The BERT architecture for the prediction of peptide presentation by MHC class I proteins

Hans-Christof Gasser

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2021

Abstract

One of the biggest obstacles faced in prolonging human life expectancy in high income countries today is cancer. In addition, recent events have highlighted that viruses still pose a serious threat. The major histocompatibility complex (MHC) class-I pathway facilitates the detection of both of these by the immune system. It presents parts of proteins present inside a cell (peptides) on its membrane surface. If visiting immune cells detect non-self peptides, they can terminate the cell.

Being able to predict which peptides will get presented and which not can help improve the efficiency of certain therapies. Current state of the art (SOTA) models mainly use ensembles of shallow fully connected neural networks. In this thesis, we explore the application of BERT to this task and use modern interpretation frameworks to gain biological insights.

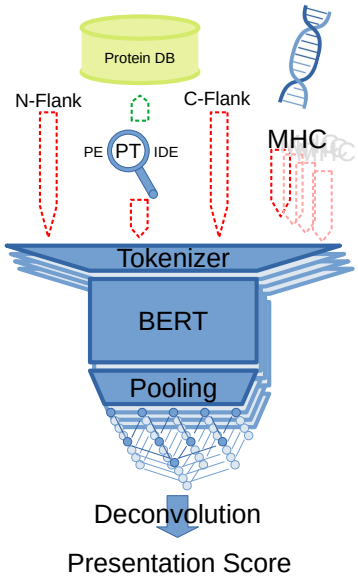


Figure 1: Graphical summary

As can be seen in Figure 1 our model (ImmunoBERT) takes as input an amino acid sequence consisting of several parts. The main one is the peptide for which we want to predict presentation. The next one is the peptide's surrounding in its source protein - which is the result of mapping the peptide to a protein database. And finally, we also input a representation of the MHC class I protein which potentially presented the peptide. We use a novel representation for this sequence and feed it into a BERT model. Based on experiments we chose a fitting pooling layer and feed its output into a multilayer perceptron. The fact that there are up to six MHC proteins in each individual is a complication that requires us to perform deconvolution.

Using motifs and a novel application of SHAP and LIME to this domain, we identify the most important parts of the input sequence. In particular, we find that amino acids close to the N- and C-terminals of the peptides are highly important. Also, some positions of MHC proteins (in particular in the A, B and F pockets) are often assigned a high importance ranking. We also observe that the flanks have little importance and this decreases with distance to the peptide.

Acknowledgements

This thesis would not have been possible without the support and understanding my parents have provided throughout my life. Thank you for this - in particular for being so accepting of my extraordinary career choices.

It would also not have been possible without Ajitha Rajan, Javier Alfaro and Georges Bedran who have proposed this exciting area of research, were always happy to speak about the project and through discussions jointly supervised the project. Thank you.

Also, I thank the PL-Grid and CI-TASK Infrastructure, Poland, for providing their hardware and software resources.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Hans-Christof Gasser*)

Table of Contents

Chapter 1

Introduction

The immune system defends the human body from a broad range of threats. Some of these manifest inside the body's own cells. For example viruses and cancer both utilize the cell's gene expression system to facilitate their own reproduction and spreading. Cytotoxic T-lymphocytes (CTL), a special kind of T-cells, can detect affected cells and terminate them. To do so, they require a way to 'look inside' the body's cells. A system revolving around the MHC proteins facilitates this in many higher vertebrates [1]. MHC proteins are also called Human Leukocyte Antigen (HLA) in humans. This thesis focuses on the MHC class I (MHC-I) proteins and their antigen presentation pathway which is active in all nucleated cells of the human body [2]. Proteins present in these cells are constantly being fragmented into smaller pieces - so called peptides. These then bind to the MHC proteins forming peptide:MHC protein complexes (pMHCs), which are then transported to the cell membrane. There, the peptides get presented on their 'MHC pedestal' to the outside world. The pMHC are antigens for the T-cell receptors (TCRs). The exact part of the antigen where the TCR binds is also called an epitope. [3, 4]

An infection by a virus or a cancer causing mutation, can both result in the production of proteins that would otherwise not be present in a healthy cell. Eventually this leads to the presentation of neo-antigens (non-self pMHC) to the outside world [5, 6]. Dependent on whether the CTL consider the peptides presented to them as self or non-self, they decide to terminate the cell. The human body, therefore, already has the ability to fight viruses and cancers. Many cancers, however, have evolved mechanisms to circumvent the immune system. An example is tumor-induced immune suppression [7]. Check-point inhibitors are used to counter this. Peptide-based vaccines are another tool that can be used to strengthen the immune response. Identifying which

peptides will most likely be presented by a cancerous cell and elicit an immune response (immunogenic peptides [8]) is an important component in the development of peptide-based vaccines [9]. One way to achieve this *in silico*, is by utilizing machine learning (ML). The usage of the BERT model to predict the first part of this - peptide presentation, is the topic of this thesis.

Although an important step, presenting foreign peptides (neo-antigens) is not sufficient to elicit an immune response. For example, the presented peptide needs to bind sufficiently strong to a TCR as well. This step is not part of our analysis. Another group of models deals with this step (for example DeepImmuno [8]). Many other factors need to be considered as well in vaccine development. For example, the MHC-I pathway is typically only capable of presenting the cell's endogenous protein fragments. Several methods can be used to internalize the peptides from a vaccine. For example professional antigen presenting cells (e.g. dendritic cells) can internalize exogenous material and then present it to the outside world via the MHC-I pathway. Glycans and other adjuvants might be required to facilitate the vaccine uptake [7]. This is an active research area.

As stated above, this thesis explores the application of the BERT architecture to the epitope presentation prediction via the MHC-I pathway in humans. To do so we first develop a for this problem novel architecture based on a pretrained BERT model. The resulting model achieves competitive performance to current SOTA models on an independently curated benchmark set. In addition, a motif analysis confirms that our model has indeed learnt MHC allele dependent peptide specificities. In a novel step we then apply SHAP and LIME to find the parts of the peptide, MHC protein and surrounding flanks of the peptide, that are particularly relevant for presentation.

Chapter 2 presents the relevant biological and ML concepts before lining out the current SOTA in T cell epitope prediction. Then Chapter 3 introduces the datasets and methodology we used. The results achieved and the interpretation are presented in Chapter 4. We finish with a small discussion and suggestion for future research in Chapter 5.

Chapter 2

Background

A solid understanding of the underlying biology is required to design a model capable of predicting the peptides presented by a cell. So, this is introduced before the relevant ML concepts.

2.1 Biology

Our discussion of the biology begins with a subsection about proteins, which also lines out why peptide presentation is a suitable mechanism to transfer information about a cell's state of health to the outside world. Then we present the MHC-I pathway that implements the presentation process. The experimental datasources used for its examination in the context of presentation prediction are discussed then. Finally, we will introduce the MHC protein and its naming convention.

2.1.1 Proteins

Proteins play a central role in the human body. They consist of amino acids (AAs) chained together via peptide bonds [10]. In general, an AA consists of an amino group a central carbon (with a hydrogen atom), an AA specific side chain and on the opposite side to the amino group, there is a carboxyl group [11]. During the formation of the peptide bond, the carboxyl group of one AA is connected to the amino group of the next one. The end of the chain with the amino group is called N-terminus the end with the carboxyl group is called C-terminus. There are 20 AAs encoded by the genetic code [10]. Each of them can be represented by a letter. Once joined together by peptide bonds, we refer to the single AAs as residues. Short chains of residues are referred to

as peptides, while longer ones are referred to as polypeptides.

The blueprints for the proteins produced by a cell are stored in deoxyribonucleic acid (DNA) molecules. These are chains of nucleotides. Each nucleotide contains one of four bases. Parts of these very long DNA molecules can be transcribed (copied) into shorter messenger ribonucleic acid (mRNA) molecules. Dependent on the bases, ribosomes then translate groups of three mRNA nucleotides into AAs that get appended to a protein in production [10].

While peptides behave like flexible strings, longer polypeptide chains and proteins typically adopt a 3D molecule structure due to the features of their residues (e.g. hydrophobic and hydrophilic regions) and bonds forming between them. We distinguish four levels of protein structure - with the lower levels influencing the emergence of the higher ones [11]. Primary structure refers to the polypeptide's sequence of AAs. It can be represented as letters (one per AA), written from N to C-terminus. Due to hydrogen bonds forming between distant AAs in the polypeptide a secondary structure emerges. This could be structural features like α-helices and β-sheets. These secondary structures can get packed to form regions of defined 3D shape - tertiary structures like β-barrels or coiled coils. Eventually, several of these (then called domains or subunits) can combine - referred to as quaternary structure [11].

In eukaryotic proteins, one domain often corresponds to one exon (translated area of the gene code, interrupted by introns - not translated ones). Similar exons coding for similar domains are often found at distant parts of the genome. They are like "modules" that can be used in several proteins. The genes coding for these typically share a common ancestor. Such a relationship is referred to as homology and they are often similar on a functional level but may differ in AA sequence [10, Chapter 3]. This makes splitting up the dataset into train, test and validation set to assess generalization capability more complex (see Subsection 3.1.2).

Proteins perform tasks as diverse as breaking up nutrition into its components, muscle movement and sustaining cell structure. Because of the wide range of functions they can perform, controlling which proteins are present in a cell is so important. For example, cancer is essentially a set of DNA mutations (and/or epigenetic changes) that change cell behaviour by causing changes to the cell's proteome. In healthy cells, cell division is strictly controlled and most cells enter a non-dividing state. However, if sufficient pathogenic changes to the genetic material accumulate, the brakes on this system can loosen (e.g. due to mutations that render tumour suppressor proteins useless) and the inappropriate activation of oncogenes (supporting cell division) can lead

to uncontrolled propagation of the tumour [10, Chapter 41, Section X]. In comparison, a defining feature of viruses is their inability to reproduce themselves. They, therefore, need to capture the host's protein expression system to do so. This will also lead to changes to the cell's proteome.

2.1.2 The MHC class-I pathway

Therefore, presenting information about the proteins present inside the cell transfers valuable information about the cell's state of health to the outside world. This is achieved via the MHC-I pathway which has the following steps [12]:

1. Proteasomes fragment the cell's internal proteins into peptides
2. transporter associated with antigen processing (TAP) proteins transport these peptides into the endoplasmic reticulum (ER)
3. In the ER there reside membrane bound MHC-I proteins, which can bind with the peptides to form pMHC.
4. The pMHCs get transported to the cell membrane, where the MHC protein acts as a pedestal for the peptide and presents it to the extracellular fluid
5. A CTL with a fitting TCR could bind a presented neo-antigen. This might trigger an immune reaction, as CTLs do not strongly bind to "self" peptides (the body's own peptides) but only non-self ones. CTLs are trained for this in the thymus by positive- and negative selection [3].

With regards to step 1 Figure 2.1 shows that inside the cell's cytoplasm, proteins are constantly degraded into peptides (mostly of 7-8 AAs length [13]) by proteasomes. There are three different types of proteasomes, having different specificities [12]. While the proteasomes generate the exact C-terminal cut, the produced fragments can further be cut down by peptidases on their N-terminals in the cytoplasm [14]. To reach the ER these are bound by TAP proteins that are responsible for the peptides' transfer to the ER (step 2). The affinity of these TAP proteins peaks at peptide lengths of 11 AA [13]. Since the peptides go through both filters - proteasomes and TAP - the distribution of those making it to the ER concentrates at 9-10 AAs. With regards to sequence specificity, MHC binding is far more restrictive than proteasomal cleavage as well as TAP transport and it seems their main role is to deliver right length peptides to the ER and not to act as a sequence filter [13].

In the ER the peptides encounter membrane bound MHC proteins. It has been shown that the binding of MHC proteins to peptides (step 3) is a particularly restrictive step in the MHC-I pathway - with only roughly 1 in 200 randomly generated peptides binding [15]. These MHC proteins differ between individuals. In fact, each individual has several different MHC proteins as well. Within the human MHC located on chromosome 6 at 6p21.3 there are three main loci coding for MHC-I proteins: HLA-A, HLA-B and HLA-C [2]. As *Homo sapiens* is a diploid species, each human therefore can express up to six different HLA-A, HLA-B and HLA-C proteins. Each of those can have different binding properties. As these gene regions are highly polymorphic and there are many different HLA alleles in the human population, there is a large variety in immunopeptidomes (the entirety of all presented peptides) across humanity. In fact, currently there are more than 4,064 HLA-A, 4,962 HLA-B and 3,831 HLA-C proteins known [16]. Each of these can bind roughly 1,000 to 10,000 different peptides [17].

If the binding between MHC and peptide in step 3 is successful, a complex (pMHC) forms. This can then migrate to the cell membrane. Here, the peptide is presented to the extracellular environment by the MHC protein - which acts as a pedestal for the peptide (step 4). Whether this results in an immune reaction or not, depends on whether a CTL with a fitting TCR gets in contact with the pMHC and the presence or absence of other signals. Approximately half of the peptides will not be detected due to a limited TCR (CD8+) repertoire [15]. In total, [15] estimate that just one out of 2000 random peptides (8-11 AA long) of a foreign antigen will be presented by a particular MHC protein and cause an immune reaction.

Interestingly, in addition to their role in adaptive immunity, MHC proteins also have an inhibitory influence on natural killer cells [2] (part of innate immunity). This way, a tumour cell that switches off MHC production completely will also be destroyed. However, what happens after peptide presentation at the cell surface is not part of our examination.

2.1.3 Experimental datasources

As the most restrictive step in epitope presentation is MHC binding, measuring the binding affinity (BA) between a particular MHC protein and a peptide *in vitro*, can give us some insight into how likely it is that a particular peptide will be presented to the extracellular environment. Early presentation predictors were, therefore, only trained on this BA data. Its biggest disadvantage is, that it is costly to carry out the

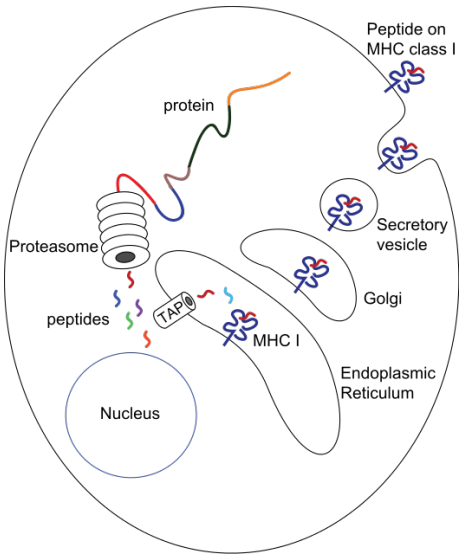


Figure 2.1: MHC-I pathway [18]

experiments that then only generate little data [19].

In the modern high throughput eluted ligand (EL) approach, the whole immunopeptidome of the cell is harvested and then the peptides are identified using mass spectrometry [20]. This identifies many peptides at once. However, it is typically not possible to measure which of the up to six different MHC alleles presented which eluted peptide. Monoclonal cell lines, that only express a single MHC allele are a potential way around this limitation. There also exist algorithms to identify which allele is responsible for a peptide's presentation. This step is called "deconvolution" in the literature. However, it also often relies on data that can be unambiguously ascribed to a single allele to kickstart it (see Subsection 2.2.1).

Another disadvantage of the EL approach is, that it does not generate definitive negative examples. It only reports about the presence of a peptide at the cell's surface. It cannot assert the absence of a peptide from the individuals immunopeptidome. For example a peptide present in the human genome, might only not be presented by the cell because its protein is not being expressed by the particular cell. Despite it's drawbacks, the high quantity of data generated by this approach will make it the backbone of our examinations.

2.1.4 MHC proteins and peptide binding

There are several types of MHC proteins. We will only concentrate on the classical MHC-I proteins here. Each of those is a hetero-dimer consisting of the non-covalently

bound transmembrane heavy chain α and the light chain β_2-microglobulin [21]. The α chain has two membrane distal α_1 and α_2 domains that form a potentially peptide binding groove. The α_1 and α_2 domains each consist of a sheet of four antiparallel beta strands and a single helical region. The beta sheets of the two α domains together, form the floor of the groove, while the helices each form one of its sides [22]. The residues lining this groove are also the ones showing the highest number of polymorphism. Six pockets (A-F) were identified for peptide binding. In particular pockets A and F are responsible for binding the peptide's N- and C-termini [21]. Most peptides are firmly embedded into the binding groove and terminated on both sites by the A and F pocket. So, there is little variation with regards to bound peptide length. Bulges and in a very few cases protrusions out of the groove can, however, lead to longer peptides binding [21].

