Explainable Artificial Intelligence for Peptide Presentation Using Statistical Fault Localisation Techniques

Gwenyth Rooijackers

MInf Project (Part 2) Report
Master of Informatics
School of Informatics
University of Edinburgh

2022

Abstract

The rise of complex black-box machine learning models has led to an increase in explainable AI research to understand their reasoning and build trust in these models. Trust is especially important in medical applications. Peptide presentation is the immune system's window into the cell and a key step in peptide vaccine development. This project aimed to interpret the decisions of the peptide presentation prediction model ImmunoBERT using the DeepCover explainability technique.

The quality of the produced explanations was evaluated by comparing the explanations to previous research and known biological properties of the input. This showed that the positions uncovered by DeepCover could be biologically motivated, and that the explanations were meaningful. Comparisons of the explanations produces by DeepCover to the explanations produced by the other explanation techniques LIME and DC-Causal showed that the different techniques agreed less on the ranking of the input positions. However, we note that evaluation by agreement is flawed and where they did not agree it is likely that they uncovered different but valid explanations as the problem does not have one true explanation. The evaluation also showed that the different fault localisation metrics implemented in DeepCover produced very similar explanations. Lastly, the runtime was significantly improved through data parallelisation when explaining multiple instances.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

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.

(*Gwenyth Rooijackers*)

Acknowledgements

A big thank you to my supervisor Dr. Ajitha Rajan for the ongoing support, frequent meetings and helpful feedback on the report.

Also thank you to Hans-Christof Gasser for the work on and support with ImmunoBERT and to Linda Mazánová for sharing her DC-Causal results included in the comparisons.

Table of Contents

Chapter 1

Introduction

Peptide Presentation. A vital task of the immune system is to kill infected and tumorigenic cells. The role of cytotoxic T-lymphocytes is to detect affected cells and kill them [1, 2, 3]. However, they cannot look inside the cell to see if it is affected so the cell itself has a system to broadcast its internal state to the immune system [1, 2, 3]. Cells do this using major histocompatibility complex-I (MHC-I) molecules that present snippets, called peptides, of the internal proteins on the cell surface for the immune system to diagnose [1, 2, 3].

ImmunoBERT is a model that predicts peptide presentation [4]. It is a Bidirectional Encoder Representations from Transformers (BERT)-based architecture that treats peptides and MHC-I proteins as amino acid sequences and outputs a presentation score representing how likely it is that a peptide would be presented by the given MHC-I molecule [4]. Understanding and modelling peptide presentation is important to aid peptide vaccine development for viruses and cancer [2, 4]. However, ImmunoBERT, like other peptide presentation models, is a black-box model and interpreting how its decisions are made is crucial to actually understanding more about which peptides are most likely to be presented by the MHC-I molecule [4, 5].

Explainable artificial intelligence (AI) aims to do just that, explaining the reasoning of AI models to expose bias, build trust and better understand the model and the problem it models [6]. Moreover, decision transparency and trust is particularly important in the medical domain [7], with predictability and translation of *in silico* (computer models or experiments) to the clinical setting being of special interest. DeepCover is an explainable AI technique developed for images [8]. DeepCover repeatedly mutates the input and measures the effect on the output using statistical fault localisation measures from software testing [8]. It computes an importance score per component of the input (such as input position or super-pixel) and provides an algorithm for building minimal explanations that uses the importance scores to produce minimal and sufficient explanations as a subset of the components of the input instance [8]. The minimal explanation algorithm is one of the main things that sets DeepCover apart from the common explanation method LIME, which also uses local perturbations to create explanations [9]. The goal of this project was to modify DeepCover to interpret the ImmunoBERT model.

1.1 Overview of the Report

Aim. The aim of this project was to use DeepCover to explain the classification decisions made by presentation prediction model ImmunoBERT and evaluate the quality of the explanations. Another aim was to improve the runtime of DeepCover to make it more realistic for explaining multiple instances.

Chapters. Chapter 2 provides further background on peptide presentation biology and ImmunoBERT as well as an overview of other explanation techniques in the related works in Section 2.3. Chapter 3 explains how DeepCover was modified to interpret the ImmunoBERT model which involved handling the input format, mutating the amino acid sequences using substitution mutations mirroring real life and parallelising the generation of explanations. Chapter 4 outlines the setup of the experiments to evaluate whether DeepCover was successful in the peptide presentation domain and Chapter 5 presents the results of the experiments. The final chapter is Chapter 6 which discusses the results, limitation and further work.

