.

[29] *API Reference — scikit-learn 0.21.2 documentation*. https://scikit-learn.org/stable/modules/classes.html#sklearn-metrics-metrics. Accessed: 2019-5-29.

[30] fastai. *fastai/fastai*. https://github.com/fastai/fastai. Accessed: 2019-5-28.

[31] Sheng Wang. *realbigws/PDB_Tool*. https://github.com/realbigws/PDB_Tool. Accessed: 2019-5-27.

[32] *basic_train | fastai*. https://docs.fast.ai/basic_train.html#Discriminative-layer-training. Accessed: 2019-5-27.

[33] Leslie N Smith and Nicholay Topin. "Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates". In: (Aug. 2017). eprint: 1708.07120.

[34] *torch.nn — PyTorch master documentation*. https://pytorch.org/docs/stable/nn.html#crossentropyloss. Accessed: 2019-5-29.

[35] Somaye Hashemifar et al. *Predicting protein–protein interactions through sequence-based deep learning*. 2018.

[36] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2016.

[37] Sabyasachi Sahoo. *Residual blocks — Building blocks of ResNet*. https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec. Accessed: 2019-5-22. Nov. 2018.

[38] *NumPy — NumPy*. https://www.numpy.org/#. Accessed: 2019-5-29.

[39] *Python Data Analysis Library — pandas: Python Data Analysis Library*. https://pandas.pydata.org/. Accessed: 2019-5-29.

[40] *JupyterLab Documentation — JupyterLab 0.35.6 documentation*. https://jupyterlab.readthedocs.io/en/stable/. Accessed: 2019-5-29.

[41] *scipy.stats.mannwhitneyu — SciPy v1.3.0 Reference Guide*. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.mannwhitneyu.html. Accessed: 2019-5-29.

[42] *A Grammar of Graphics for Python — plotnine 0.5.1 documentation*. https://plotnine.readthedocs.io/en/stable/. Accessed: 2019-5-29.

[43] *Bio.pairwise2*. https://biopython.org/DIST/docs/api/Bio.pairwise2-module.html. Accessed: 2019-5-29.

[44] micked. *micked/mpl-logoplot*. https://github.com/micked/mpl-logoplot. Accessed: 2019-5-29.

[45] *Homepage | FoldX*. http://foldxsuite.crg.eu/. Accessed: 2019-5-28.

[46] Sankar Basu and Björn Wallner. "DockQ: A Quality Measure for Protein-Protein Docking Models". In: *PLoS One* 11.8 (Aug. 2016), e0161879.

[47] Et al Uziela K. *ProQ3D: improved model quality assessments using deep learning. - PubMed - NCBI*. https://www.ncbi.nlm.nih.gov/pubmed/28052925. Accessed: 2019-5-31.

8 Appendix

8.1 Randomized parititon heatmaps

Figure 31: Heatmaps of the randomized partitions. For each of the groups of peptide, TCR and MHC sequences, each sequence was compared with all other sequences using pairwise alignment. The identity score was plotted as a heatmap. Clustering of high % identity scores was done per partition. The partitions are ordered from 0 - 4 along the x and y-axis, separated by white borders. The diagonal boxes represent sequences within a partition compared to itself, with the red squares within each box representing a cluster of the same or high identity percentage sequences. % identity score calculated from pairwise-alignment using Bioconda [43]

Figure 32: Heatmaps of the randomized partitions. For each of the groups of peptide, TCR and MHC sequences, each sequence was compared with all other sequences using pairwise alignment. The identity score was plotted as a heatmap. Clustering of high % identity scores was done per partition. The partitions are ordered from 0 - 4 along the x and y-axis, separated by white borders. The diagonal boxes represent sequences within a partition compared to itself, with the red squares within each box representing a cluster of the same or high identity percentage sequences. % identity score calculated from pairwise-alignment using Bioconda [43]