As already mentioned, there are many different MHC alleles. So, the following naming convention has been adopted in humans [23]. The full name of a MHC allele could look like: HLA-A*04:18:03:01N. The letter after HLA ('A') signifies the gene that we are referring to. Then, the following number ('04') stands for the allele group. This often corresponds to the serological antigen carried by it. The second number ('18') refers to the subtype. Importantly, HLA alleles that differ in any of those three parts ('A', '04' or '18'), will also differ on an amino acid level. So, these parts determine the specific HLA protein. In contrast, the third number ('03') only signifies substitution mutations in the gene that do not lead to a different protein. The last number ('01') orders the alleles with respect to mutations in the introns. Finally, the suffix ('N') can be used to transfer information about the expression of the allele. For example, if an allele would not be present on the cell surface. For our purposes, we will only consider the gene and the first two numbers.

Differences in HLA proteins present in an individual do have real world consequences. For example, while HLA-B*42:01 is linked to better outcomes for HIV sufferers, HLA-B*42:02 is not, despite them only differing in one amino acid position [24]. In contrast, HLA-B*35:03 is suspected to be connected with fast AIDS progression, while the only at one position differing allele HLA-B*35:01 is not [24]. The differences are not limited to viral infections. In the Gambia HLA-B*53 was shown to be present in 25% of healthy and mild malaria cases while only to be present in 15% of severe cases - suggesting an approximately 40% reduction in the risk of severe malaria [2].

2.2 Machine Learning

Based on the last section, predicting the immunopeptidome of a particular individual is difficult mainly because of three facts. First, as any of the up to six MHC-I alleles present in the cell might be responsible for a peptide observation in an EL experiment, the problem does not fit well into the supervised learning framework in which x results in y. Second, the high throughput eluted ligand assays only produce positive examples resulting in the need to create artificial negative ones (decoys). This is, third, aggravated by the fact that in reality most peptides will not be presented on the cell's surface (imbalanced dataset). In this section we discuss these issues in turn and then present the backbone architecture used by us - BERT, before introducing the techniques we will employ for its interpretation.

2.2.1 Deconvolution

With regards to the first obstacle, deconvolution refers to the process in which each peptide is associated to its presenting MHC protein [19]. Several approaches have been put forward to deal with this challenging task. We already mentioned the possibility to use cell lines only expressing a single MHC allele. However, bioinformatics has also produced *in silico* ways to tackle this task. For example the usage of unsupervised sequence clustering - like Gibbs clustering. This, however, needs manual curation like providing the correct number of clusters and sometimes also during the assignment of those clusters to MHC proteins [19]. As an alternative, [25] have developed a mixture model approach that models the likelihood of the observed eluted ligands of a sample (with up to six different HLA proteins) as the mixture of several position weight matrices (PWMs). They observe that most of the observed peptides show a good match to only one of those PWMs and so can be assigned to this. These PWMs are then assigned to known PWMs of alleles from IEDB.

Looking at deconvolution from the perspective of weakly supervised learning - in particular Multi Instance Learning (MIL) - points us towards solutions based on single allele models. MIL assumes that several instances x_{n_j} do each belong to a class $y_{n_j} = 1$ or not $y_{n_j} = 0$. However, these y_{n_j} are not directly observed. We only know the classifications y_n for bags X_n of instances ($X_n = (x_{n_1}, x_{n_2}, ..., x_{n_m})$). The presence of a single instance belonging to a class in a bag is sufficient for the whole bag to be labelled with this class ($\exists j, y_{n_j} = 1 \Rightarrow y_n = 1$). Only if all instances do not belong to the class, then the bag's label is zero ($\forall j, y_{n_j} = 0 \Rightarrow y_n = 0$). This means that $y_n = \max_j y_{n_j}$

[26]. These bag level labels y_n are what is contained in the data. There are two main categories of MIL algorithms [27, 26]. The traditional approach is instance-level based while the alternative is embedding-level based.

In instance-level based methods, a function f estimates the classification for each instance $\hat{y_{n_j}} = f(x_{n_j})$. The class of the whole bag is estimated as the maximum estimate of the bag's instances $\hat{y_n} = F(X_n) = \max_j f(x_{n_j})$ [28]. In contrast, in embedding-level approaches, f does not directly estimate the instance's classification but only produces an embedding. A pooling operation is used to obtain a fixed dimensional representation of the whole bag. This representation is then fed into a classifier for the whole bag. Although [27] have demonstrated, that the latter approach can have higher performance, we will stick to the former one as it fits better to the fact, that our data also contains observations that can be clearly assigned to a single allele. Also, the traditional approach is implicitly loosely followed by NetMHCpan's deconvolution mechanism [19] and BERTMHC [29].

So, in the instance-level approach the estimation for the bag's classification is the maximum of the individual estimates. The maximum is not differentiable everywhere which is a requirement for gradient based learning algorithms. There is, however, a differentiable version of the *max* available - the LogSumExp ($\max_j x_j \approx \frac{1}{\beta} \log \sum \exp \beta x_j$). However, since the astonishing success of the ReLU activation function in neural networks, the concerns about not being differentiable locally seem to have eased a bit and we will see that the current SOTA model NetMHCpan also implicitly relies on the max pooling via the NNAlign_MA framework (see Subsection 2.3.1).

2.2.2 Negative examples

The previous section argued that our problem differs from the traditional supervised learning setting with regards to the inputs. Here we argue, that it also differs with regards to the output. In the supervised setting, the training data is assumed to be fully labelled [30]. However, in our case, there are no (few if we were to include BA data) negative examples available. We do have a lot of positive examples (peptides eluted from the cell's surface) and even more unlabelled data (the whole part of the human proteome expressed in a sample). This fits into the category of positive and unlabelled (PU) learning, which is closely linked to the problem of one-class classification in which only positive examples are available. [31] have analysed the PU setting in particular with regards to the transporter classification database for mem-

brane transport protein analyses (TCDB) which stores proteins involved in signalling across membranes (positive examples) and the SwissProt database (unlabelled examples). In addition to the input variable x and the class label $y \in \{0,1\}$ they introduce s, that signifies whether an example is labeled ($s = 1$) or not ($s = 0$) and assume:

- Only positive examples are labeled: $P(s = 1|x, y = 0) = 0$
- For a positive example, whether it is labelled or not is independent of x: $c = P(s = 1|x, y = 1) = P(s = 1|y = 1)$

They call a classifier $g(x) = P(s = 1|x)$ that distinguishes between labelled and unlabelled examples a non-traditional classifier and proof that:

$$P(y = 1|x) = \frac{P(s = 1|x)}{c} \tag{2.1}$$

Therefore, $f(x) = \frac{g(x)}{c}$ is a traditional classifier for the original class. Furthermore, [31] show three ways to estimate c using the non-traditional classifier g and a validation set V, consisting of positive (P) and unlabelled data. The first estimator is $\hat{c}_1 = \frac{1}{|P|}\sum_{x \in P} g(x)$, the second $\hat{c}_2 = \frac{\sum_{x \in P} g(x)}{\sum_{x \in V} g(x)}$ and the third one $\hat{c}_3 = \max_{x \in V} g(x)$. These estimators rely on the classifier g being powerful enough to capture the probability exactly (not mis-specified). They argue, that if $g(x) = P(s = 1|x)$ only approximately, then $\hat{c}_1$ is the best choice.

While Equation 2.1 directly points towards adjusting the threshold of $g(x)$, that is typically 0.5 to the value $\frac{0.5}{\hat{c}_1}$ to construct the classifier $f(x)$ for the actual class we are interested in predicting, they also propose a second way that leaves the threshold unchanged to achieve this. This relies on splitting up every unlabelled example into a positive one and a negative one and train the classifier using different weights for these. In an example analysis they perform with the TCDB and SwissProt data mentioned before, they find that those two versions perform similarly well.

Their results deliver the theoretical justification for our approach. However, we will not re-scale the threshold since we will only benchmark our classifier vs generated negative data (decoys). If we, however, were to use it for real prediction, we would have to scale the threshold in accordance to Equation 2.1.

2.2.3 Imbalanced datasets and metrics

Although we only have positive examples, the unobserved implicit negative ones outweigh the observed positive ones by a considerable margin as only one out of roughly

200 randomly generated peptides will bind (see Subsection 2.1.2). This raises the question of how to deal with this imbalance. Many standard learning algorithms experience significant problems when dealing with imbalanced data - however to what degree the original dataset should be balanced is still open to debate [32]. It seems that it has to be answered on a case by case basis. However, Weiss and Provost [33] have shown that selecting accuracy as the relevant performance metric tends to favour sticking close to the natural class distribution, while selecting area-under-the-curve (AUC) based metrics favours more balanced class distributions.

He and Garcia [32] argue, that accuracy is in general not a good metric when assessing performance on highly skewed datasets. If the majority- to minority class ratio was 999:1, then a classifier always predicting the majority class has an accuracy of 99.9% - despite it missing all minority class examples. There are alternative metrics available that result in a more balanced assessment.

A typical binary classifier scores an example with values between 0 and 1. Scores above a predefined threshold are considered positive (belonging to the class), those below negative. If a positive prediction is correct, it is a true positive (TP) example, if incorrect a false positive (FP) one. Equally, a correct negative assignment is a true negative (TN) example, and an incorrect one a false negative (FN). In the equations below, TP(threshold) means the threshold dependent count of all TP examples across the relevant dataset. Equivalently, FP, TN and FN are defined [32]:

$$\text{Recall or TPR(threshold)} = \frac{TP(threshold)}{TP(threshold) + FN(threshold)} \tag{2.2}$$

$$\text{FPR(threshold)} = \frac{FP(threshold)}{FP(threshold) + TN(threshold)} \tag{2.3}$$

$$\text{Precision(threshold)} = \frac{TP(threshold)}{TP(threshold) + FP(threshold)} \tag{2.4}$$

$$\text{AP} = \sum_n (Recall(T_n) - Recall(T_{n-1})) \cdot Precision(T_n) \tag{2.5}$$

A common way to visualize a classifier's diagnostic ability is the receiver-operating-curve (ROC). It emerges when we let the classifier's threshold slowly increase from zero to one and plot the false positive rate (FPR) (Equation 2.3) on the x-axis and the true positive rate (TPR) (Equation 2.2) on the y-axis. A good classifier will have a high area under this emerging curve - the AUC-ROC.

An alternative visualizations are precision-recall (PR) curves. They emerge, when we let the threshold slowly increase from zero to one and plot the recall (Equation 2.2) on the x-axis and the precision (Equation 2.4) on the y-axis. The average precision

(AP) [34] is a common way to summarize a PR curve. It is calculated based on a series of increasing threshold values (T_n, see Equation 2.5).

We will use the AUC-ROC due to its easy interpretation. However, [32] argue that PR curves provide a more informative representation of a model's performance under highly imbalanced data. Therefore, we will visualize the performance using PR curves and also calculate the AP metric.

2.2.4 BERT

The backbone of our classifier is a Bidirectional Encoder Representations for Transformers (BERT) [35] model. This belongs to the Transformer class of neural networks that have been pushing the SOTA performance in many natural language processing tasks. In particular big models like GPT-3 [36] have captured the public's imagination [37]. While the original Transformer consists of an encoder and a decoder [38], BERT only has an encoder stack, producing one contextualized embedding for each input token.

The revolutionary idea in transformers is the consequent usage of self-attention layers - signified by the original paper's title "All you need is attention" [38]. Attention enables a neuron to focus on the output of any preceding layer position. Crucially, it can focus on far away positions and directly access their outputs - without them going through many operations. This is particularly useful when dealing with long sequence data. In comparison, recursive neural networks struggle to take into account far away text parts (long-range dependencies) and are prone to vanishing gradients [38, 39, 40]. More recently, these advantageous properties of transformers have also been exploited for visual tasks by [40]. While convolutional neural networks (CNNs) can connect information from two distinct points in a picture only in their upper layers (small receptive fields in low ones), transformers can do this already in early ones [40].

Transformers use a very special type of attention. There are in general three types of attention. Hard attention, soft attention and self-attention [41]. Let us assume that for each example, a lower layer produces n vectors o_j as output. In our setting, each of these n vectors could correspond to an amino acid of a peptide we want to classify. Hard attention would only consider a subset of $m \leq n$ of these as input for the next higher layer - completely disregarding the others. This is a non-differentiable operation that we will not consider any further in this thesis. Soft-attention, in comparison, is often used for pooling. It calculates weights for each o_j (e.g. by calculating the dot

product of a learned vector with all o_j followed by a softmax operation). The weighted sum of the o_j is then the single fixed dimensional input vector to the following layer [42]. We will come back to this again when we discuss the various ways of pooling the output of our model's BERT backbone before feeding it into the classifier's head. Finally, self-attention is the form of attention used by transformers [38]. The standard implementation uses learned transformations to produce a value v_j, query q_j and key k_j vector for each output vector o_j. An intuitive way to think about those is to see the key vector as encoding what type of information is encoded in the position (similarly to the key in e.g. a hash table). The query vector then encodes which type of information a position in the next layer requires as input. If the information encoded at position j and the information required at position i are similar, then k_i and q_k should be similar. Therefore, a dot product (cosine simularity) is calculated for each of the n^2 possible position pairs. This represents the weighting that the v_j will have in the input to position i. In addition, BERT performs several of these in parallel (multiple heads), allowing a neuron to focus on several aspects at once.

Although transformers have been successful in tackling the long-range dependency issue, their biggest strength is also their biggest weakness. Self-attention assesses the importance of each lower layer output for each higher layer input. This defines a squared complexity task [43]. Therefore, BERT models are often limited to relatively short sequence lengths. There are initiatives to alleviate this. For example google's Reformer [44] network uses the sparseness of the attention matrix. Using locality sensitive hashing allows them to focus the attention calculation on the values with the closest indices to the keys. This approach allows them to reduce the complexity of the self-attention operation to $n\log(n)$. The application of these optimized models to our task is an promising area for future research.

BERT pretraining: The BERT framework [35] postulates two steps in training a model for a new task - pretraining and fine-tuning. For pretraining, a vast amount of unlabelled data is used to perform self-supervised learning tasks. The original BERT paper [35] uses Masked Language Modelling (MLM) and Next Sentence Prediction (NSP). In MLM, part of an input sentence (typically 15% of the tokens) are hidden - 80% with a masking token, 10% with a random token and 10% with the same token. The model has to fill only those positions with their original tokens. This contrasts to de-noising auto-encoders, that reconstruct the whole original input [35]. As MLM allows and requires the network to use information from both sides of the gaps, it encourages the emergence of bidirectional representations for the input tokens. So, the

final BERT layer's outputs can be seen as *context-sensitive* features. This contrasts to traditional embeddings, that are only dependent on a single token and not its context. By not always replacing the hidden parts with the masking token, [35] want to alleviate the mismatch between pre-training vocabulary and fine-tuning vocabulary (which typically does not include the masking token). They also mention that keeping the transformer in the dark about exactly which parts have been replaced, forces it to keep a contextual representation of every position. To encapsule relationships between sentences, NSP pretraining is used. This should prepare BERT for tasks like question answering and natural language inference [35]. The self-supervised learning task is to predict whether the second sentence input into the model follows after the first one. The pretrained protein BERT model we use (TAPE), was only pretrained using MLM.

TAPE [45]: In the domain of protein prediction, the Tasks Assessing Protein Embeddings (TAPE) transformer model [45] implements the hugging face interface for the BERT model. It has 12 layers with 12 attention heads each. It has been pretrained for 1 week on 4 GPUs on more than 32 million protein domains [45, supplement] from the Pfam database using masked-token prediction (MLM) [46]. Next to the BERT transformer model, [45] also pretrained several other models (LSTM, ResNet).

After the pretraining, they fine-tuned common SOTA task specific model architectures using the embeddings generated from these models as well as baseline embeddings (like one hot encoded amino acids). During fine tuning they trained the full stack (including the pretrained model). The TAPE framework includes tasks coming from three important areas of protein biology: structure prediction, detection of remote homologs and protein engineering. They find, that pretraining positively influences the performance on downstream tasks and that the best performing architecture (BERT, LSTM, ResNet) for embedding generation is task dependent [45, Table 2]. There are alternative protein transformer models available as well - like ProtTrans [47]. However, due to its small size, our limited computing resources and that it has already been fine tuned previously for MHC class II (MHC-II) prediction (see Subsection 2.3.3), we use the TAPE transformer model as the backbone for our model.

As transformers have shown impressive performance in many ML tasks [38, 36, 40, 45, 47], we want to explore whether they can also be applied to our task outlined in the introduction. Before, we present two frameworks for model interpretation and introduce the current SOTA approaches in this field.

2.2.5 Interpretable AI

To check, whether our model f has learnt relationships grounded in biology, we employ two explanatory frameworks: Local Interpretable Model-agnostic Explanations (LIME) [48] and SHapley Additive exPlanations (SHAP) [49]. We can only give a brief introduction on those two methods here.