Results. The results showed that DeepCover was successfully modified to handle the ImmunoBERT input, but it struggled with the binary class problem (presented or not presented), which made it unable to create full and minimal explanations for a portion of the instances. The explanations that were produced were useful, as they highlighted positions of the input which could be biologically motivated as important for binding between the peptide and MHC-I molecule. The explanations produced using the different statistical fault localisation measures included in DeepCover agreed on both the trends and on the instance-level. Meanwhile, comparison with the other explanation methods LIME and DC-Causal showed less agreement, with the caveat that evaluation by agreement is a disputed evaluation method for explainable AI [10]. Finally, the runtime of DeepCover was poor compared to LIME but improvements from parallelisation implemented in this project made it competitive.

Contributions. In summary, the contributions presented in this report are:

- Modifying DeepCover to handle the protein input and explain the decisions of the ImmunoBERT model.
- Parallelising the execution of DeepCover for explaining multiple instances.
- Experiments to evaluate whether DeepCover can produce explanations for ImmunoBERT and evaluation of the quality of explanations for ImmunoBERT through biological interpretation and comparisons with other explainability techniques.
- Comparing the explanations produced using DeepCover's different fault localisation measures.

MInf Project (Part 1). The first part of the MInf project was creating a tool to produce execution traces of compiled Java programs [11]. The current report presents MInf project part 2 which is loosely related to the first part as it explores another application of sequences. Part 2 is not a direct continuation of part 1 as it reached a suitable and natural end when it resulted in a journal publication [12].

Chapter 2

Background

2.1 Biology: Peptide presentation by MHC-I molecules

The immune system is a network of biological processes of cells and molecules that defend the body against disease. There are two types of immune responses. The first type are innate (natural) responses which happen with the same strength every time they are activated by a certain entity, e.g. a pathogen. The second type are acquired (adaptive) responses make up the memory of the immune system, and are improved every time they are activated in response to a specific threat [1].

One cell that is part of an acquired immune response mechanism is cytotoxic T-lymphocytes (CTLs), also called CD8+ T-cells, whose role is to identify and kill infected (particularly with viruses) and tumorigenic cells [1, 2, 3]. However, CTLs cannot see what is happening inside the cells by themselves.

Major histocompatibility complex (**MHC**) molecules are present in all nucleated cells in the body, and their task is to expose the internal state of the cell to the immune system by presenting snippets (**peptides**) of internal proteins on the outside of the cell [1, 2, 3]. CTLs specifically bind to MHC class I (MHC-I) proteins, also referred to as human leukocyte antigens (**HLAs**) in humans, and kill the cell only if the presented peptides are not part of the normal set of proteins expressed in the cell, the proteome [2]. Briefly, the steps in the MHC-I pathway are: [3]

1. Proteins (both normal to the proteome and pathogenic, i.e. disease-causing) in the cell are degraded into peptides by proteasome. They are further trimmed and many are destroyed by cytosolic peptidases.
2. The surviving peptides are transported into the endoplasmic reticulum of the cell by a peptide transporter called transporter associated with antigen processing.
3. The peptides in the endoplasmic reticulum are loaded into the MHC-I molecules, which produce a stable complex that can be transported to the cell surface for presentation.
4. The MHC-I/peptide complexes are presented on the outside of the cell and can be detected and bound to by scanning CTLs.

Understanding which peptides will be presented by the MHC-I molecules is important in vaccine development. For instance, by identifying which cancer antigens will be presented on the cell surface using a combination of *in silico* prediction models and *in vitro* experiments (outside of a living organism), vaccines can be developed that leverage the memory capabilities of the acquired immune response of the CTLs [2].

2.1.1 Biological Properties of the MHC-I Molecules and Peptides

Alleles are variant forms of a gene. The MHC-I protein is encoded in the genetic code/DNA on chromosome six which contains three encodings for MHC-I molecules, HLA-A, HLA-B and HLA-C [2, 3]. Additionally, because most human cells are diploid, meaning that they have two versions of most chromosomes, between three and six different HLA alleles are expressed in an individual's cells [3]. MHC molecules are unusually polymorphic, meaning that different people express different MHC-I molecules, which affects which peptides are presented on their cell surface [2, 3]. The set of peptides presented on the surface of the cells is called the immunopeptidome. The diversity of the MHC molecules and immunopeptidome in the population is important as it creates a diverse protection against antigens and viruses.

The peptide and MHC-I proteins are presented as sequences of amino acids in this project where the start side in reading direction is called the **N-terminus** and the end the **C-terminus**. The MHC molecule has an outward-facing groove where the peptides bind which has been divided into pockets A-F [14], as shown in Figure 2.1. **Pockets A and F** usually accommodate the N and C termini of the peptides, respectively [14]. Most of the preserved mutations (polymorphism) exist in the peptide-binding groove as that diversifies which peptides are presented by the MHC molecule [14].