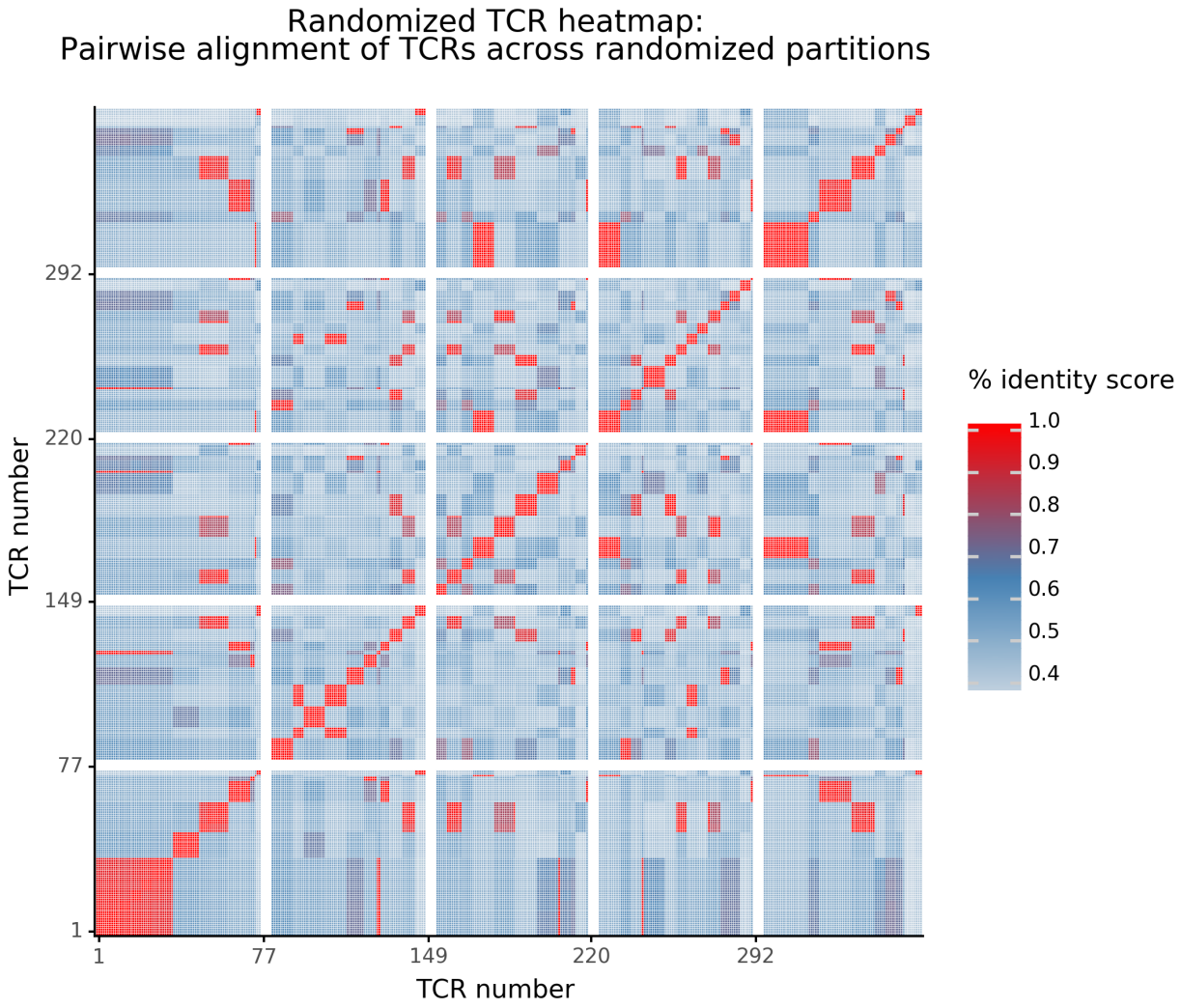


Figure 33: Heatmaps of the randomized partitions. For each of the groups of peptide, TCR and MHC sequences, each sequence was compared with all other sequences using pairwise alignment. The identity score was plotted as a heatmap. Clustering of high % identity scores was done per partition. The partitions are ordered from 0 - 4 along the x and y-axis, separated by white borders. The diagonal boxes represent sequences within a partition compared to itself, with the red squares within each box representing a cluster of the same or high identity percentage sequences. % identity score calculated from pairwise-alignment using Bioconda [43]