LIME [48]: As its name suggests, LIME produces local (for a particular example) explanations, treating the model f to be explained as a black-box (model-agnostic). It is motivated by the observation that simple models are often easy to explain but deliver worse performance than more complicated ones [48]. Therefore, the LIME framework builds a surrogate model g for the actual model's behaviour around a particular example x. The standard surrogate model used by the package is a ridge regression. This is trained to explain the original model's predictions. As input it receives interpretable data. This could be a binary vector z' with the same dimension as the feature space of the original model. The binary vector is used to generate a mutated example. Where a feature of the binary vector is 1, the feature value of the original example is used. Where the binary vector's feature is 0, an imputed value is used. The imputation is domain specific. We will use the text version. In this case, the feature j (word) is just deleted if $z'_j = 0$. In contrast, LIME tabular samples values from a Gaussian around the feature mean or the original feature - the LIME package (https://github.com/marcotcr/lime) offers a lot of optionality here. The conversion from binary into original feature space is performed by the function $z = h_x(z')$ which depends on the example to explain x. In a next step, a distance to the original example is calculated, which is used as a weight for the sample in the training of the surrogate model. If a feature is particularly important for the prediction of the original example, then its addition in the binary feature should change the surrogate's estimate a lot. This means having a high coefficient.

Shapley values: Although intuitively clear, LIME has little theoretical grounding. In fact, it is well suitable to explain, how the predictions of the original model will change in the vicinity of the example to be explained, however, in general it fails to attribute the difference between the average prediction over the dataset and the example's prediction fairly to the various features. This is, because the only method that does this (and satisfies certain conditions) are Shapley values [50], which measure the contribution of each feature value to the prediction for a particular example. Shapley values have their origin in game theory and are the only way to fairly attribute the

outcome of a 'game' satisfying four quite basic axioms [49, supplement]:

- **Efficiency:** The sum of the Shapley values is equal to the original prediction
- **Symmetry:** If the inclusion of two features individually has the same effect on the estimator, then their Shapley values must be the same
- **Null effects:** Features not considered by the model, have a Shapley value of zero
- **Linearity:** A feature's, Shapley value for the sum of two models is the same as the sum of its Shapley values in the two models

A feature's Shapley value is computed by calculating the difference between the predicted value of a model trained to take into account that feature and one not taking it into account. This is done and averaged over all possible combinations of feature subsets, resulting in the feature's Shapley value. Equation 2.6 shows how the Shapley value of feature i from the feature set F is calculated for the prediction of an example x. f_S denotes the model trained only on the features in set S and x_S the subset of S features from x. Computing the whole expression is impractical for more than only a few features, due to the need to train and evaluate many models ($O(2^{\text{nr of features}})$). This led to the introduction of Shapley sampling values. These only sample a subset of all possible feature combinations and do not retrain models, but rather approximate the impact of removing variables by marginalizing these over the training set [49].

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S)] \qquad \text{[49, equation 4]} \quad (2.6)$$

SHAP [49]: Based on the idea of Shapley values, Lundberg and Lee [49] developed the SHAP package for the efficient approximation of SHAP values. These are the Shapley values under the assumption that the model's output restricted to a subset of the features is given by the expected model prediction conditioned on this subset.

To estimate SHAP values, the package offers several model specific ways as well as the model agnostic Kernel SHAP which we will use. It is based on the observation that a linear explanation model (like LIME) can be trained with certain parameters (in particular a special kernel) to approximate SHAP values. However, while LIME parameter choices are heuristic [49], Kernel SHAP trains the linear model in a specific way. Similarly to Shapley sampling values, Kernel SHAP does not consider all possible feature combinations, but just a few. The standard value is 2048 + 2 times the number of features. In addition, to estimate the conditional expectations, Kernel

SHAP requires the data's background distribution. This is provided in the form of representative examples. Ideally, these are the whole training set. In total, if we had 20 features and a training set of 1000 examples, this would lead to 2.088.000 (=1000 x (2048 + 2x20)) model predictions to estimate the SHAP values. Therefore, the SHAP developers suggest to use just few or only a single reference value for larger problems. The SHAP values will then, however, only explain the difference in prediction vs this reference value and not the overall set.

The main advantage of using a linear regression approximation vs directly the Shapley equations (see Equation 2.6) is that the regressor can approximate all Shapley values at the same time - resulting in higher sample efficiency [49].

2.3 Current state of the art

Now that we have discussed the necessary biological and machine learning methods, we want to take a look at the most popular models currently available. These have in particular evolved along three dimensions over the past decades.

The first one regards their input data. As outlined in Subsection 2.1.2, the most restrictive step in the MHC-I pathway is the binding of the MHC-I protein with the peptide. Early models like [51], and in fact even quite recent ones like NetMHCpan-3.0 [52] were, therefore, just considering BA data. Predicting binding affinity is theoretically a regression task. However, for example [52] transformed the binding affinity values to be between 0 and 1 by using the cut off transformation 1 - log(binding affinity IC50 in nM)/log(50,000). Also, an IC50 of 500 nM is often used as a threshold to distinguish between binders and non-binders [53, 54, 55, 56]. Newer models like NetMHCpan-4.1 [55] do typically also take into consideration EL data. This data naturally defines a categorization task (presented, not presented).

The second dimension of evolution, concerns the treatment of different MHC alleles. Early models were MHC protein specific. This means that a single model is trained for a particular MHC allele. Recent models typically also consider information about the MHC protein concerned. This is often input via a pseudo sequence (subsequence of the MHC protein's full amino acid sequence). Models able to deal with various MHC alleles are often referred to as 'pan' in the literature (e.g. NetMHC vs NetMHCpan). The next evolution step in this direction was the addition of deconvolution capabilities (see Subsection 2.2.1).

Finally, the third dimension regards the context of the peptide. Most models still

today do not take into account the amino acids surrounding the peptide in its source protein. However, based on the idea to capture the whole MHC-I pathway, which also includes cleavage by proteasomes and proteases that might work better on certain contexts, models like MHCflurry [56] do exactly this.

After this sketch of the evolution of MHC-I immunopeptidome predicting models, we describe two of the most popular ones in more detail. Then we introduce one transformer model - BERTMHC - that has been trained for the prediction of peptide presentation by a related MHC protein - the MHC-II protein. This has been an important inspiration for our work. At the end of the section we give a very brief overview of some other notable approaches taken in the literature for binding affinity and/or presentation prediction.

2.3.1 NetMHCpan

The probably most commonly used model today is NetMHCpan. It has got a very long history which we could trace back to the early 2000s [51] and is currently in its version 4.1 [55]. The 'pan' in its name means that it can handle various different MHC proteins. NetMHCpan 4.1 is an ensemble of 50 single hidden layer feed forward neural networks. To support BA as well as EL data, the outputs of these networks have two heads [55, supplement]. The information about the MHC allele is given to the model in form of a pseudo sequence consisting of only 34 AAs. These were identified by [57] as being particularly close to the presented peptide (closer than 4.0 angstrom from the peptide for a representative set of HLA-A and HLA-B structures). The idea is that those should be particularly relevant for peptide binding.

For deconvolution of multi-allele (MA) data, NetMHCpan takes advantage of the NNAlign_MA [19] framework. First only single-allele (SA) data is used to train a classifier (which takes as input a peptide and a single MHC allele). Then follows the deconvolution step. Each observation that could be caused by multiple MHC alleles, is deconvolved separately. To do so, the classifier trained in the previous step is used to predict the likelihood of each potential peptide:MHC protein combination independently. The MHC allele showing the highest scaled prediction (the scaling essentially results in the z-score of allele's predictions) is chosen and used as the MHC protein responsible for the observation for all purposes until to the next deconvolution step. In case of a negative example, a MHC allele is picked at random. Except for the scaling, this is also roughly the approach we will follow.

The generation of negative examples is not explained in detail. However, [19] say that they use random peptides from the UniProt database. For each peptide length they generate an equal amount of random negatives. This amount is five times the number of observations for the most abundant peptide length in a dataset (they combine several sources) [19]. In total, it seems that for NetMHCpan there are roughly 18 negative examples per positive one in their training set [55, Supplement Table 1].

2.3.2 MHCflurry

Next we introduce MHCflurry 2.0 [56]. It explicitly models the process of MHC binding separately from the others (e.g. proteasomal cleavage). This results in a natural integration of BA and EL data. MHCflurry 2.0 consists of three sub-models. First, MHCflurry BA models the process of the peptide binding to a MHC protein. Second, MHCflurry AP is supposed to model the remaining antigen processing steps, like proteasomal cleavage and TAP transportation (see Subsection 2.1.2). Finally, MHCflurry PS combines the output of those two models to predict peptide presentation. We will quickly introduce those three models in more detail here.

First, MHCflurry BA is trained to predict the binding between peptides of up to 15-mer length and a MHC protein. As does NetMHCpan, MHCflurry BA receives the MHC-I identity in the form of a pseudo sequence. Only they extend it by 3 additional positions to separate some otherwise indistinguishable alleles. MHCflurry BA is trained on BA and mono-allelic EL data. For its training they generate random decoys from the same distribution as the EL hits - not from the distribution in the proteome [56]. They trained 140 different models and selected ten of those to use in the final ensemble. These ten would have varying architecture, with the input passing through up to three dense neural network layers, each followed by dropout. Some of these layers might also use L1 penalty and skip connections [56].

Since the activity of proteasomes and peptidases can be affected by the residues surrounding the peptide, the second model - MHCflurry AP - takes the 15 residues before the peptide's N-terminus and the 15 residues after the peptide's C-terminus as an additional input to the peptide. Doing this, [56] found a small but consistent positive improvement in model performance. In contrast to MHCflurry BA, MHCflurry AP does not receive any information about the MHC protein. The training set for MHCflurry AP is generated based on mono allelic EL data enriched by 100 decoys per observation. The decoys have the same lengths and are from the same proteins as the

hits. Then they let MHCflurry BA predict the binding affinity of the peptides in the enriched dataset. The 2% of peptides (hits and decoys) predicted to bind the strongest, are selected for the training set of MHCflurry AP. 44% of these were hits. As these were both observed in the EL experiment as well as predicted to bind by MHCflurry BA, also the remaining antigen presentation steps need to have been successful. Hence, MHCflurry AP should predict a positive result. On the other hand, the remaining 56% of decoys, were predicted by MHCflurry BA to bind to the MHC allele, but were not observed in the experiment. Assuming that the mass spectrometry did not miss those, there must have been a problem with a different antigen presentation process. Therefore, MHCflurry AP should predict a negative result. Similarly to MHCflurry BA, they train 512 different neural networks and select the best 8 of them to combine in an ensemble. These are four layer neural networks. The first layer is a convolutional one producing as many output positions as input positions and allows for the integration of contextual information. Then follow two parallel subnetworks. Each of these has two layers and is applied to every position independently. They are supposed to predict potential C- and N-terminal cut sites at each position. Their outputs are then pooled in various ways and fed through a single fully connected layer to produce the final prediction. [56]

Finally, the third sub-model is MHCflurry PS. It receives the MHCflurry BA's binding affinity prediction as well as MHCflurry AP's antigen processing prediction and outputs a presentation score. For multi-allele examples, the strongest BA prediction by MHCflurry BA for any of the alleles is used. MHCflurry, therefore, also implicitly relies on the results from MIL for deconvolution (see Subsection 2.2.1). While MHCflurry BA and AP were both trained on mono-allelic data, MHCflurry PS is trained on multi-allelic EL data. To generate negative examples, they sample two random decoys per hit from its corresponding protein.

[56] benchmarked their performance on held-out MS data against NetMHCpan 4.0 and MixMHCpred 2.0.2. They find that their model has substantially better performance (with regards to their chosen metric - positive predictive value). We will also use this benchmark dataset for comparing our model to those three SOTA models.

2.3.3 BERTMHC

The network constructed by [29] has a very similar structure to the one we trained for this thesis. In contrast to us, however, their aim is to predict peptide presentation via a

different pathway by the MHC-II protein. This pathway is used by professional antigen presenting cells to present longer peptides to CD+4 (mostly helper) T-cells. Their network takes as input the MHC-II pseudo sequence and the peptide concatenated. This is in contrast to our network, that also takes into account the peptide's context. The input is then tokenized (one embedding per amino acid) and fed through a pretrained BERT model. This is the TAPE transformer model (see Subsection 2.2.4) also used by us. Its outputs are then pooled with average pooling. The fixed dimensional output of this pooling operation is then fed through a multi layer perceptron (MLP) consisting of two layers with a hidden dimension of 512. Its output layer has two neurons delivering two values - binding affinity and surface presentation.

2.3.4 Notable other approaches

Several other approaches have been proposed in the literature for the prediction of peptide:MHC protein binding and presentation.

[58] propose ACME (Attention-based Convolutional neural networks for MHC Epitope binding prediction) - a pan-specific deep CNN with an attention mechanism that is computed in parallel to the CNN and then weights the CNN's output before it is input into the model's head. ACME is restricted to predicting MHC-I peptide binding affinities for single-allele data. Their primary aim is to generalize binding prediction to before unseen HLA alleles and to use the attention mechanism to explain the underlying rules of peptide:MHC binding.

Another CNN based model is PUFFIN (Prediction of Uncertainty in MHC-peptide aFFInity using residual networks) [59]. It receives a MHC allele and a peptide as input and produces the probabilistic binding affinity distribution as output. Based on this they introduce a 'binding likelihood'. This is defined as the probability of binding at a specified affinity threshold. [59] find that using this with an affinity threshold of 500nM for prioritizing peptides leads to improved precision over using binding affinity.

MHCAttnNet [60] has another interesting architecture. It uses two encoder stacks - one for the peptide and one for the MHC allele. Each begins with a bidirectional long short-term memory (bi-LSTM) network, which allows them to deal with variable-length inputs. Then this is fed through a fully connected network to produce the binding probability score. MHCAttnNet predicts MHC-I and MHC-II binding.

Chapter 3

Method

In this chapter, we describe the data used by us, the model architecture trained as well as the methodology followed by us to produce the charts for the model interpretation.

3.1 Data

We combined data from two datasources. The first one was provided by Georges Bedran a PhD student at the University of Gdansk. It consists of a collection of peptides from EL assays mapped to the GRCh38 *Homo sapiens* reference genome and proteins within the Ensembl v94 database. The ultimate source of the data were studies included in the PRoteomics IDEntifications Database (PRIDE) [61]. We applied some manual preprocessing and removed samples without linked HLA proteins or where only the HLA supertype was available. We also removed "cryptic" matches (neo-antigens). The second data-source is the HLA Ligand Atlas [62]. This includes tissue and HLA allele specific ligands from EL experiments. In contrast to the first data-source, the HLA Ligand Atlas maps peptides to the Uniprot proteome.

We only consider peptides of lengths between 7 and 15 amino acids (inclusive). This compares to peptide length restrictions of 8 to 14 AA for NetMHCpan 4.1 [55] and 8 to 15 AA for MHCflurry [56].

There are several distinct entities present in our problem. A sample represents the experiment carried out on a particular cell-line/individual. It is, therefore, linked with up to 6 different HLA alleles (see Subsection 2.1.2) and the observations (hits) of peptides during the experiment. Each peptide was mapped to proteins in the human genome. In case all of those mappings had the same 15-mer context (N-flank and C-flank), these flanks were used. Otherwise, they were left out (not given to the model

as an input). For each observation we generated 99 artificial decoys (see Subsection 3.1.1). Examples are either observations/hits or decoys and define a single training instance. However, this means that the same peptide:MHC allele combination can be shown several times to the model during the same epoch. This will effectively lead to an overweighting of these for the updating of the model parameters. From the perspective, that those examples were also observed during multiple independent experiments this seems justified. An alternative way would have been to collapse all of those examples into one. We suggest further research into this topic.

Using the representation outlined above, we get the following Table 3.1 of observations, unique peptides and MHC alleles occurring in our combined dataset.

DATASOURCE	SAMPLES	OBSERVATIONS		UNIQUE PEPTIDES	MHC ALLELES
		SA	MA		
UNIVERSITY OF GDANSK	271	293,334	1,256,916	390,959	104
LIGAND ATLAS	198	0	409,486	90,422	51
TOTAL	469	293,334	1,666,402	429,339	109

Table 3.1: Overview of total dataset

3.1.1 Decoy generation

Negative example (decoy) generation is very important, particularly due to the imbalanced nature of the dataset (see Subsection 2.2.2). Our way of doing this is inspired by how the MHCflurry benchmark dataset was generated [56]. For each hit we sample 99 decoys. To match the observations' length distribution, the decoy peptides have the same length as their associated hits. To generate a decoy peptide we would randomly select any protein from the set of proteins observed in a sample. These are all proteins that any of the sample's peptides was mapped to. Then we randomly select a position within the protein as the start of the decoy peptide. This step should be improved in a next version of our model, as it might skew the decoys towards shorter proteins. Also, implicitly we (and other SOTA models) assume that all proteins that a sample's peptides got mapped to must also be expressed in the sample. Based on this we take the absence of a peptide's observation as evidence for it actually not being presented. So, we assume that the data comes from high quality EL experiments and that peptides are only matched to actually expressed proteins. We think that these assumptions need more scrutiny by the community and that in a next step weighting examples by our

confidence that they hold, might be beneficial.