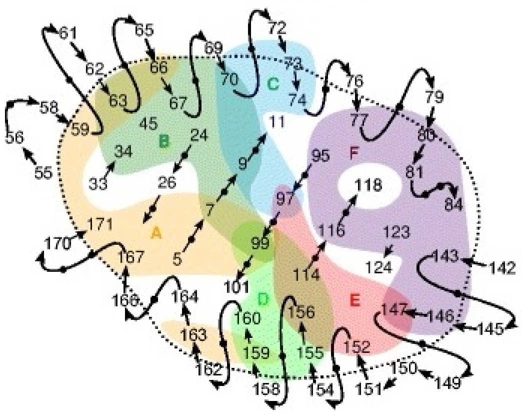


Figure 2.1: The positions of the peptide binding groove of the MHC protein and which pocket they belong to for allele HLA-A2. Image from van Deutekom and Keşmir [13].

Another important structural property of the MHC-I/peptide complexes are the **anchor residues**, i.e. the amino acids (also called residues when part of a protein or peptide) of the peptide that bind to the pockets of the MHC-I molecule [3].

2.1.2 ImmunoBERT

ImmunoBERT is a recently developed *in silico* MHC-I peptide presentation prediction model. Bidirectional Encoder Representations from Transformers (BERT) is a transformer model originally developed for natural language tasks [15]. ImmunoBERT is built on a pre-trained BERT architecture called Tasks Assessing Protein Embeddings

(TAPE) for protein inputs [4]. ImmunoBERT answers the question "Will the given peptide be presented by the given MHC molecule?" with a likelihood score between zero and one [4]. The input to ImmunoBERT is the peptide to predict the presentation score for, its surroundings in the source protein, also called the flanks, and a pseudo sequence of the MHC-I molecule that might present the peptide [4]. The **pseudo sequence** is a sub-sequence of the MHC-I protein containing the amino acids that are physically close to the peptide in the MHC-I/peptide complexes [16].

The data used for training and evaluating the ImmunoBERT model was collected with the eluted ligand approach which gives the whole immunopeptidome of a cell and all MHC alleles in the cell [4]. The consequences of eluted ligand are that the data only contains positive/presented examples and that the peptide presentation is not mapped to a specific MHC allele [4]. Negative examples, or decoys, are peptides which were not presented by the MHC-I molecules. To address the first problem and create negative examples, the author sampled 99 random decoy peptides per positive example from the set of proteins that the observed peptides originated from [4]. The second problem was handled during training, by adding deconvolution to associate each peptide to its presenting MHC-I allele [4].

2.2 Explainable AI

More complex machine learning (ML) models such as deep neural networks act as a black box: they are given an input and produce an output, but the reasoning behind the decision is not clear [6]. Explainable AI aims to present the relationship between the input instances and a machine learning model's predictions, to make the decisions more transparent [6, 17]. The ultimate goal of explanations is to increase the human's trust in the model and its decision-making [6, 7, 17]. The confidence, safety, security, privacy, ethics, fairness and trust needed in medicine and medical research makes explainability extra important in this domain [7].

While many explainable AI techniques are general and can be applied to different types of data and ML models, most methods are developed with image or natural language processing (NLP) tasks in mind. Techniques developed for either natural language or image data can be beneficial for the other as well [18]. Proteins represented as a sequence of amino acids have allowed for the use of NLP ML models to be applied to protein tasks [19]. Ofer et al. [19] outlined strategies and pitfalls of using NLP ML models on proteins. They recognise that NLP techniques have had great success in the protein domain but do not extend their comparisons to explainability.

Explainable AI methods can be divided into categories to describe their applicability. These categories include [6]:

- Origin of explanation: **Intrinsic** (model transparency) explainability describes ML methods that are explainable by nature and include simple linear models (such as linear or logistic regression) and example-based models such as K-nearest neighbours. **Post-hoc** explanation techniques explain the ML model's decision without providing insight into the mechanisms by which the model reached it, much like an explanation of a human's reasoning where the brain acts as a black box.

- Application domain: **Model-specific** techniques are dependent on which ML architecture they are explaining while **model-agnostic** techniques can be applied to any ML model, in a post-hoc fashion.
- Scope of interpretability: The scope of interpretability defines whether the method explains the prediction of a single input instance (**local**) or the behaviour of the entire model (**global**).

2.3 Related work

Recent and common explainable AI research for images, natural language and biological applications was surveyed and included in this section. Table 2.1 presents the techniques and their classifications as outlined above. The focus of this section is on post hoc methods as these can be applied in settings with an existing prediction model. The techniques included are the most recent developments or the most common, measured as the highest number of citations.

Table 2.1: Classifications of the presented Explainable AI methods.