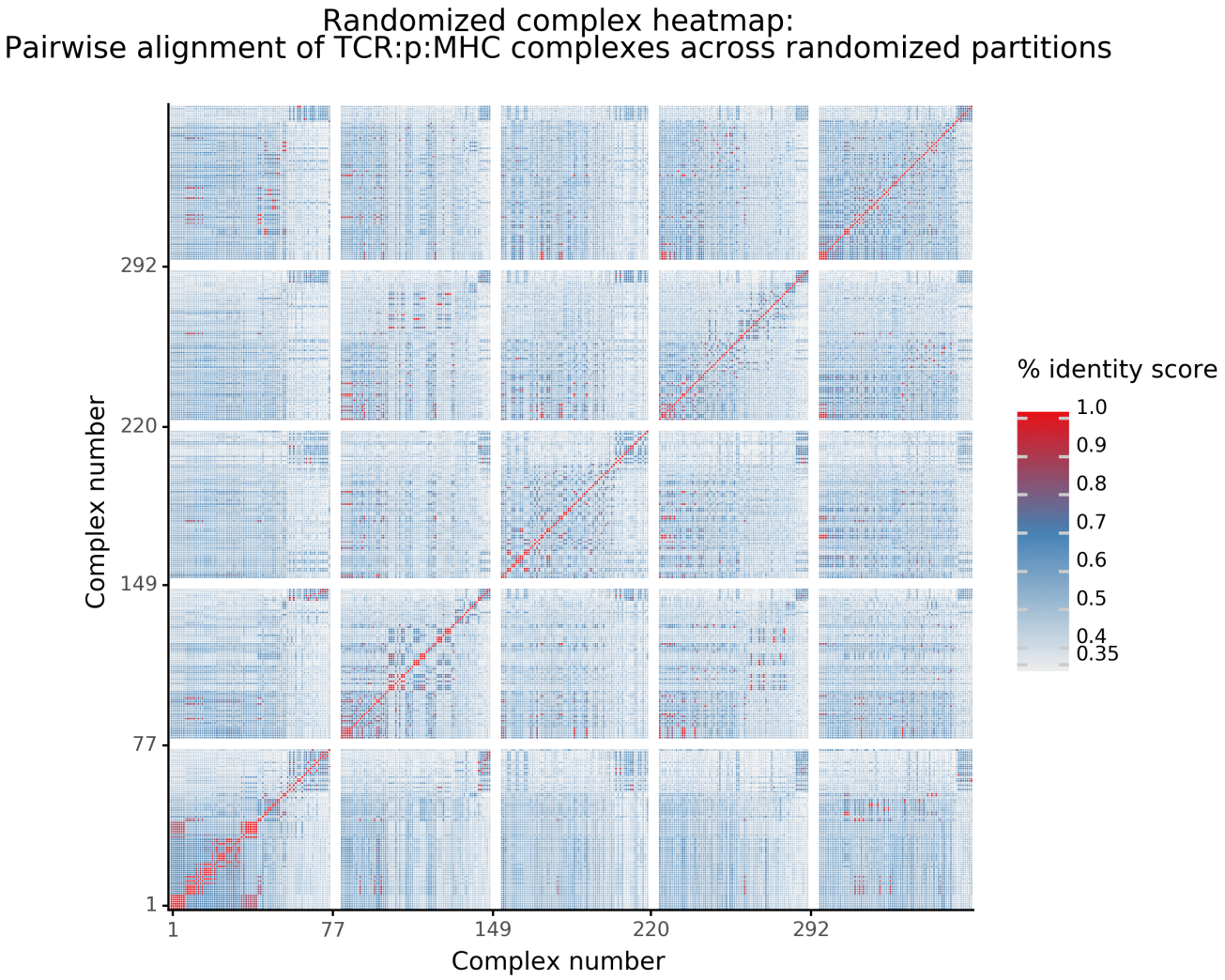


Figure 34: Heatmaps of the randomized partitions. For each of the groups of peptide, TCR and MHC sequences, each sequence was compared with all other sequences using pairwise alignment. The identity score was plotted as a heatmap. Clustering of high % identity scores was done per partition. The partitions are ordered from 0 - 4 along the x and y-axis, separated by white borders. The diagonal boxes represent sequences within a partition compared to itself, with the red squares within each box representing a cluster of the same or high identity percentage sequences. % identity score calculated from pairwise-alignment using Bioconda [43]