The benchmark dataset of MHCflurry is used to test MHCflurry's performance. As seen in Subsection 2.3.2, they generated 99 decoys from the same protein as the observation for the training of MHCflurry AP but only 2 decoys per observation for the training of MHCflurry PS. Ninety-nine decoys per hit fits better to the true distribution (see Subsection 2.1.2) but training on an imbalanced dataset might be an issue (see Subsection 2.2.3). Subsection 2.3.1 mentions that NetMHCpan 4.1 used roughly 18 decoys per hit. So, we consider using 19 or 99 decoys per hit in our hyper-parameter search (Section 4.1). With regards to whether those should be chosen from the same protein as the observation or any protein observed within the sample, the implicit assumption for both is that absence of evidence is taken as evidence of absence. However, practically, there could be gene products that are too short to accommodate the 99 decoys per hit. So we decided to sample from all proteins observed during an experiment.

3.1.2 Data splits

Splitting the data into a train, test and validation set is not trivial, as:

1. We would like to assess generalization along 2 dimensions - primarily to unseen MHC alleles and secondarily to unseen proteins
2. Each observation can be associated with up to six MHC alleles from which at least one is responsible for the presentation
3. There are many homologues (see Subsection 2.1.1) in the human genome. Ideally, a group of homologues would not span different splits

MHC allele dimension: As each individual normally has at least one working copy of each HLA gene (A/B/C), it is not possible to hold out a full gene. So, we hold out observations on the HLA group level (e.g. HLA-A*01, ...). First we count how many observations belong to each HLA group (from a donor/cell with at least one HLA gene being in the group). The result can be found in Appendix Table 6.1. We find that the groups are highly unequally represented in the dataset. To find at least 5 groups for each set, we perform the following steps until a satisfactory split is found.

First, we randomly assign HLA groups (and the linked examples) to the validation set until its target number of examples is reached - overriding the standard training set assignment. Then we randomly assign allele groups to the test set until its target number of examples is reached - overriding any earlier assignment. If we assigned too

Figure 3.1: Illustration of splitting procedure

few or too many we repeat. The final split can be found in Appendix Table 6.1. Figure 3.1 illustrates this process which ensures that no validation or test MHC group enters the training phase and no test MHC group enters the validation or training phases.

Protein dimension: Following this, we split off another validation and test set from the remaining training set. This second split is, however, based on an observation's mapped proteins not its linked MHC alleles. Due to homology we cannot just split the dataset based on the protein names. There are many different approaches to deal with this. For example [45] have decided to go for sequence identity and ensure that no sequence in the test and train set have more than 25% sequence identity. We feel that this approach might be the right one for the particular applications [45] had in mind - assessing for example overall protein structure. However, in our case we are more concerned with small local features - peptides. So we do not want to have any similarity of subsequences above 7 amino acids long. As checking this would be prohibitively computationally expensive, we chose another approach. We use the python networkx package, which allows to build and explore graph structures. With it we use the Ensembl BioMart paralogue table [63] to link related genes as well as proteins to their respective gene. We then randomly assign disconnected sub-graphs to the various splits until our target values are reached.

Following the above, our dataset is split up into 5 sets. One training set, two validation sets and two test sets. Combining the two validation sets would be problematic. Let us assume an observation is part of the MHC validation set. As the protein split is done on the remaining observations after the MHC split, it cannot be part of the protein test or validation splits. However, it could still come from a protein, that is on the list of proteins defining the protein test set. Therefore, it is cleaner to keep these two perspectives independent.

After the split, we obtain the partition of our data in Table 3.2. Since generalization in the MHC dimension is more important (as the peptides in our dataset already spread across the whole human genome), we assigned it a bigger proportion.

SPLIT	TRAIN	VAL-PROT	TEST-PROT	VAL-MHC	TEST-MHC
TOTAL OBSERVATIONS	1,408K (71.8%)	70K (3.6%)	71K (3.6%)	204K (10.4%)	206K (10.5%)
SINGLE ALLELE	206K (10.5%)	10K (0.5%)	11K (0.6%)	24K (1.2%)	43K (2.2%)
MULTIPLE ALLELE	1,202K (61.3%)	60K (3.1%)	60K (3.1%)	181K (9.2%)	164K (8.3%)

Table 3.2: Observations per dataset split

3.2 Model Architecture

One of the major ML developments in the past few years has been transfer learning. This has been particularly successful in the domain of natural language processing where big networks like GPT-2 and T-5 have been trained on multiple GPUs for weeks and extensive text corpora and are now available in model zoos to download for everybody and fine-tune quickly to their particular task. In this spirit, we adapt and use the pre-trained TAPE transformer (see Subsection 2.2.4) as backbone for our model.

3.2.1 Tokenization and Embedding

The primary input into our model is the peptide's AA sequence. If a peptide could be mapped to multiple proteins, then we check if all contexts are the same. If so, we provide the model with this context as input, otherwise not (as well as in the input to the decoy examples). The context consists of the up to 15 AAs that occur to the 'left' of the peptide in its source protein's AA sequence (the N-flank) as well as the up to 15 AAs that occur to the 'right' of it (the C-flank). The 15 AAs length was chosen to be compatible with MHCflurry and be able to use their benchmarking data. Finally, the model also receives the MHC pseudo sequence as defined by NetMHCpan (see Subsection 2.3.1) as input.

The input sequence starts with a <cls> token. Then come the N-flank, peptide, C-flank and MHC allele. In between those components and at the end of the sequence there are <sep> tokens. The <sep> are used in BERT to separate sentences and the <cls> is meant to be used for classification [35].

All of these components are represented by letters symbolizing amino acids. However, BERT requires embedding vectors as input. At first a tokenizer splits the input

sequence up into tokens and maps each to several integers - in our case token ID, position number and token type ID. Then, embedding layers assign a different learnt embedding vector to each token ID, position number and token type ID. The overall embedding is the sum of these vectors (in our case a 768 dimensional vector per token). Crucially, this embedding does not take into account its context. So every histidine at peptide position 3 will always result in the same embedding - independent of the other amino acids in the input sequence. In addition, there is an input mask with which it is possible to hide tokens from the model (set the mask to zero).

To take advantage of the pretraining, we need to use the same token IDs as used by the TAPE transformer. This represents each amino acid by a separate number. The tokenizer distinguishes 20 standard AAs, the two non-standard AAs (pyrrolysine, selenocystein) and unknown AAs [45]. There exist alternative tokenizations. For example, several amino acids could be combined to form a single token.

Off the shelf, the TAPE transformer supports only a single token type id. To make it easier for the model to distinguish between the various input parts, we use a novel representation of the input and extended the TAPE model's token type embedding matrix to four different token types - one each for the N-flank, peptide, C-flank and MHC protein. Their values are initialized with the TAPE standard token type embedding values and will then diverge during training.

With regards to the position numbering, the N-flank begins with number one being the residue closest to the peptide (numbering from C- to N-terminus). This is done, so that the peptide's proximal amino acid is always at position 1 - even if the N-flank is shorter than 15 AAs long. In contrast, the peptide and C-flank position numbering is done from N- to C-terminus. For the MHC pseudo sequence, we use the AA's position in the full MHC amino acid sequence. This is done, so that in future, the model can be easily fine tuned to support full MHC sequences. An example can be found in the Appendix.

3.2.2 Encoder

The resulting vectors from the embedding step are fed through the TAPE encoder consisting of 12 self attention layers with 12 heads each. The output is again a vector of dimension 768 per position. This, however, is now a contextual embedding for each position. This means, that other elements of the input will have influenced this vector - so it will change if other positions in the input were to change.

3.2.3 Pooling

The encoder produces as many vectors as input tokens. These cannot be fed directly into a MLP which expects a fixed dimensional input. Pooling is necessary. [42] showed that the optimal pooling operation is task dependent. BERTMHC (Subsection 2.3.3) use average pooling. However, we felt that the meaning of the peptide, context and MHC sequence are quite different and so we considered three options: to use averaging, to use an attention layer and to use the classification token's vector. We compare those as part of the hyper-parameter search (see Section 4.1).

3.2.4 Head

The structure of our model's head is similar to the one used by BERTMHC [29]. It is also a MLP consisting of two fully connected layers with a hidden dimension of 512. In contrast to BERTMHC we only have a single output neuron, with sigmoid activation. This is because we only estimate a presentation score and not binding affinity.

3.3 Training

The training procedure of our model is inspired by the NNAlign_MA framework and general results of MIL. In the first training epoch we only use SA data. Then follows a deconvolution phase. During this, we deconvolve each MA observation by at first predicting the presentation score for each potentially responsible allele and then selecting the one with the highest score as the relevant allele. This means, that for forward-propagation and backward-propagation, the MA example is treated as if coming from the relevant allele until the next deconvolution phase, which happens after each epoch. The relevant MHC allele of the decoys, follows the one of the observation.

For training we use standard binary cross entropy as loss function. The parameter gradients were calculated using standard back-propagation. We trained the full network (including encoder and embedding layers). The parameters were updated using the ADAM optimizer which adjusts each parameter's learning rate over time. We use its standard hyper-parameters from the original paper ($\beta_1 = 0.9, \beta_2 = 0.999$ [64]). Although it is quite robust with regards to hyper-parameter choice, and adjusts the learning rates over time, a sensible choice of initial learning rate is still necessary [65, page 306]. We choose this as part of the hyper-parameter search (see Section 4.1).

3.4 Evaluation

To shed light on what our model has learnt and to assess its quality, we performed:
- Evaluation on the test sets
- Comparison of our model to MHCflurry and NetMHCpan
- Model interpretation by LIME analysis of peptide, flanks and pseudo sequence feature importances, using motifs and by SHAP analysis of peptide AAs contributions

3.4.1 Evaluation on the test set

In Section 4.2 we evaluate our final model for AP, ROC-AUC and accuracy (see Subsection 2.2.3) on the MHC allele- and protein test sets (see Subsection 3.1.2).

3.4.2 Comparison to MHCflurry and NetMHCpan

The MULTIALLELIC benchmark dataset of MHCflurry consists of 9,158,100 examples. Each has a peptide, N-flank (15 AA), C-flank (15 AA), up to six HLA alleles as well as the predictions of NetMHCpan, MixMHCpred and MHCflurry for the example. [56] generated this dataset from 11 studies using EL data. For each hit they randomly generated 99 decoys. A more detailed description and the full dataset is available in [56, Supplement Data S1]. We run two evaluations on this - one on the whole dataset (9,158,100 examples) and one for which we removed examples of peptides that were already part of our training dataset (2,781,898 examples). For these we predict our model's presentation score, calculate performance metrics and plot PR curves for MHCflurry, NetMHCpan and our model (ImmunoBERT).

3.4.3 Interpretation

To reduce complexity, we restrict our interpretation of the model to 9-mer peptides and single-allele data. We used only the test set during interpretation. So, our model has not seen the data before. Therefore, the analysis demonstrates our model's ability to generalize to unseen MHC proteins.
LIME analysis of all features: We first assess the importance of all AA features in the input-sequence. For this, we use the LIME framework, as this is in general faster than SHAP - in particular for many features (in our case 73 amino acid positions). As described in Subsection 2.2.5 we adapted the text version of LIME. We implement the deactivation of a feature by setting the input mask token to zero. We use the standard

cosine distance metric (in binary space) between the original example and the sampled examples. For each test set HLA allele, we selected a random 500 observations and for each of those one decoy (total 1000 examples) to be explained. Each example gets explained by sampling a random 2000 feature combinations.

For visualization, Figures 4.1, 4.6 and 4.9 show in each bar the proportion of examples with a given importance-ranking. If for example the bright red bar (1st) of peptide position 9 showed 0.5, this means that 50% of examples had this as the most important feature.

Sequence Motifs: We want to utilize the very common sequence motif logos [66] to visualize for each test set HLA allele, how often various amino acids occur at presented peptide positions. To do so, we generate 100,000 random 9-mer peptides from the human proteome (as well as their context). Then we predict for each of those 100,000 examples the presentation score for the HLA protein concerned. We select the ones for which our model predicts presentation (presentation score > 0.5) and use them to create the allele dependent logo for our model (using the logomaker [67] package). This analysis is similar to the one carried out by [68], which however use the 2% highest scoring peptides for the model motif. Afterwards, we take the data from our test set and use all of the 9-mer peptides unambiguously presented by the HLA allele to create another logo for the data as well.

The created motif logos have the following form. For each of the nine peptide positions, a stack of AA letters is displayed. The size of each letter is proportional to the AA's frequency at this position. Also, more frequent AAs can be found on top. Each stack is then scaled with the position's information content (IC), resulting in a representation as bits [66]. The lower the position's entropy, the higher the IC and, so, the logo. We do not display positions with IC < 0.5 to avoid distraction. Amino acids with similar chemistry are coloured the same.

SHAP analysis of peptide positions: Finally, we look for additional insights by examining the average contribution of peptide amino acids using SHAP. For each test set HLA allele, we select a random five-hundred 9-mer single allele hits and one decoy per hit to explain from the test set. From these 1000 examples we also sample 250 as background distribution (see Subsection 2.2.5). The nine features in the peptide would result in a maximum of 512 feature subsets. We carry out the Kernel SHAP analysis using a sample of 64 of these (see Subsection 2.2.5). For the whole process we ignore the flanks. In Figures 4.4, 4.8 and 4.11 we plot the average SHAP value for each amino acid at each position.

Chapter 4

Results

This chapter presents the results obtained by applying the methodologies from Chapter 3. First, we justify some of the parameters chosen before benchmarking our model to two alternative SOTA models. Eventually, we use explainability techniques to analyse what our model has learnt about peptide presentation by three test set HLA proteins.

4.1 Hyperparameter search and training

We searched parameters along 3 dimensions: pooling mechanism, decoys per hit and learning rate. We considered using the <cls> tokens embedding, averaging and a classical attention mechanism (see Subsection 3.2.3) as pooling mechanism. Further, we compare using datasets enriched by 19 and 99 decoys per observation. Eventually, we try using 1e-04, 1e-05 and 1e-06 as initial learning rates. For the hyperparameter search, each model was trained on 10% of the SA data for 64,599 steps (one epoch for the 99 decoys per observation datasets and five epochs for the 19 decoys per observation datasets - so both of them have seen the same observations at least once).

We evaluated each model on 10% of the MHC-validation and protein-validation set [1] SA data (using 99 decoys per hit). Table 4.1 shows the result for the hyper-parameter search using a learning rate of 1e-05. The results for a learning rate of 1e-06 can be found in Appendix Table 6.2 and show, that the learning happened too slow. Using a learning rate of 1e-04 would most of the time not result in any detected hits.

Table 4.1 shows various performance metrics (best column values in red) on our two validation sets for the 6 models described above at two points during their training. Different initializations might deliver different results, as mentioned above we

[1] our two validation sets, see Subsection 3.1.2

didn't use the full dataset and the training will not have converged after 64,559 steps yet. However, given our limited computing resources and the time it takes to train transformers, we had to base our decision on these numbers. Dependent on the chosen metric, one or the other pooling mechanism and one or the other hits-to-decoys ratio looks best. In general, the unbalanced dataset (99 decoys per hit) at first (step 12,911) leads to quite poor classifiers in terms of ROC-AUC and AP. However, after a full epoch they are able to make up most of this. In fact, as found by [33], models trained on an unbalanced dataset close to the actual data distribution (99 decoys per observation) show better accuracy. Actually, the models using only 19 decoys per observation have a worse accuracy than just always predicting negative. However, as we have seen in Subsection 2.2.3 accuracy is not informative and reliable when dealing with highly imbalanced data [2]. As we see in Table 4.1 the models performing best on ROC-AUC and AP all were trained using 19 decoys-per-hit. So we will use this. The table also shows that using the classification token as input to the head had the best performance (in red) in 5 cases while the attention mechanism only had the best performance in 3 cases and the averaging in 2 cases. We will, therefore, train our final model using the classification token's output as input to the model's head.