Method	**Origin**	**Application domain**	**Scope**	**Type of data**
iRF [20]	Intrinsic	Specific	Global	Biological data
iRF-LOOP [21]				Biological data
ALPODS [22]			Local	High dimensional biological data
SP-LIME [9]	Post hoc	Agnostic	Global	Any
MMD-critic [23]				Any
LIME [9]			Local	Any
STREAK [24]				Images
DLIME [25]				Any
LIMETREE [26]				Any
Anchor [27]				Any
MAPLE [28]				Tabular
DICE [29]				Any
DeepCover [8]				Images
OLM [30]				Natural language
SHAP [31]			Local/Global	Any
k-LIME [32]				Any
PoSHAP [33]				Biological sequences
Integrated gradients [34]		Specific	Local	Any
DeepLIFT [35]				Any
Attention flow [36]				Natural language
Attention rollout [36]				Natural language
MICE [37]				Natural language
LinearSHAP [31]			Local/Global	Any
DeepSHAP [31]				Any
TransSHAP [38]				Natural language
GradSHAP [39]				Any
Representation erasure [40]				Natural language/Any

2.3.1 Common Explainable AI methods for images

Image tasks usually use black box ML models such as deep neural networks, so it is not surprising that explainable AI has been focused on image inputs. While many of these methods can be applied to any type of data, they were originally developed to explain image data.

2.3.1.1 Local Interpretable Model-Agnostic Explanations (LIME)

LIME is a popular explainability technique which randomly samples inputs around the instance being explained using post hoc perturbations of the original input instance. By feeding these samples into the model, LIME creates a new, local and linear interpretable model around the original input [9]. The authors provide a method called SP-LIME to choose a set of representative explanations that cover a diverse set of globally important features as their take on a global explanation [9].

Variations of LIME include DLIME [25], k-LIME [32] and LIMEtree [26]. LIME uses random perturbations which can result in different explanations for the same input and nonsensical instances. DLIME addresses the instability of the explanations created by standard LIME and the risk of random perturbations resulting in nonsensical instances [25]. DLIME uses agglomerative clustering to cluster similar samples in the training data together [25]. The produced dendrogram is cut where the distance between two successive clusters is the greatest [25]. A k-nearest neighbour classifier is used to assign new data points to a cluster and an explanation is constructed as a linear regression model fitted to all instances in the assigned cluster [25].

Similarly to DLIME, k-LIME clusters the training data points first, but using k-means clustering [32]. The local linear model is then fitted to all the data points in the cluster, unless the cluster is less than 20 instances in size [32]. In that case, k-LIME uses a global linear surrogate model fitted to the entire training set [32]. The global surrogate model is also used to provide a global explanation of the original model [32].

LIMEtree instead considers the case when the local behaviour cannot be modelled with a linear substitute model and uses a regression tree as its surrogate model [26]. Because of the local regression tree, LIMEtree can consider multiple classes for the local samples at the same time, rather than fitting a separate one-vs-rest linear model for each class as standard LIME does [26]. Anchor [27] is another explainable AI method developed from the concern of LIME's linear nature. They use a set of if-then rules as the surrogate model and explanation, which besides being able to capture non-linear local behaviour also benefits from being more intuitive for users [27].

Another downside of LIME is that it has a high computational cost because of the many perturbations necessary. STREAK [24] is an explainable AI method that is inspired by LIME but reports a much faster runtime.

2.3.1.2 Shapley Additive Explanations (SHAP)

Shapley values are found in game theory to decide how to fairly distribute a payout based on each person's contribution to a coalitional game [41]. In the explanation technique SHAP, this is viewed as the input features' contribution to the prediction [31].

The paper presenting SHAP [31] defines additive feature attribution models as models where the sum of the feature's contribution score matches the original prediction:

$$f(x) \simeq g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i z_i', \tag{2.1}$$

where $f(x)$ is the prediction of the original input x, $g(x')$ is the local approximation of the simplified input x', ϕ_i is the contribution score of each simplified input feature and ϕ_0 a bias contribution, M the number of simplified input features and $z_i' \in \{0,1\}$ indicates whether feature i is present (1) or absent (0) in the simplified input [31]. The simplified input x' is the interpretable input that the explanation should relate to, such as a set of super pixels for an image where the input x to the prediction model is a multidimensional array of three colour channels per pixel [9].

They note that some existing models like LIME [9] or DeepLIFT [35] (see Section 2.3.1.3) adhere to Definition 2.1 and so are additive feature attribution models. They define SHAP as the unique additive feature attribution model that satisfies three desirable properties: local accuracy, missingness and consistency. This results in explanations consisting of the Shapley values. To find the explanations, SHAP perturbs the input locally and observes how the model's prediction changes, calculating the SHAP values in the process. How the SHAP values are approximated depends on which SHAP model is used [31]. The local explanation provided by SHAP can be summarised to a global explanation [31].

SHAP exists as a model-agnostic version and in various model-specific versions including LinearSHAP [31] for linear models, DeepSHAP [31] for neural networks, TransSHAP [38] for transformer-based neural networks and GradSHAP [39] which is based on integrated gradients.