8.2 PDB data [28]

PDB ID	MHC-I	Peptide	PDB ID	MHC-I	Peptide
3vxr	HLA-A24:02	RYPLTFGWCF	3tjh	H2-Ld	SPLDSLWWI
1g6r	H2-Kb	SIYRYYGL	3qdg	HLA-A02:01	ELAGIGILTV
5c07	HLA-A02:01	YQFGPDFPIA	3gsn	HLA-A02:01	NLVPMVATV
4mnq	HLA-A02:01	ILAKFLHWL	4qok	HLA-A02:01	EAAGIGILTV
4qrp	HLA-B08:01	HSKKKCDEL	4jff	HLA-A02:01	ELAGIGILTV
2vlr	HLA-A02:01	GILGFVFTL	3utt	HLA-A02:01	ALWGPDPAAA
2oi9	H2-Ld	QLSPFPFDL	1mwa	H2-Kb	EQYKFYSV
4jfe	HLA-A02:01	ELAGIGALTV	5c08	HLA-A02:01	RQWGPDPAAV
5bs0	HLA-A01:01	ESDPIVAQY	5brz	HLA-A01:01	EVDPIGHLY
5c0c	HLA-A02:01	RQFGPDWIVA	3w0w	HLA-A24:02	RFPLTFGWCF
2vlk	HLA-A02:01	GILGFVFTL	3qeq	HLA-A02:01	AAGIGILTV
4g8g	HLA-B27:05	KRWIILGLNK	2e7l	H2-Ld	QLSPFPFDL
3e3q	H2-Ld	QLSPFPFDL	2bnq	HLA-A02:01	SLLMWITQV
5c09	HLA-A02:01	YLGGPDFPTI	3mv8	HLA-B35:01	HPVGEADYFEY
4prp	HLA-B35:01	HPVGQADYFEY	5eu6	HLA-A02:01	YLEPGPVTV
5c0a	HLA-A02:01	MVWGPDPLYV	3uts	HLA-A02:01	ALWGPDPAAA
5hhm	HLA-A02:01	GILGLVFTL	3qdm	HLA-A02:01	ELAGIGILTV
4eup	HLA-A02:01	ALGIGILTV	5d2n	HLA-A02:01	NLVPMVATV
3ffc	HLA-B08:01	FLRGRAYGL	3qdj	HLA-A02:01	AAGIGILTV
2ypl	HLA-B57:03	KAFSPEVIPMF	5hho	HLA-A02:01	GILEFVFTL
3vxs	HLA-A24:02	RYPLTLGWCF	4mji	HLA-B51:01	TAFTIPSI
2f54	HLA-A02:01	SLLMWITQC	5men	HLA-A02:01	ILAKFLHWL
4mxq	H2-Ld	SPAPRPLDL	2f53	HLA-A02:01	SLLMWITQC
2p5w	HLA-A02:01	SLLMWITQC	2pye	HLA-A02:01	SLLMWITQC
3hg1	HLA-A02:01	ELAGIGILTV	3tfk	H2-Ld	QLSDVPMDL
5e9d	HLA-A02:01	ELAGIGILTV	3mv7	HLA-B35:01	HPVGEADYFEY
2nx5	HLA-B35:01	EPLPQGQLTAY	2p5e	HLA-A02:01	SLLMWITQC
3vxm	HLA-A24:02	RFPLTFGWCF	5c0b	HLA-A02:01	RQFGPDFPTI
4l3e	HLA-A02:01	ELAGIGILTV	4jfd	HLA-A02:01	ELAAIGILTV
5euo	HLA-A02:01	GILGFVFTL	3mv9	HLA-B35:01	HPVGEADYFEY
2bnr	HLA-A02:01	SLLMWITQC			

8.3 Feature importance [28]

Importance	Feature	Importance	Feature
		0.0067	AB_entropy sidechain
0.0429	PA_energy Ionisation	0.0066	M_backbone clash
0.0249	PA_electrostatic kon	0.0065	PA_Electrostatics
0.0239	A_Backbone Hbond	0.0064	MP_electrostatic kon
0.0237	MP_Interaction Energy	0.0063	AB_Sidechain Hbond
0.0185	A_energy Ionisation	0.006	P_backbone clash
0.0138	AB_entropy mainchain	0.0057	A_entropy sidechain
0.013	PB_entropy sidechain	0.0056	AB_Electrostatics
0.012	B_energy Ionisation	0.0055	M_disulfide
0.0112	PB_Electrostatics	0.0055	MP_Electrostatics
0.0099	PA_Solvation Hydrophobic	0.0055	MB_torsional clash
0.0097	P_entropy sidechain	0.0054	MP_Backbone Hbond
0.0095	P_total energy	0.0053	B_torsional clash
0.0092	A_total energy	0.0052	MB_energy Ionisation
0.0088	MP_torsional clash	0.0052	MA_Electrostatics
0.0083	P_entropy mainchain	0.0052	A_Electrostatics
0.0082	MA_helix dipole	0.0051	MA_Van der Waals clashes
0.0081	PB_Interface Residues	0.0051	AB_Solvation Polar
0.0079	AB_backbone clash	0.005	PA_entropy mainchain
0.0075	MA_torsional clash	0.005	B_disulfide
0.0075	MA_Van der Waals	0.005	A_backbone clash
0.0074	PA_Interaction Energy	0.0048	MB_electrostatic kon
0.0074	B_entropy sidechain	0.0046	PA_torsional clash
0.0073	P_Solvation Polar	0.0046	MP_Sidechain Hbond
0.0073	MA_Solvation Polar	0.0046	B_Sidechain Hbond
0.0069	MA_entropy sidechain	0.0046	A_Solvation Hydrophobic
0.0067	MP_Solvation Polar	0.0046	A_Sidechain Hbond
0.0067	MP_IntraclashesGroup2	0.0045	B_entropy mainchain
0.0067	MB_Interface Residues	0.0044	P_Solvation Hydrophobic
0.0067	MA_Interaction Energy	0.0043	PB_Van der Waals clashes