Pooling	Decoys	After x steps	val-mhc			val-protein		
			ROC AUC	AP	Acc°	ROC AUC	AP	Acc°
Cls	19	12911	0.938	0.319	0.986	0.956	0.447	0.987
		64559	0.949	0.394	0.982	0.964	0.552	0.983
	99	12911	0.823	0.062	0.990	0.917	0.232	0.990
		64559	0.945	0.349	0.991	0.960	0.492	0.992
Attn	19	12911	0.936	0.298	0.973	0.957	0.447	0.982
		64559	0.941	0.410	0.983	0.964	0.535	0.982
	99	12911	0.792	0.044	0.990	0.884	0.184	0.990
		64559	0.938	0.315	0.990	0.958	0.496	0.992
Avg	19	12911	0.926	0.252	0.982	0.954	0.417	0.985
		64559	0.949	0.369	0.977	0.960	0.506	0.980
	99	12911	0.689	0.026	0.990	0.835	0.145	0.990
		64559	0.924	0.287	0.990	0.956	0.471	0.992

Table 4.1: Performance comparison for a learning rate of 1e-05 (°... accuracy)

[2] e.g. when the majority to minority class ratio is 999:1 a classifier always predicting the majority class will have 99.9% accuracy

Table 4.2 shows how various metrics develop during its training. Due to time reasons and little improvement we stopped after epoch 5 and use this as our final model.

AFTER		VAL-MHC			VAL-PROTEIN		
EPOCHS	STEPS	AP	ROC AUC	ACC°	AP	ROC AUC	ACC°
1	128494	0.571	0.966	0.989	0.667	0.985	0.988
2	1008417	0.646	0.976	0.992	0.762	0.993	0.993
3	1888340	0.671	0.978	0.993	0.767	0.993	0.994
4	2768263	0.673	0.978	0.992	0.768	0.993	0.993
5	3648186	0.683	0.979	0.992	0.765	0.993	0.994

Table 4.2: Performance comparison during training (°... accuracy)

4.2 Evaluation on the test set

To make sure we didn't overfit the hyperparameters to the validation set, Table 4.3 shows the test set performance of our selected final model. We see that the values are not very different from the ones for the validation set in Table 4.2.

AFTER		TEST-MHC			TEST-PROTEIN		
EPOCHS	STEPS	AP	ROC AUC	ACC°	AP	ROC AUC	ACC°
5	3648186	0.704	0.981	0.993	0.755	0.992	0.993

Table 4.3: Performance on the test set (°... accuracy)

4.3 Benchmarking

Unluckily, there are no generally agreed upon standard benchmarking datasets available in our domain. However, [56] have curated a benchmark dataset (see Subsection 3.4.2). We ran the below analysis on the full dataset as well as on one in which we exclude peptides from our training set. We calculated the AP as well as the ROC-AUC for MHCflurry, NetMHCpan and ImmunoBERT. For the full set we also plotted the PR-curves for them (Table 4.4). The models were trained on different datasets. So any judgement about the advantageousness of the architectures is not valid. However, the comparison is useful to compare the practical predictive power of the models.

For both datasets, our model shows an AP in between MHCflurry and NetMHCpan. In particular, it does well for thresholds corresponding to intermediate recall levels. However, it achieves less ROC-AUC than the others, possibly caused by the sharp drop in performance for higher recall values. The performance on the reduced set is far worse for all models. So, also the other two models might have already seen similar peptides as were removed during their training. As we explicitly only removed ours, this skews the reduced dataset against our model.

We decided to generate decoys once and show the model the same decoys in each epoch. This was done to ensure reproducibility of results. In contrast, NetMHCpan and MHCflurry resample decoys each epoch [56]. In hindsight, this might be a better design choice and might have led to later convergence during training of our model.

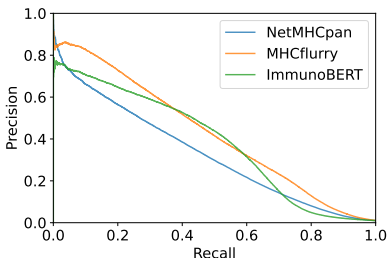
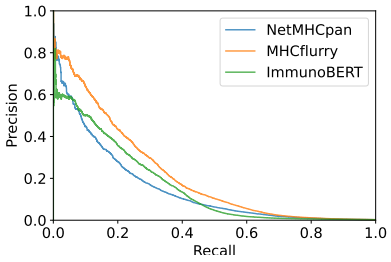
	Model / Metric	AP	ROC AUC	PR-Curve
Full Dataset	NetMHCpan	0.327	0.916	
	MHCflurry	0.427	0.938	
	ImmunoBERT	0.383	0.893	
Reduced Dataset	NetMHCpan	0.151	0.873	
	MHCflurry	0.215	0.890	
	ImmunoBERT	0.163	0.831	

Table 4.4: Performance comparison on benchmark dataset

4.4 Interpretation

Next we use LIME, motifs and SHAP as described in Subsection 3.4.3 to interpret our model's predictions for three selected test set HLA proteins. We chose them to cover different HLA genes and degrees of perceived matching between their data and model motifs. As there are too many test HLA alleles to discuss within the boundaries of this thesis, we put the figures for the remaining test HLA alleles into the Appendix. To improve readability all SHAP values were multiplied by 100.

4.4.1 HLA-A*33:01

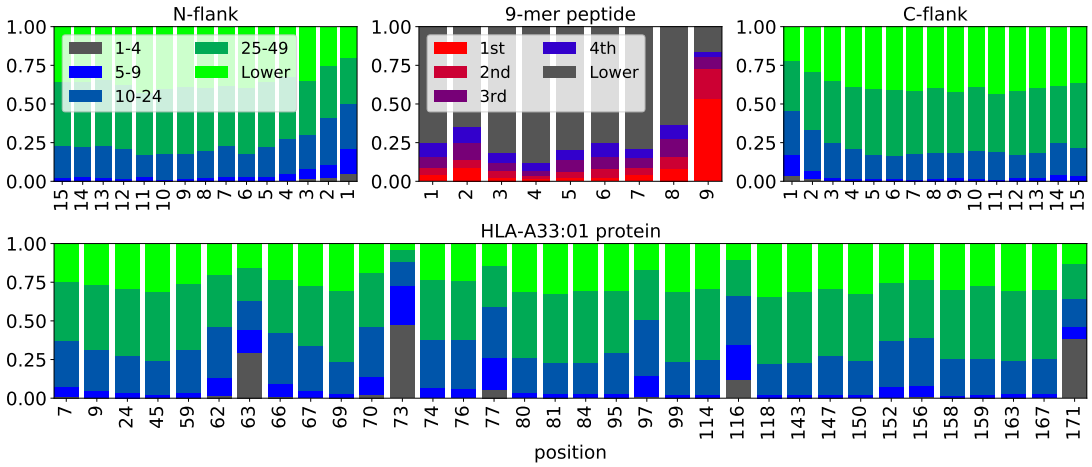


Figure 4.1: LIME feature importance rank distribution for HLA-A*33:01

Figure 4.1 shows for each input position the proportion of examples in which this position had a certain importance ranking (see Subsection 3.4.3). The peptide's AAs tend to be highly ranked. Its position 9 is ranked first in roughly half of the examples. The MHC pseudo sequence positions display a high variance in their ranking distributions. Some positions tend to be particularly highly ranked - like 63, 73, 116 and 171.

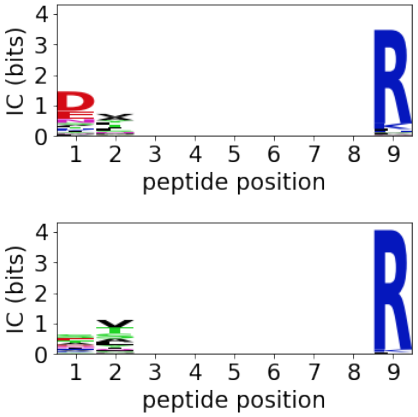


Figure 4.2: Data(top) & model motif

MHC positions 171 and 116 are located in the MHC's A and F pockets [24] which both are supposed to house a peptide terminus [21]. This might explain what we see in the motifs of Figure 4.2 [3]. A peptide C-terminus arginine (R) is very common and the N-terminus is enriched as well. In contrast, the significance of MHC positions 73 (in pocket C) and 63 (in pocket B) is unclear. There might be confounding with other positions or some biological reason for this. For example pocket B could house peptide position 2 - also explaining the slightly elevated importance of this in Figure 4.1. This is just speculation and biological experiments are needed to confirm this.

Finally, the flanks seem the least important in Figure 4.1. [56] also found that including them only results in a small but consistent improvement. Within the flanks, quite sensibly our model attributes most importance to peptide proximal positions.

[3] motifs show the frequency of AAs at positions in observed peptides (hits), Subsection 3.4.3

The data and model motifs in Figure 4.2 are not exactly the same. This might be because they were generated from different background distributions [4]. We argue below that also the SHAP values are context, and so background distribution, dependent.

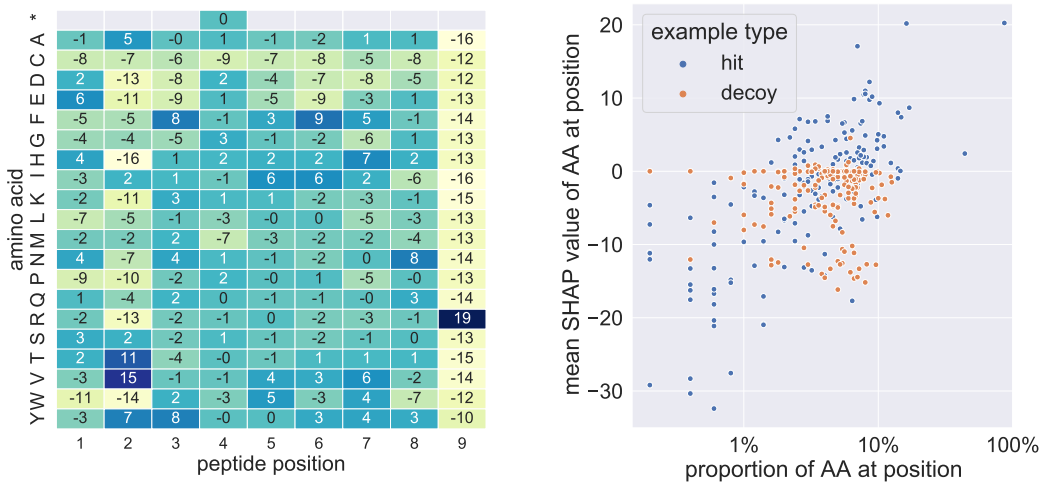


Figure 4.4: Mean SHAP values (x 100) for HLA-A3301examples

With regards to SHAP values, Figure 4.4 left side shows the mean SHAP value of each AA at each peptide position [5]. We see the strong positive mean contribution of R at peptide position 9, making its high frequency in Figures 4.2 plausible. It also shows high values for V and T at position 2 which are also enriched in the model motif.

Intuitively, higher mean SHAP values should be linked to more occurrences of the AA at a position. Figure 4.4 right side depicts this relationship. For hits it seems to be positive - not so for decoys. To investigate this further, Figure 4.5 shows for each AA the difference between its mean SHAP value at position 8 in hits (dot) and in decoys (end of arrow). Especially the values for Asparagine N and Cysteine C change a lot. This demonstrates the context's importance for a feature's SHAP value. So, high average SHAP values need not lead to higher occurrences under all background distributions.

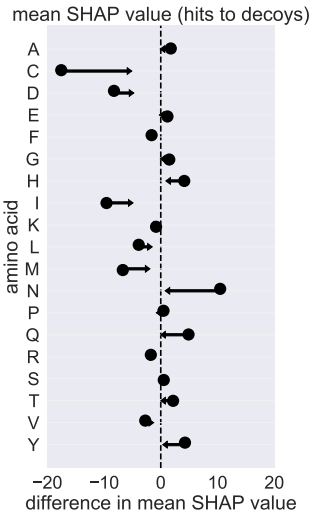


Figure 4.5: position 8: change in SHAP values

[4] While the data motif is based only on the peptides that were expressed in the samples of the test HLA allele, the model motifs are based on the whole human genome - Subsection 3.4.3

[5] see Subsection 3.4.3

4.4.2 HLA-B*54:01

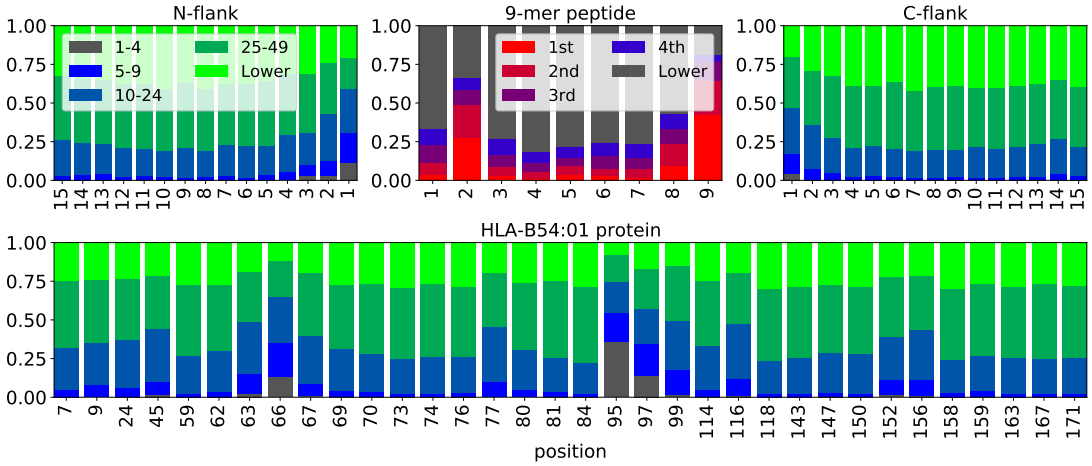


Figure 4.6: LIME feature importance rank distribution for HLA-B*54:01

We see a similar picture in Figure 4.6 as in Figure 4.1. The peptide is the most important element followed by the MHC allele and the flanks are the least important. However, this time two positions in the peptide are of especially great importance - 2 and 9. This corresponds to what we see in the motifs in Figure 4.7 and in fact the SHAP values in Figure 4.8.

With regards to the MHC protein, its most important position is 95. This is located in the F pocket [24]. This makes sense given the high SHAP values of peptide position 9. Its second most important position is 66 in pocket B. Given the enrichment of peptide position 2 by P and A we speculate that pocket B houses those. Indeed, [69] claim that this is the case. Also the other MHC alleles in our test set with a high specificity at peptide position 2 in the data motif (HLA-B*37:01, HLA-B*58:01 and HLA-B*58:02) show slightly elevated importances for pocket B positions (e.g. 66, 67) - see Appendix.

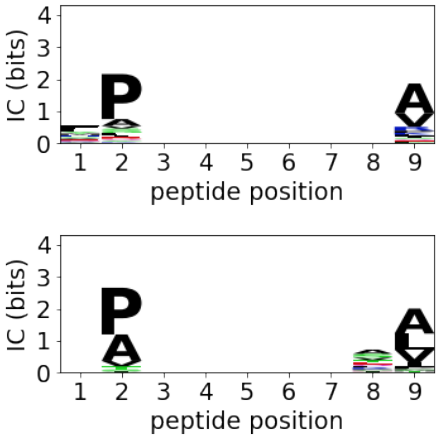


Figure 4.7: Data (top) & model motif

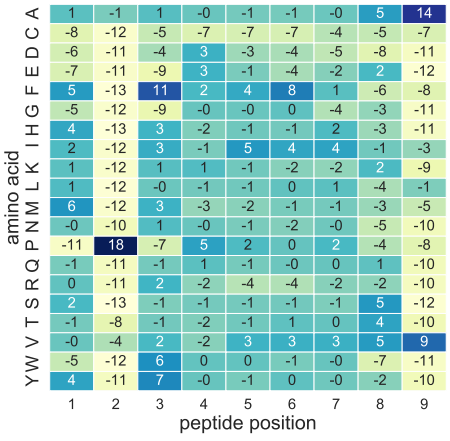


Figure 4.8: Mean SHAP values

4.4.3 HLA-C*01:02

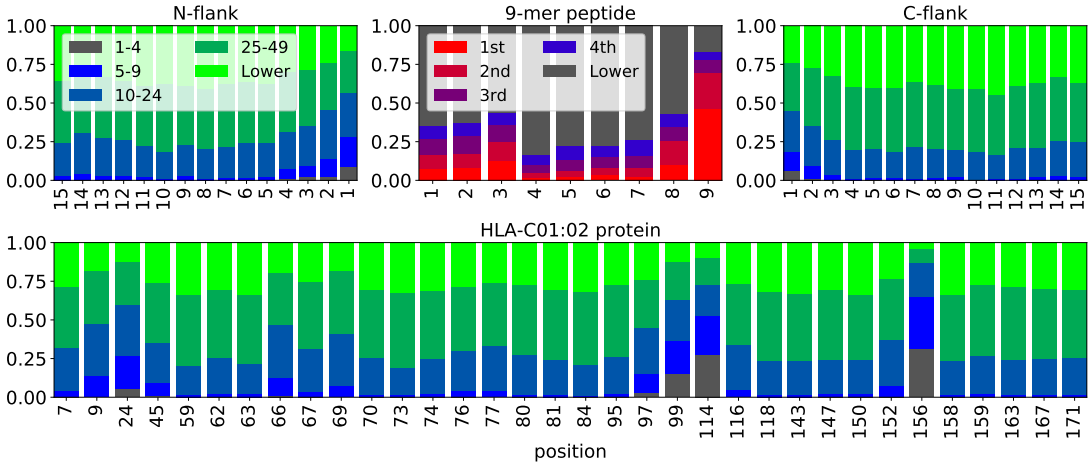


Figure 4.9: LIME feature importance rank distribution for HLA-C*01:02

Figure 4.9 shows a similar picture as Figures 4.1 and 4.6. However, the data and model motifs in Figure 4.10 are quite different. First, the model motif misses the enrichment in particular by S and A at data motif position 8. Interestingly, those AAs have relatively high mean SHAP values in Figure 4.11. Also Figure 4.9 shows elevated importance of this position. So, this might just be caused by different background distributions between the sets used to generate the model (based on the whole human genome) and the data motif (based on test set examples). Next, the model motif shows an enrichment at position 4. This is primarily caused by E, which also has a high mean SHAP value. Given the low IC of this, the difference in motifs could be caused by chance. At peptide positions 2 and 3 the two motifs look very similar - except for ordering. With regards to MHC positions 114 and 156 are especially important in Figure 4.9. These lie close to each other in pocket E [24] and might house one of the enriched peptide positions requiring further research.