2.3.1.3 Deep Learning Important FeaTures (DeepLIFT)

DeepLIFT [35] was designed for explaining neural networks. In DeepLIFT, a reference input is chosen by the user to represent typical background values for the input features. The reference state is defined as the prediction of the reference input. From the reference state, the difference is backpropagated through all the neurons as the difference between the original and reference activations back to the input layer, resulting in the contribution scores of the input features [35].

2.3.1.4 Integrated Gradients

Integrated gradients [34] is a local explanation method for deep neural networks that satisfies the two axioms defined in the paper: sensitivity and implementation invariance. It is a generalisation of DeepLIFT designed to satisfy implementation invariance [34]. Integrated gradients are computed by first identifying the straight line path between the baseline input (for instance, a black image or a zero vector) and the input to be explained. The integral is then approximated by summing the gradients at sufficiently small steps along the straight line [34]. As this requires evaluating many inputs and extracting many gradients, it is computationally expensive [42].

2.3.1.5 Example-based explanation techniques

Example-based explanation techniques select representative instances or produce new instances that explain the model's decisions, making it closely related to the global explanation methods. The example instances that represent the model's predictions are called prototypes and some example-based methods also create criticisms, instances that are poorly represented by the prototypes. Example-based methods include MMD-critic [23], MAPLE [28] and DICE [29]. Example-based methods are not the focus of this thesis as they make less sense for amino acid sequences.

2.3.1.6 DeepCover

Sun et al. [8] developed the image explanation technique DeepCover that is the focus of this thesis. They define their explanations as the minimum subset of features needed for the model to make the same prediction as with the original input. To decide which input features to include in the explanations, they first create a test suite by randomly masking pixels in images by setting them to a background colour. The number of masked pixels is dynamic and depends on whether the masking changed the model's prediction. For each pixel of the input, they calculate the number of times the pixel was masked or not masked and whether the prediction of the image changed. Using these counts and statistical fault localisation measures, they rank the pixels based on importance. The minimal explanations are constructed by adding pixels to a blank image in the ranking order until the model reaches the original prediction.

Images and Proteins as Sequences. When used as input for machine learning models, pictures are often simplified to sequences of pixels. That is why some explanation techniques used for image classification can be adapted for sequence domains such as natural language or proteins. The goal of this project was to adapt the DeepCover tool for peptide presentation prediction. Practical reasons why DeepCover was chosen for this project are the accessibility of the tool, the code is available and a tutorial on how to get started and run the tool was provided, the adaptions were feasible in the time frame of the project, and because it is a recent and relevant technology.

Statistical Fault Localisation Measures. DeepCover uses Statistical Fault Localisation (SFL) approaches to rank positions, pixels, of the input [8]. SFL is a technique from software testing that ranks program elements such as statements or assignments by how likely they are to contain a sought-after bug. It does this by using statistical measures and how frequently each program element appears in passing or failing test executions.

The SFL measures listed below use the parameters $(a^s_{ep}, a^s_{ef}, a^s_{np}, a^s_{nf})$ which represent the number of times a program element s was executed (e), not executed (n) in passing (p) and failing (f) test executions [8]. In DeepCover, "passing" was instead defined as the mutated instance being predicted as the original label, where original label was the one predicted by the model for the original input instance and not necessarily the true label. "Failing" means that the mutation caused the label to change. For executions, DeepCover instead considers whether the single pixel s of the input was mutated or not. A pixel s of the test input was considered “not executed” if it was masked and “executed” if the pixel was the same as in the original instance [8].

There are many options for SFL measure [43]. The SFL measures used for this project were those that were already chosen and implemented for the DeepCover tool. These were Tarantula, Ochiai, Wong-II and Zoltar [8]:

$$\begin{aligned} &Tarantula : \frac{\frac{a^s_{ef}}{a^s_{ef}+a^s_{nf}}}{\frac{a^s_{ef}}{a^s_{ef}+a^s_{nf}}+\frac{a^s_{ep}}{a^s_{ep}+a^s_{np}}} \qquad &&Ochiai : \frac{a^s_{ef}}{\sqrt{(a^s_{ef}+a^s_{nf})(a^s_{ef}+a^s_{ep})}} \\ &Wong-II : a^s_{ef}-a^s_{ep} \qquad &&Zoltar : \frac{a^s_{ef}}{a^s_{ef}+a^s_{nf}+a^s_{ep}+\frac{1000a^s_{nf}a^s_{ep}}{a^s_{ef}}} \end{aligned} \tag{2.2}$$

Minimal Subset Explanations. Sun et al. [8] describe a good explanation as sufficient, minimal and not obvious. The explanations of images and peptide presentation are both not obvious as they are open problems and we can imagine multiple different possible explanations. DeepCover returns a full explanation that consists of an importance score for each position of the input. The minimal explanation algorithm proposed by Sun et al. [8] attempts to reduce the full explanation to a minimal and sufficient explanation. The algorithm works by first masking all input positions to the background colour. It then unmasks pixels in the order of importance given by the full explanation until the original prediction is achieved, fulfilling the sufficiency criterion. It is considered minimal because it stops unmasking positions as soon as the original prediction is recovered. Because the method does not try all possible combinations of minimal explanations, it is not ensured to be the smallest possible explanation that satisfies the constraint of resulting in the original prediction. However, it is likely close to minimal because the positions are added in order of importance.

DC-Causal. The creators of DeepCover (DC) extended their work to create DC-Causal, an explanation method based in causal theory which performs better on occluded images [44]. The algorithm splits the image into rectangular super-pixels and calculates the responsibility of each super-pixel. Mutants of the instances are created by masking different combinations of super-pixels. The responsibility of a super-pixel $P_{i,j}$ is defined as the

> ‘minimum difference between a mutant image and the original image over all mutant images x_m that do not mask $P_{i,j}$, are classified the same as the original image x, and masking $P_{i,j}$ in x_m changes the classification’ [44]

where the differences between the original images and its mutant is the number of super-pixels that were masked in the making of the mutant. DC-Causal then iteratively refines the super-pixels and calculates the responsibility for each further split until the partitions are small enough or all partitions have the same responsibility.

2.3.2 Explainable AI for NLP

The concept of basing explanations on how the predictions change when masking parts of the input has also been applied to natural language processing. Representation Erasure [40] focuses on global explanations which can relate to an input dimension, specific input words, or subphrases (the minimum subset of words that, if removed, changes the model's prediction). They compute the importance of a feature (word or dimension) as the sum of the relative difference in output when simply removing parts of the input representation.

Harbecke and Alt [30] argue that gradient-based explanations like Integrated Gradients are unsuitable for NLP tasks, as natural language follows a discrete distribution. The previous perturbation methods often delete an input feature (often a word) or mask it with a nonsensical background value (such as [UNK]) which in the NLP domain results in sentences that would not naturally occur. Their method, occlusion and language models (OLM), substitutes input features with ones generated by a language model that depends on the distribution of the data set the input was sampled from. The explanation method then assigns a relevance score to each input feature as the difference between the prediction of the original input and the inputs with the feature resampled using the language model.

Minimal Contrastive Editing (MICE) [37] is a perturbation and example-based explanation method especially developed for NLP models. They describe contrastive explanations as answers to questions in the form ”Why p and not q?”, why event p happened instead of another event q where p is the prediction and q is called the contrast case. This produces for instance the contrastive explanation ”This movie is bad” which is classified as the contrast case negative sentiment for the input ”This movie is great!” classified as a positive sentiment. They aim for the edits and resulting explanations to be fluent (meaning that they should be text natural for the domain) and minimal. In the case of amino acid sequences, contrastive examples would be less meaningful than in natural language tasks.

2.3.2.1 Attention-based Explanations

Attention plays an important role in transformers. Attention as explanations is a controversial topic [45, 46], but they can still be used to interpret a model. Transformer models pass the input through many layers, which combine the features of the previous layer using self-attention. After repeatedly combining the features, the attention scores at a given layer cannot be associated to specific features of the input, except for the attention scores of the input layer. Abnar and Zuidema [36] propose two methods to compute the attention scores of the input features from any layer by representing the features at each layer as nodes in a directed acyclic graph where the edges are the attention going between the features of each of the layers. They define attention rollout as the sum of the scores given to each path between an input feature and the feature in the layer of interest, where the score of a path is the product of its edges. They define a second method, attention flow, as the sum of the minimum capacity of the links between an input feature and the node/feature in the layer of interest.

2.3.3 Explainable AI for biology

While there are plenty of explainable AI techniques for images and NLP tasks, the field is not as advanced in the biological AI domain. A few recent attempts of developing explainable AI methods include PoSHAP [33], ALPODS [22] and iRF-LOOP [21].

Dickinson and Meyer [33] applied SHAP to an MHC-I binding affinity prediction LSTM model in a method they call Positional SHAP (PoSHAP) in an attempt to produce explanations that show the impact of positions in the input sequences.

ALPODS is an explainable AI system for diagnosis based on flow cytometry results, which has high dimensions with more than 100 000 cells and 10 or more features. ALPODS clusters the data into subpopulations and creates rules for each subpopulation. The rules for the subpopulations are combined (by fuzzy conjunction) to a decision for the input which gives the diagnosis [22].

Iterative random forest (iRF) [20] builds on the idea that the location of a feature in a decision tree describes its feature importance. They argue that the importance of chosen features when building the decision trees are conditional on the previously chosen features, which accounts for interconnected dependencies common in biological settings. iRF iteratively creates weighted random forests after weighting the input features by the feature importance from the previous forest [20]. iRF Leave One Out Prediction (iRF-LOOP) [21] expands on the idea of iRF to create a matrix to represent how the input features are related. iRF-LOOP applies iRF to get the importance of all features for predicting an input feature. Doing this n times for the n features of the input creates a n x n importance matrix which when normalised can be interpreted as a directional adjacency matrix defining which features most implement each of the other features. For instance, Garvin et al. applied iRF-LOOP and Random Intersection Trees (RIT) to SARS-CoV-2 sequences to find the relationship between mutations in genetic material [47].