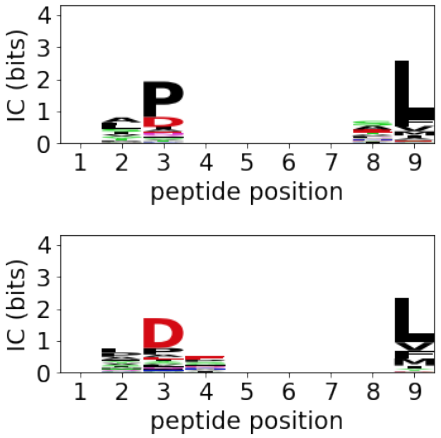


Figure 4.10: Data(top) & model motif

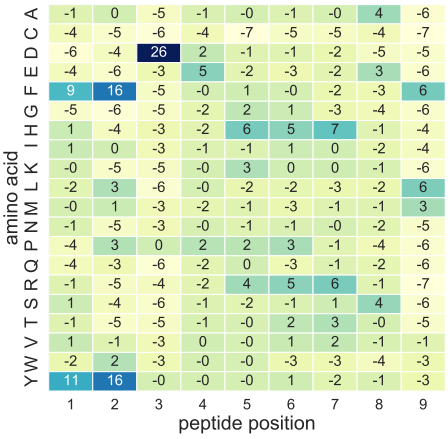


Figure 4.11: Mean SHAP values

Chapter 5

Conclusion and future work

We have seen in Section 4.3 that our model based on the BERT architecture was able to compete with current SOTA models that have been developed over many years. This highlights BERT's high potential in this area. Then, we compared various pooling techniques and decoy to hit ratios in Section 4.1 and found that the best choice will depend on the evaluation metric. As a novel approach in Section 4.4 we used the explainability techniques LIME and SHAP to find that our model learnt biologically sensible importance rankings and feature contributions. Namely, it placed particularly high importance on AAs near the N- and C-termini of the peptide and varying MHC positions in the A, B and F pockets. In contrast, the flanks showed less importance, which explains why [56] found that including them only results in a small but consistent model improvement. The motifs we found using our model followed broadly those observed in the data. As these analysis were all carried out on held out MHC proteins, it also demonstrates the generalization ability of our model.

Given the high degree of imbalance in the data, we think that more advanced methods are required for decoy generation. Generative adversarial networks might be a way to achieve this. An alternative could also be to utilize the ability of transformer models to assess all positions of a whole sequence at once. As input the model would not only receive a single peptide but a longer sequence of amino acids. It could then be trained to identify the presented sub sequences. This would make the artificial generation of decoys and in particular the search for a good ratio between decoys and hits unnecessary.

To summarize, this thesis has highlighted the high potential of transformer models for peptide presentation prediction and as a novelty applied advanced explanation techniques to this task, with which we could demonstrate biological relationships.

Chapter 6

Appendix

6.1 MHC split

HLA-A			HLA-B			HLA-C		
GROUP	S°	HITS	GROUP	S°	HITS	GROUP	S°	HITS
HLA-A01	99	359,049	HLA-B07	66	378,396	HLA-C01 **	10	36,665
HLA-A02	156	505,417	HLA-B08 *	37	128,339	HLA-C02	73	322,819
HLA-A03	92	644,773	HLA-B13	18	42,331	HLA-C03	96	392,462
HLA-A11	80	200,286	HLA-B14	47	219,850	HLA-C04	114	566,983
HLA-A23	25	73,141	HLA-B15	75	228,832	HLA-C05	37	416,633
HLA-A24	60	299,863	HLA-B18	25	177,056	HLA-C06	75	199,060
HLA-A25	3	14,163	HLA-B27	54	285,520	HLA-C07	173	614,980
HLA-A26	9	16,779	HLA-B35	108	572,469	HLA-C08	48	223,603
HLA-A29	24	244,649	HLA-B37 **	30	74,723	HLA-C12	9	77,625
HLA-A30	16	22,321	HLA-B38	5	45,822	HLA-C14	7	34,824
HLA-A31	18	199,800	HLA-B39 *	6	60,623	HLA-C15 **	5	17,682
HLA-A32	39	272,779	HLA-B40	43	156,648	HLA-C16	25	58,074
HLA-A33 **	2	8,015	HLA-B41	2	12,971	HLA-C17 **	4	18,124
HLA-A34 *	2	7,749	HLA-B42 *	1	4,077			
HLA-A36 **	1	3,960	HLA-B44	83	398,328			
HLA-A66 *	1	2,532	HLA-B45	17	30,927			
HLA-A68	75	246,119	HLA-B46 **	1	1,203			
HLA-A69	22	50,198	HLA-B47 *	6	3,720			
HLA-A74 **	1	3,543	HLA-B49	30	78,815			
			HLA-B51 *	9	29,332			
			HLA-B52 *	1	1,585			
			HLA-B53 *	1	2,883			
			HLA-B54 **	1	1,622			
			HLA-B55	7	29,812			
			HLA-B56 *	1	1,780			
			HLA-B57	8	24,397			
			HLA-B58 **	24	40,795			

Table 6.1: MHC groups in the datasets (°... samples, *... validation set, **... test set)

6.2 Tokenization example

Peptide: ’LWDSAVSHF’
N-flank: ’SVSHKFLSLPMLVRQ’
C-flank: ’FSLPFKKSLLAALIL’
MHC pseudo sequence: ’YYAGYREKYRQTDVNKLYLRYDSYTWAEWAYEWY’

This will result in the following string:

‘<cls>SVSHKFLSLPMLVRQ<sep>LWDSAVSHF<sep>FSLPFKKSLL AALIL<sep>YYAGYREKYRQTDVNKLYLRYDSYTWAEWAYEWY<sep>’

This string will be tokenized into the following input token ids: 2, 22, 25, 22, 12, 14, 10, 15, 22, 15, 19, 16, 15, 25, 21, 20, 3, 15, 26, 8, 22, 5, 25, 22, 12, 10, 3, 10, 22, 15, 19, 10, 14, 14, 22, 15, 15, 5, 5, 15, 13, 15, 3, 28, 28, 5, 11, 28, 21, 9, 14, 28, 21, 20, 23, 8, 25, 17, 14, 15, 28, 15, 21, 28, 8, 22, 28, 23, 26, 5, 9, 26, 5, 28, 9, 26, 28, 3

These input token ids will be linked to the following token type ids: 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

The corresponding position ids are: 0, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 0, 7, 9, 24, 45, 59, 62, 63, 66, 67, 69, 70, 73, 74, 76, 77, 80, 81, 84, 95, 97, 99, 114, 116, 118, 143, 147, 150, 152, 156, 158, 159, 163, 167, 171, 0

If all features should be considered, then every input mask position will be 1: 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

6.3 Hyperparameter search

Pooling	Decoys	after x steps	val-mhc			val-protein		
			ROC AUC	AP	Acc°	ROC AUC	AP	Acc°
Cls	19	12911	0.660	0.020	0.990	0.771	0.061	0.990
		64559	0.902	0.209	0.974	0.943	0.356	0.975
	99	12911	0.584	0.013	0.990	0.689	0.033	0.990
		64559	0.721	0.027	0.990	0.835	0.119	0.990
Attn	19	12911	0.649	0.018	0.990	0.774	0.073	0.990
		64559	0.902	0.210	0.975	0.937	0.349	0.975
	99	12911	0.594	0.014	0.990	0.694	0.034	0.990
		64559	0.731	0.029	0.990	0.829	0.132	0.990
Avg	19	12911	0.657	0.020	0.990	0.776	0.081	0.990
		64559	0.892	0.202	0.969	0.937	0.349	0.970
	99	12911	0.601	0.016	0.990	0.700	0.037	0.990
		64559	0.733	0.031	0.990	0.840	0.134	0.990

Table 6.2: Performance comparison for a learning rate of 1e-06 (°... accuracy)

6.4 Interpretation

On the following pages, the charts for the remaining HLA alleles that were note selected for detailed discussion in Section 4.4 can be found. The methodology for constructing the charts was explained there and in Subsection 3.4.3.

6.4.1 HLA-A*33:03

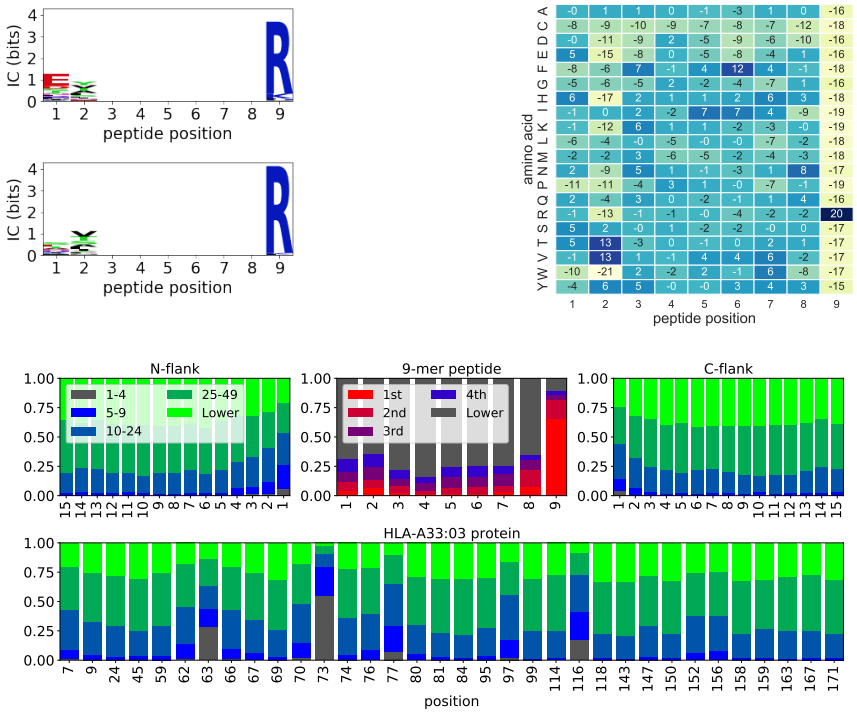


Figure 6.1: Motifs (top left), mean SHAP values (top right) and LIME feature ranks

6.4.2 HLA-A*36:01

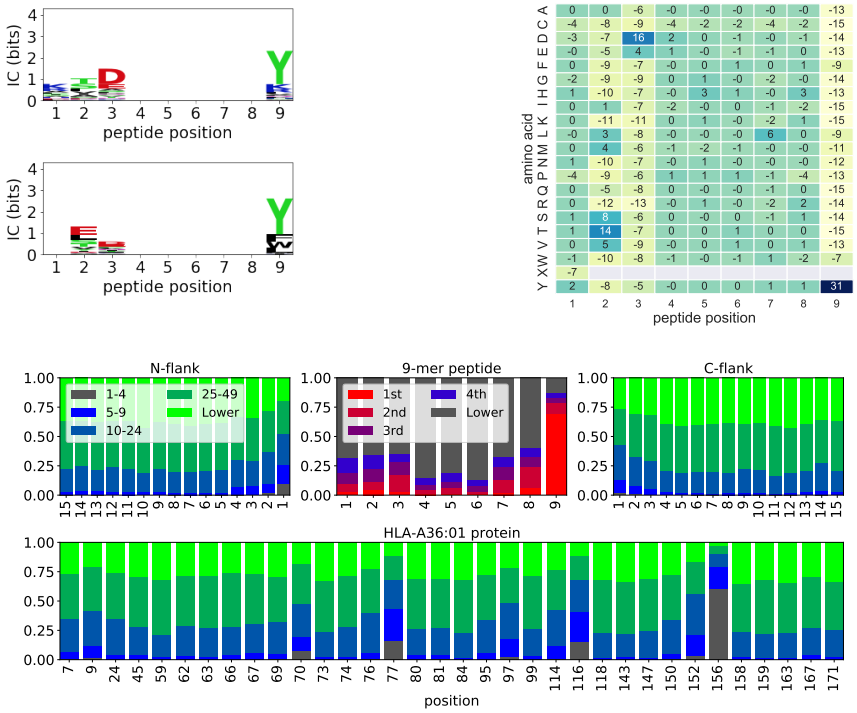


Figure 6.2: Motifs (top left), mean SHAP values (top right) and LIME feature ranks

6.4.3 HLA-A*74:01

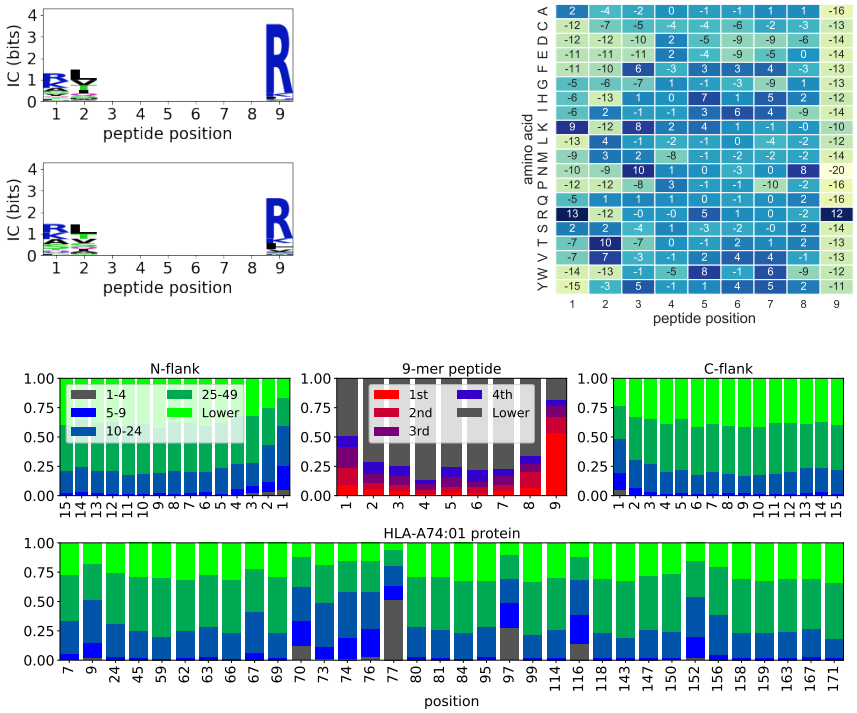


Figure 6.3: Motifs (top left), mean SHAP values (top right) and LIME feature ranks