Attention has also been considered as explanations in ML models applied to proteins [48]. They look at the attention in transformer models applied to protein datasets. They found that attention (1) captures the three-dimensional relationships as it connects amino acids that are far apart in the amino acid sequence but close together in the protein's three-dimensional structure, (2) targets binding sites and (3) captures increasingly complex representations of structure and function with increasing layer depth.

The papers presenting ImmunoBERT [4, 5] also put a focus on the interpretability of their model. Because the model is a complex black-box transformer model, they apply the common post hoc explanation methods LIME and SHAP.

Chapter 3

Methods

As mentioned in Section 1.1, the aim of this project was to use DeepCover to interpret black-box presentation prediction model ImmunoBERT. This section outlines how DeepCover was modified to create explanations for the ImmunoBERT model. Modified aspects of DeepCover were:

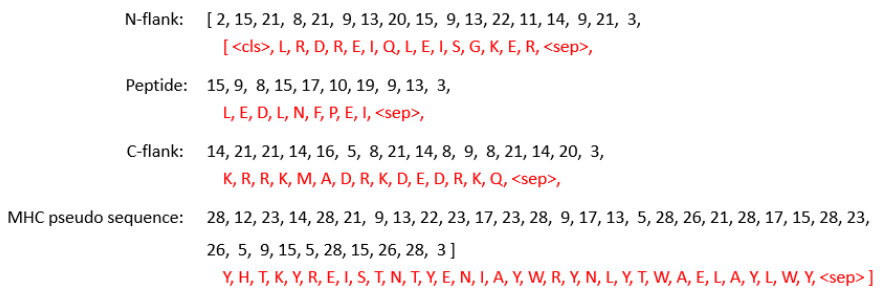
- DeepCover was developed by the original authors for explaining image models [8]. To adapt the explanation tool to ImmunoBERT, DeepCover was modified in this project to handle the amino acid sequence input rather than the expected images.
- The original DeepCover creates mutations by masking super-pixels with a given background colour. As mutations occur naturally in genetic code, DeepCover was adapted to simulate natural mutations when mutating the amino acid input.
- Parallelisation of the execution of DeepCover by adding options to split the input instances into batches to run the batches in parallel.

Unchanged aspects of DeepCover that are explained in Section 2.3.1.6 include:

- The fault localisation metrics Tarantula, Zoltar, Ochiai and Wong-II implemented in the DeepCover tool.
- The minimal subset explanation algorithm to use the full explanations produced by DeepCover to select a minimal subset of the original input sequence as an alternative explanation.

3.1 Overview of Input

DeepCover was originally developed for image-based explanations and such the expected inputs were 2- or 3-dimensional arrays, depending on the number of colour channels. The main changes implemented in this project were related to the format of the input data and how the input was mutated. To understand the changes, it is helpful to first look at the format of the amino-acid instances of the dataset.



```
N-flank:                [ 2, 15, 21,  8, 21,  9, 13, 20, 15,  9, 13, 22, 11, 14,  9, 21,  3,
                          [ <cls>, L, R, D, R, E, I, Q, L, E, I, S, G, K, E, R, <sep>,

Peptide:                15, 9,  8, 15, 17, 10, 19,  9, 13,  3,
                          L, E, D, L, N, F, P, E, I, <sep>,

C-flank:                14, 21, 21, 14, 16,  5,  8, 21, 14, 8,  9,  8, 21, 14, 20,  3,
                          K, R, R, K, M, A, D, R, K, D, E, D, R, K, Q, <sep>,

MHC pseudo sequence:    28, 12, 23, 14, 28, 21,  9, 13, 22, 23, 17, 23, 28,  9, 17, 13,  5, 28, 26, 21, 28, 17, 15, 28, 23,
                        26,  5,  9, 15, 5, 28, 15, 26, 28,  3 ]
                          Y, H, T, K, Y, R, E, I, S, T, N, T, Y, E, N, I, A, Y, W, R, Y, N, L, Y, T, W, A, E, L, A, Y, L, W, Y, <sep> ]
```

Figure 3.1: Example of the token embedding of an instance. The black array is the token embedding part of the ImmunoBERT input and the red array below it shows the origin amino acid sequences and special tokens <cls> for sentence-level representation and <sep> to separate the sequences. The left-hand column shows the different parts of the input, encoded in the segment embedding. "Positions" of an input instance refer to the indices in the token embeddings.