6.4.4 HLA-B*37:01

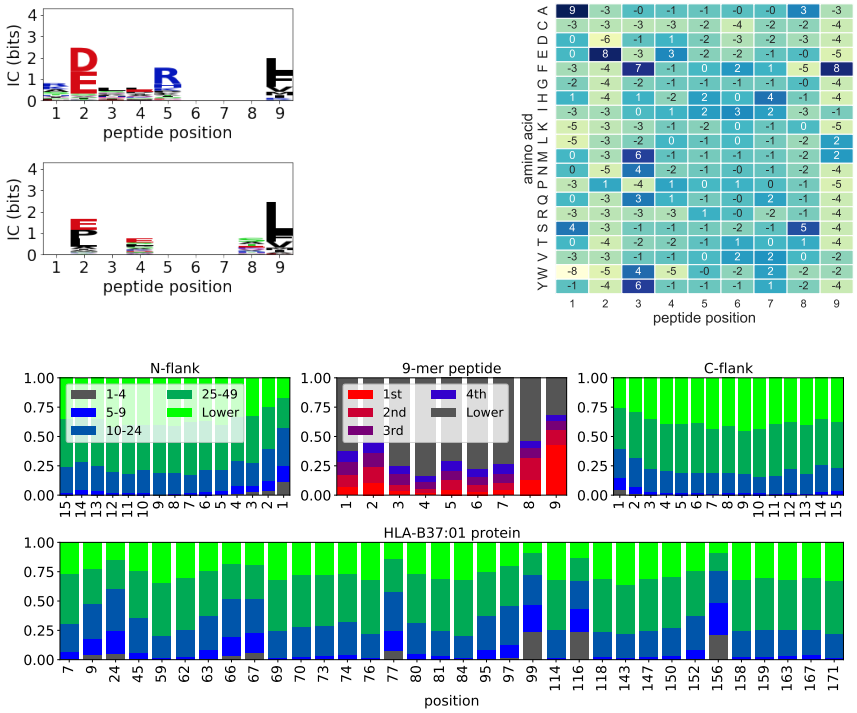


Figure 6.4: Motifs (top left), mean SHAP values (top right) and LIME feature ranks

6.4.5 HLA-B*46:01

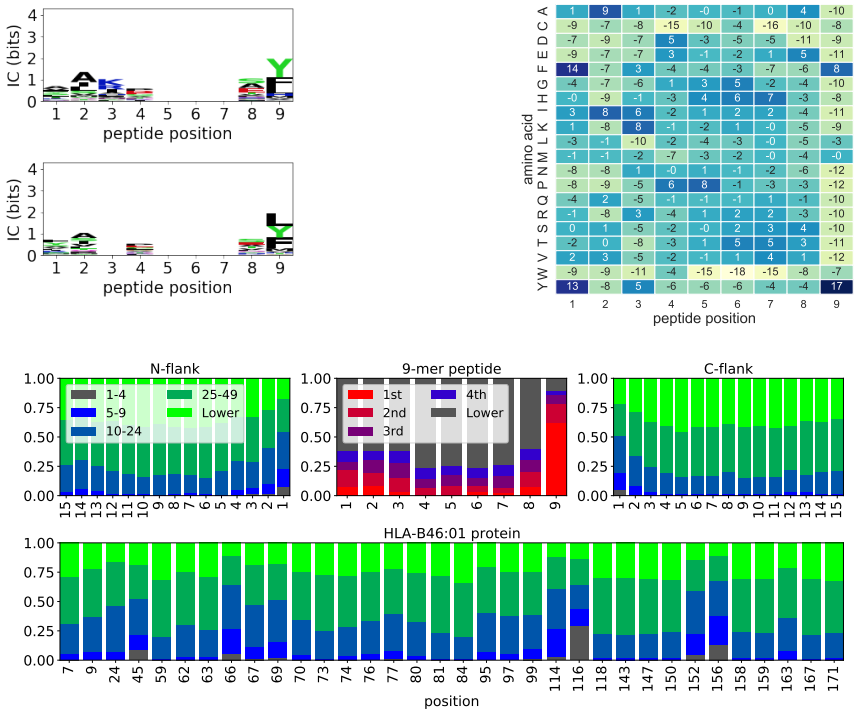


Figure 6.5: Motifs (top left), mean SHAP values (top right) and LIME feature ranks

6.4.6 HLA-B*58:01

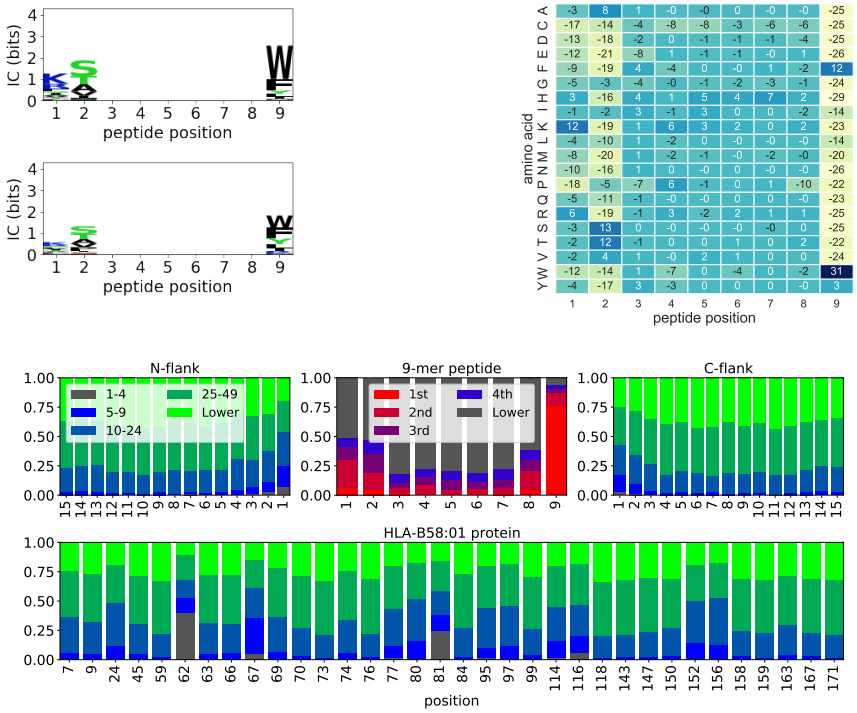


Figure 6.6: Motifs (top left), mean SHAP values (top right) and LIME feature ranks

6.4.7 HLA-B*58:02

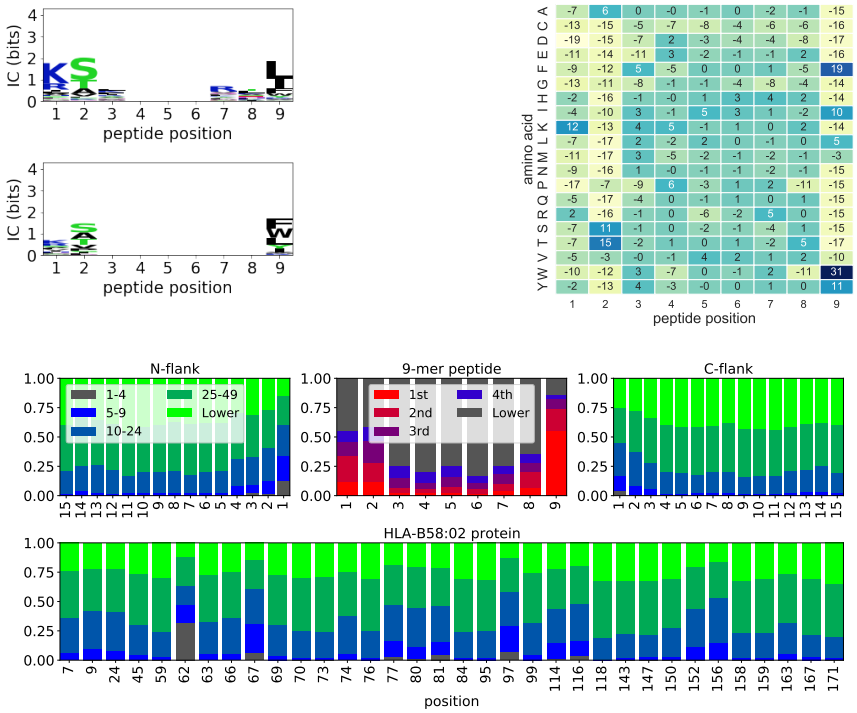


Figure 6.7: Motifs (top left), mean SHAP values (top right) and LIME feature ranks

6.4.8 HLA-C*15:02

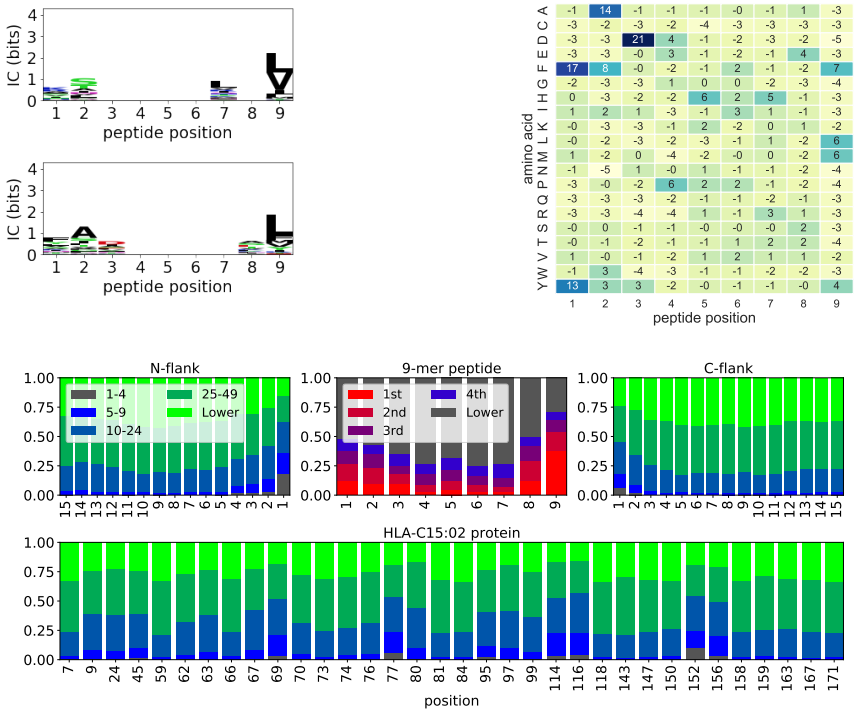


Figure 6.8: Motifs (top left), mean SHAP values (top right) and LIME feature ranks

6.4.9 HLA-C*17:01

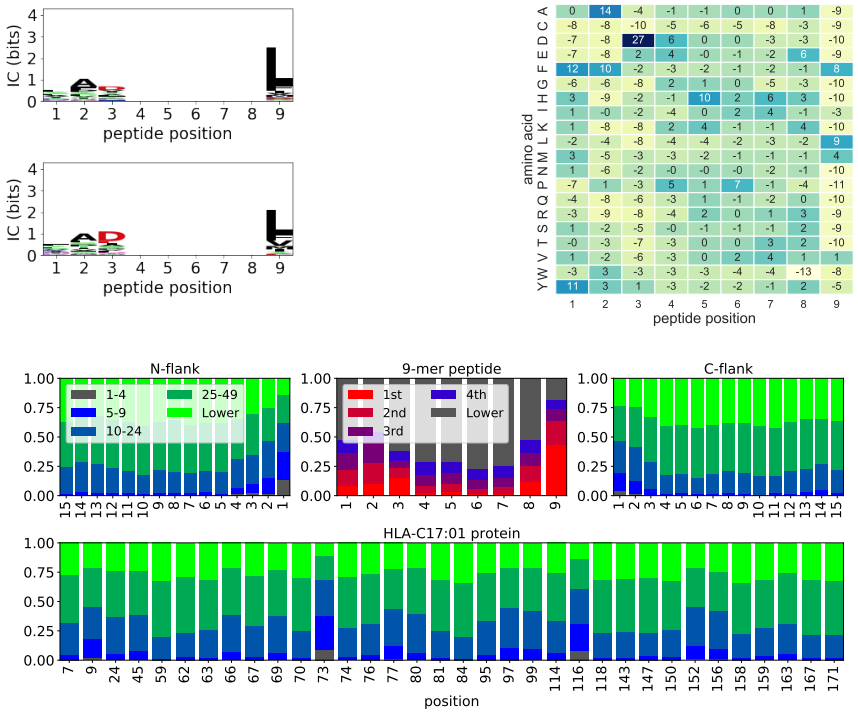


Figure 6.9: Motifs (top left), mean SHAP values (top right) and LIME feature ranks

Acronyms

AA amino acid. 3–6, 19, 23, 27, 28, 30, 31, 36, 37, 39

AP average precision. 12, 13, 30, 33–35, 44

AUC area-under-the-curve. 12, 13, 30, 33–35

BA binding affinity. 6, 10, 18–21

BERT Bidirectional Encoder Representations for Transformers. i, 2, 9, 13–15, 22, 27, 40

CNN convolutional neural network. 13, 22

CTL Cytotoxic T-lymphocytes. 1, 5, 6

DNA deoxyribonucleic acid. 4

EL eluted ligand. 7, 9, 18–21, 23, 24, 30

ER endoplasmic reticulum. 5, 6

FN false negative. 12

FP false positive. 12

FPR false positive rate. 12

HLA Human Leukocyte Antigen. 1, 6, 8, 9, 22, 23, 25, 30–32, 35, 37, 44

IC information content. 31, 39

LIME Local Interpretable Model-agnostic Explanations. i, 2, 16, 17, 30, 35, 36, 38–40, 45–49

LSTM Long Short-term Memory. 15

MA multi-allele. 19, 24, 29

MHC major histocompatibility complex. i, 1–3, 5–7, 9, 18–22, 24–30, 32, 36, 38–40

MHC-I MHC class I. 1–3, 5–7, 9, 18–20, 22

MHC-II MHC class II. 15, 19, 22

MIL Multi Instance Learning. 9, 10, 21, 29

ML machine learning. 2, 3, 15, 27

MLM Masked Language Modelling. 14, 15

MLP multi layer perceptron. 22, 29

mRNA messenger ribonucleic acid. 4

NSP Next Sentence Prediction. 14, 15

pMHC peptide:MHC protein complex. 1, 5, 6

PR precision-recall. 12, 13, 30, 34, 35

PRIDE PRoteomics IDEntifications Database. 23

PU positive and unlabelled. 10

PWM position weight matrix. 9

ResNet residual network. 15

ROC receiver-operating-curve. 12, 13, 30, 33–35

SA single-allele. 19, 24, 29, 32

SHAP SHapley Additive exPlanations. i, 2, 16–18, 30, 31, 35, 37–40, 45–49

SOTA state of the art. i, 2, 10, 13, 15, 21, 24, 32, 40

TAP transporter associated with antigen processing. 5, 20

TAPE Tasks Assessing Protein Embeddings. 15, 22, 27, 28

TCR T-cell receptor. 1, 2, 5, 6

TN true negative. 12

TP true positive. 12

TPR true positive rate. 12

Bibliography

[1] Encyclopaedia Britannica. *Major histocompatibility complex*. July 2021. URL: https://www.britannica.com/science/major-histocompatibility-complex.

[2] Y. M. Mosaad. "Clinical Role of Human Leukocyte Antigen in Health and Disease". In: *Scandinavian Journal of Immunology* 82.4 (2015), pp. 283–306. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/sji.12329 (visited on 07/13/2021).

[3] Kenneth Murphy and Casey Weaver. *Janeway's Immunobiology*. 9th. Garland Science, 2017.

[4] Hans-Christof Gasser. *Informatics Project Proposal: Generalizable models of antigen presentation for the characterization of understudied immune-landscapes*. May 2021.

[5] Ton N. Schumacher and Robert D. Schreiber. "Neoantigens in cancer immunotherapy". In: *Science* 348.6230 (Apr. 2015). Publisher: American Association for the Advancement of Science Section: Review, pp. 69–74. URL: https://science.sciencemag.org/content/348/6230/69 (visited on 06/15/2021).

[6] Zheying Zhang et al. "Neoantigen: A New Breakthrough in Tumor Immunotherapy". In: *Frontiers in Immunology* 0 (2021). Publisher: Frontiers. URL: https://www.frontiersin.org/articles/10.3389/fimmu.2021.672356/full (visited on 08/11/2021).

[7] Cynthia M. Fehres et al. "Understanding the Biology of Antigen Cross-Presentation for the Design of Vaccines Against Cancer". In: *Frontiers in Immunology* 5 (2014). Publisher: Frontiers. URL: https://www.frontiersin.org/articles/10.3389/fimmu.2014.00149/full (visited on 06/15/2021).

[8] Guangyuan Li et al. “DeepImmuno: deep learning-empowered prediction and generation of immunogenic peptides for T-cell immunity”. In: *Briefings in Bioinformatics* (May 2021). URL: https://doi.org/10.1093/bib/bbab160 (visited on 06/09/2021).

[9] Joseph D. Comber and Ramila Philip. “MHC class I antigen presentation and implications for developing a new generation of therapeutic vaccines”. In: *Therapeutic Advances in Vaccines* 2.3 (May 2014). Publisher: SAGE Publications Ltd STM, pp. 77–89. URL: https://doi.org/10.1177/2051013614525375 (visited on 04/21/2021).

[10] Thomas D. Pollard et al. *Cell Biology*. 3rd ed. Elsevier Health Sciences, Nov. 2016.

[11] Jonathan Crowe and Tony Bradshaw. *Chemistry for the Biosciences - The essential concepts*. 3rd. Oxford University Press, 2014.

[12] Kenneth L. Rock, Eric Reits, and Jacques Neefjes. “Present Yourself! By MHC Class I and MHC Class II Molecules”. In: *Trends in Immunology* 37.11 (Nov. 2016), pp. 724–737. URL: https://www.sciencedirect.com/science/article/pii/S1471490616301004 (visited on 04/05/2021).

[13] Morten Nielsen et al. “Immunoinformatics: Predicting Peptide–MHC Binding”. In: *Annual Review of Biomedical Data Science* 3.1 (2020), pp. 191–215. URL: https://doi.org/10.1146/annurev-biodatasci-021920-100259 (visited on 04/08/2021).

[14] Morten Nielsen et al. “The role of the proteasome in generating cytotoxic T-cell epitopes: insights obtained from improved predictions of proteasomal cleavage”. In: *Immunogenetics* 57.1 (Apr. 2005), pp. 33–41. URL: https://doi.org/10.1007/s00251-005-0781-7 (visited on 07/08/2021).

[15] Jonathan W. Yewdell and Jack R. Bennink. “Immunodominance in major histocompatibility complex class i–restricted t lymphocyte responses”. In: *Annual Review of Immunology* 17.1 (Apr. 1999). Publisher: Annual Reviews, pp. 51–88. URL: https://www.annualreviews.org/doi/10.1146/annurev.immunol.17.1.51 (visited on 07/02/2021).

[16] EBI. *Immuno Polymorphism Database - Statistics*. 2021. URL: https://www.ebi.ac.uk/ipd/imgt/hla/stats.html (visited on 04/05/2021).

[17] Jennifer G. Abelin et al. “Mass Spectrometry Profiling of HLA-Associated Peptidomes in Mono-allelic Cells Enables More Accurate Epitope Prediction”. In: *Immunity* 46.2 (Feb. 2017). Publisher: Elsevier, pp. 315–326. URL: https://www.cell.com/immunity/abstract/S1074-7613(17)30042-0 (visited on 04/22/2021).

[18] Wikipedia. *Major histocompatibility complex*. Page Version ID: 1020006475. Apr. 2021. URL: https://en.wikipedia.org/w/index.php?title=Major_histocompatibility_complex&oldid=1020006475 (visited on 05/04/2021).

[19] Bruno Alvarez et al. “NNAlign_MA; MHC Peptidome Deconvolution for Accurate MHC Binding Motif Characterization and Improved T-cell Epitope Predictions”. In: *Molecular & Cellular Proteomics* 18.12 (Dec. 2019). Publisher: Elsevier, pp. 2459–2477. URL: https://www.mcponline.org/article/S1535-9476(20)31649-2/abstract (visited on 04/05/2021).

[20] D. F. Hunt et al. “Characterization of peptides bound to the class I MHC molecule HLA-A2.1 by mass spectrometry”. In: *Science* 255.5049 (Mar. 1992). Publisher: American Association for the Advancement of Science Section: Reports, pp. 1261–1263. URL: https://science.sciencemag.org/content/255/5049/1261 (visited on 07/19/2021).

[21] Anette Stryhn et al. “Longer peptide can be accommodated in the MHC class I binding site by a protrusion mechanism”. In: *European Journal of Immunology* 30.11 (2000), pp. 3089–3099. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/1521-4141%28200011%2930%3A11%3C3089%3A%3AAID-IMMU3089%3E3.0.CO%3B2-5 (visited on 07/10/2021).

[22] Marie-Paule Lefranc et al. “IMGT unique numbering for MHC groove G-DOMAIN and MHC superfamily (MhcSF) G-LIKE-DOMAIN”. In: *Developmental & Comparative Immunology* 29.11 (Jan. 2005), pp. 917–938. URL: https://www.sciencedirect.com/science/article/pii/S0145305X05000650 (visited on 05/14/2021).

[23] HLA Informatics Group. *HLA Nomenclature @ hla.alleles.org*. URL: http://hla.alleles.org/nomenclature/naming.html (visited on 07/13/2021).

[24] Hanneke W. M. van Deutekom and Can Keşmir. “Zooming into the binding groove of HLA molecules: which positions and which substitutions change peptide binding most?” In: *Immunogenetics* 67.8 (2015), pp. 425–436. URL:

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4498290/ (visited on 07/10/2021).

[25] Michal Bassani-Sternberg and David Gfeller. "Unsupervised HLA Peptidome Deconvolution Improves Ligand Prediction Accuracy and Predicts Cooperative Effects in Peptide–HLA Interactions". In: *The Journal of Immunology* 197.6 (Sept. 2016). Publisher: American Association of Immunologists Section: SYSTEMS IMMUNOLOGY, pp. 2492–2499. URL: https://www.jimmunol.org/content/197/6/2492 (visited on 07/19/2021).

[26] Maximilian Ilse, Jakub Tomczak, and Max Welling. "Attention-based Deep Multiple Instance Learning". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, July 2018, pp. 2127–2136. URL: http://proceedings.mlr.press/v80/ilse18a.html.

[27] Xinggang Wang et al. "Revisiting multiple instance neural networks". In: *Pattern Recognition* 74 (Feb. 2018), pp. 15–24. URL: https://linkinghub.elsevier.com/retrieve/pii/S0031320317303382 (visited on 05/06/2021).

[28] Jan Ramon and Luc De Raedt. "Multi instance neural networks". In: *Proceedings of the ICML-2000 workshop on attribute-value and relational learning*. 2000, pp. 53–60. URL: https://lirias.kuleuven.be/retrieve/416293 (visited on 06/05/2021).

[29] Jun Cheng et al. "BERTMHC: Improves MHC-peptide class II interaction prediction with transformer and multiple instance learning". In: *bioRxiv* (Nov. 2020). Publisher: Cold Spring Harbor Laboratory Section: New Results, p. 2020.11.24.396101. URL: https://www.biorxiv.org/content/10.1101/2020.11.24.396101v1 (visited on 04/05/2021).

[30] Jessa Bekker and Jesse Davis. "Learning from positive and unlabeled data: a survey". In: *Machine Learning* 109.4 (Apr. 2020), pp. 719–760. URL: https://doi.org/10.1007/s10994-020-05877-5 (visited on 07/17/2021).

[31] Charles Elkan and Keith Noto. "Learning classifiers from only positive and unlabeled data". In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '08. New York, NY, USA: Association for Computing Machinery, Aug. 2008, pp. 213–220. URL: https://doi.org/10.1145/1401890.1401920 (visited on 07/17/2021).

[32] Haibo He and Edwardo A. Garcia. "Learning from Imbalanced Data". In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (Sept. 2009). Conference Name: IEEE Transactions on Knowledge and Data Engineering, pp. 1263–1284. URL: https://ieeexplore.ieee.org/document/5128907.

[33] G. M. Weiss and F. Provost. "Learning When Training Data are Costly: The Effect of Class Distribution on Tree Induction". In: *Journal of Artificial Intelligence Research* 19 (Oct. 2003), pp. 315–354. URL: https://jair.org/index.php/jair/article/view/10346 (visited on 08/15/2021).

[34] sklearn. *sklearn.metrics.average_precision_score — scikit-learn 0.24.2 documentation*. July 2021. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html (visited on 07/21/2021).

[35] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. URL: https://www.aclweb.org/anthology/N19-1423 (visited on 04/21/2021).

[36] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://papers.nips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html (visited on 08/07/2021).

[37] Justin Weinberg. *Philosophers On GPT-3 (updated with replies by GPT-3)*. Section: Public philosophy and outreach. July 2020. URL: https://dailynous.com/2020/07/30/philosophers-gpt-3/ (visited on 07/17/2021).

[38] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems* 30 (2017), pp. 5998–6008. URL: https://papers.nips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (visited on 11/06/2020).

[39] Yannic Kilcher. *Attention Is All You Need*. Nov. 2017. URL: https://www.youtube.com/watch?v=iDulhoQ2pro (visited on 04/01/2021).

[40] Alexey Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: Sept. 2020. URL: https://openreview.net/forum?id=YicbFdNTTy (visited on 01/22/2021).

[41] Antreas Antoniou. *MLP - Advanced Session 1: Transformers*. University of Edinburgh, Nov. 2020.

[42] Shubham Toshniwal et al. “A Cross-Task Analysis of Text Span Representations”. In: *arXiv:2006.03866 [cs]* (June 2020). arXiv: 2006.03866. URL: http://arxiv.org/abs/2006.03866 (visited on 04/25/2021).

[43] Łukasz Kaiser. *Attention is all you need; Attentional Neural Network Models — Łukasz Kaiser — Masterclass*. Pi School, Oct. 2017. URL: https://www.youtube.com/watch?v=rBCqOTEfxvg (visited on 07/31/2021).

[44] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. “Reformer: The Efficient Transformer”. In: Apr. 2020. URL: https://iclr.cc/virtual_2020/poster_rkgNKkHtvB.html (visited on 08/15/2021).

[45] Roshan Rao et al. “Evaluating Protein Transfer Learning with TAPE”. In: *Advances in neural information processing systems* 32 (Dec. 2019), pp. 9689–9701. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7774645/ (visited on 04/21/2021).

[46] Pfam. *TAPE Pfam pretraining data*. 2019. URL: http://s3.amazonaws.com/proteindata/data_pytorch/pfam.tar.gz.

[47] Ahmed Elnaggar et al. “ProtTrans: Towards Cracking the Language of Life’s Code Through Self-Supervised Deep Learning and High Performance Computing”. In: *arXiv:2007.06225 [cs, stat]* (May 2021). arXiv: 2007.06225. URL: http://arxiv.org/abs/2007.06225 (visited on 05/17/2021).

[48] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “”Why Should I Trust You?”: Explaining the Predictions of Any Classifier”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’16. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 1135–1144. URL: https://doi.org/10.1145/2939672.2939778 (visited on 08/15/2021).

[49] Scott M. Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., Dec. 2017, pp. 4768–4777. URL: https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html (visited on 06/18/2021).

[50] Lloyd Shapley. "A value for n-person games". In: *Contributions to the Theory of Games (AM-28), Volume II*. Princeton University Press, 1953.

[51] Morten Nielsen et al. "Reliable prediction of T-cell epitopes using neural networks with novel sequence representations". In: *Protein Science: A Publication of the Protein Society* 12.5 (May 2003), pp. 1007–1017. URL: https://onlinelibrary.wiley.com/doi/full/10.1110/ps.0239403.

[52] Morten Nielsen and Massimo Andreatta. "NetMHCpan-3.0; improved prediction of binding to MHC class I molecules integrating information from multiple receptor and peptide length datasets". In: *Genome Medicine* 8.1 (Mar. 2016), p. 33. URL: https://doi.org/10.1186/s13073-016-0288-x (visited on 07/18/2021).

[53] Sinu Paul et al. "HLA class I alleles are associated with peptide binding repertoires of different size, affinity and immunogenicity". In: *Journal of Immunology* 191.12 (Dec. 2013), 10.4049/jimmunol.1302101. URL: https://www.jimmunol.org/content/191/12/5831 (visited on 07/19/2021).

[54] Keiko Udaka et al. "Empirical Evaluation of a Dynamic Experiment Design Method for Prediction of MHC Class I-Binding Peptides". In: *The Journal of Immunology* 169.10 (Nov. 2002), pp. 5744–5753. URL: http://www.jimmunol.org/lookup/doi/10.4049/jimmunol.169.10.5744 (visited on 07/02/2021).

[55] Birkir Reynisson et al. "NetMHCpan-4.1 and NetMHCIIpan-4.0: improved predictions of MHC antigen presentation by concurrent motif deconvolution and integration of MS MHC eluted ligand data". In: *Nucleic Acids Research* 48.W1 (July 2020), W449–W454. URL: https://doi.org/10.1093/nar/gkaa379 (visited on 04/05/2021).

[56] Timothy J. O'Donnell, Alex Rubinsteyn, and Uri Laserson. "MHCflurry 2.0: Improved Pan-Allele Prediction of MHC Class I-Presented Peptides by Incorporating Antigen Processing". In: *Cell Systems* 11.1 (July 2020). Publisher:

Cell Press, 42–48.e7. URL: https://www.sciencedirect.com/science/article/pii/S2405471220302398 (visited on 04/05/2021).

[57] Morten Nielsen et al. "NetMHCpan, a Method for Quantitative Predictions of Peptide Binding to Any HLA-A and -B Locus Protein of Known Sequence". In: *PLOS ONE* 2.8 (Aug. 2007). Publisher: Public Library of Science, e796. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0000796 (visited on 05/08/2021).

[58] Yan Hu et al. "ACME: pan-specific peptide–MHC class I binding prediction through attention-based deep neural networks". In: *Bioinformatics* 35.23 (Dec. 2019), pp. 4946–4954. URL: https://doi.org/10.1093/bioinformatics/btz427 (visited on 04/22/2021).

[59] Haoyang Zeng and David K. Gifford. "Quantification of Uncertainty in Peptide-MHC Binding Prediction Improves High-Affinity Peptide Selection for Therapeutic Design". In: *Cell Systems* 9.2 (Aug. 2019). Publisher: Elsevier, 159–166.e3. URL: https://www.cell.com/cell-systems/abstract/S2405-4712(19)30153-X (visited on 07/26/2021).

[60] Gopalakrishnan Venkatesh et al. "MHCAttnNet: predicting MHC-peptide bindings for MHC alleles classes I and II using an attention-based deep neural model". In: *Bioinformatics* 36 (July 2020), pp. i399–i406. URL: https://doi.org/10.1093/bioinformatics/btaa479 (visited on 07/26/2021).

[61] Yasset Perez-Riverol et al. "The PRIDE database and related tools and resources in 2019: improving support for quantification data". In: *Nucleic Acids Research* 47 (Jan. 2019), pp. D442–D450. URL: https://doi.org/10.1093/nar/gky1106 (visited on 02/19/2021).

[62] Ana Marcu et al. "The HLA Ligand Atlas - A resource of natural HLA ligands presented on benign tissues". In: *bioRxiv* (July 2020). Publisher: Cold Spring Harbor Laboratory Section: New Results. URL: https://www.biorxiv.org/content/10.1101/778944v2 (visited on 04/20/2021).

[63] Ensembl. *BioMart - Paralogue Table*. June 2021. URL: http://www.ensembl.org/biomart/martview/48db0485487a9c26d139bc5d7cdda420?VIRTUALSCHEMANAME=default&ATTRIBUTES=hsapiens_gene_ensembl.default.homologs.ensembl_gene_id%7Chsapiens_gene_ensembl.default.homologs.ensembl_transcript_id%7Chsapiens_gene_ensembl.default.homologs.

ensembl_peptide_id%7Chsapiens_gene_ensembl.default.homologs.hsapiens_paralog_ensembl_gene%7Chsapiens_gene_ensembl.default.homologs.hsapiens_paralog_associated_gene_name%7Chsapiens_gene_ensembl.default.homologs.hsapiens_paralog_ensembl_peptide&FILTERS=&VISIBLEPANEL=resultspanel.

[64] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv:1412.6980 [cs]* (Jan. 2017). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980 (visited on 10/22/2020).

[65] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org/.

[66] Thomas D. Schneider and R.Michael Stephens. "Sequence logos: a new way to display consensus sequences". In: *Nucleic Acids Research* (Oct. 1990). URL: https://doi.org/10.1093/nar/18.20.6097 (visited on 07/26/2021).

[67] Ammar Tareen and Justin B. Kinney. "Logomaker: Beautiful sequence logos in python". In: *Bioinformatics* (May 2019). Publisher: Cold Spring Harbor Laboratory Section: New Results. URL: https://www.biorxiv.org/content/10.1101/635029v1 (visited on 08/01/2021).

[68] Jingcheng Wu et al. "DeepHLApan: A Deep Learning Approach for Neoantigen Prediction Considering Both HLA-Peptide Binding and Immunogenicity". In: *Frontiers in Immunology* 10 (2019). URL: https://www.frontiersin.org/articles/10.3389/fimmu.2019.02559/full (visited on 03/14/2021).

[69] Jun Liu and George F. Gao. "Major Histocompatibility Complex: Interaction with Peptides". In: *Encyclopedia of Life Sciences*. Wiley-Blackwell, 2011. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470015902.a0000922.pub2 (visited on 08/15/2021).