The input instances consisted of multiple amino acid sequences to represent the peptide, flanks (the context of the peptide in the origin protein) and MHC-I pseudo sequences. The sequences were encoded as an array of integers representing the amino acids in the sequences. To match the input format required by BERT models, the input sequences started with a classification token [CLS] and the peptide, flanks and MHC pseudo sequence were separated by [SEP] tokens, see Figure 3.1. The input was then embedded as token, segment and positional embeddings, as is standard with BERT. The embedding also contained the target label and an input mask to force the model to ignore some parts of the input. A more detailed example of the input embeddings can be found in the appendix of the ImmunoBERT paper [4].

3.2 Modifying DeepCover

DeepCover in the ImmunoBERT Context. The parameters $(a^s_{ep}, a^s_{ef}, a^s_{np}, a^s_{nf})$ used in the fault localisation measures were gathered on an amino-acid level of each instance to be explained, with one set of parameters produced for each position (in the token embedding presented in Figure 3.1) of each input instance. The four parameters were all counts of events which were counted over a test suite created by mutating the original instance. Calculating the SFL measures was again done for each position of each input instance, giving a full explanation consisting of importance scores for each instance of the same size as the instance.

Parts of Input to Mutate. Some parts of the inputs should be excluded from the mutations. For instance, the non-amino acid tokens [CLS] and [SEP] were excluded from the mutations and flags were added to DeepCover to signal which parts of the amino acid data to mutate, with options for the peptide, MHC pseudo sequence and the flanks.

Shape of Mutations. Another change to how DeepCover handled the ImmunoBERT input was the shape of the mutated regions. In the original DeepCover application, the input images were perturbed by masking super-pixels. Practically, that translated to masking rectangular subsets of pixels but ranking each individual pixel based on the

SFLs and how many times each pixel was included in super-pixels. However, the input to ImmunoBERT was one dimensional and short. Therefore, the mutated regions for the ImmunoBERT input were non-continuous.

The upcoming sections further describe the modification to mutations using substitution matrices and the parallelisation of DeepCover.

3.2.1 Domain-specific perturbations

A concern about LIME and random perturbations is the risk that the mutations result in nonsensical instances. To minimise that concern, the mutations used by DeepCover to explain the amino acid input was designed to simulate real substitution mutations that happen in genetic code. This was done using BLOSUM (BLOcks SUbstitution Matrix), which is a substitution matrix with scores of how likely a substitution mutation is in the genetic code for each pair of amino acids.

Previous research investigating NetMHCpan, an established and popular MHC-peptide binding model, has suggested that the choice of substitution mutation in the MHC section of the input has a big effect on the set of peptides that are predicted as binding to the MHC molecule [13]. They first evaluated the ability of NetMHCpan to predict a peptide-binding repertoire (the set of peptides that bind to a given MHC protein) which was already known from the literature to ensure that the model was reliable. They then used BLOSUM to perform single substitution mutations. They found that the size of the overlap between the original peptide-binding repertoire and that predicted after mutations was larger for those mutated with amino acids with higher BLOSUM scores, those being more similar to the amino acids in the position before mutating. They concluded that this means that the substitutions that are less likely to occur, because they involved amino acids that have different physiochemical properties, change the peptide-binding repertoire the most.

In the original DeepCover tool for images, mutated images were created by masking super-pixels with a background colour. Mutations happen naturally in genetic material and proteins. The mutations of DeepCover can thus naturally be extended to peptides using substitution matrices that reflect the rate of substitution mutations in proteins. When mutating our protein inputs, the randomly chosen amino acids were therefore replaced by the most similar amino acids from the BLOSUM substitution matrix. More likely substitution mutations tend to be between physicochemically similar amino acids. For instance, they might have similar size or polarity which affects how hydrophobic or hydrophilic an amino acid is [13]. We used the highest scoring BLOSUM substitutions to mutate the input to get as realistic mutations as possible.

3.2.2 Parallelisation

Command line arguments were added to DeepCover which specified batch number and batch size. These flags were used to index into the set of input instances to select a subset of instances to produce explanations for. A bash script with GNU parallel [49] was then written to iterate over the input split up into batches using the flags and run the batches in parallel.

Chapter 4

Experiments

The aim of the experiments was to establish how well the modified DeepCover could produce explanations for the ImmunoBERT model's decisions. The following evaluations were considered in the experiments:

Part 1: Performance of DeepCover in Isolation. The first evaluation concerned the performance of DeepCover in isolation. The metrics considered were:

- **Coverage of Explanations.** The coverage of explanations was interpreted as how many instances DeepCover could produce explanations for.
- **Biological Quality of Explanations.** The quality of the explanations produced by DeepCover compared to known biological properties of the MHC molecules and peptides. Biological properties include which positions are known to be important for binding between the MHC molecule and peptide in general and for specific alleles of the MHC molecule.
- **Agreement of Fault Localisation Metrics.** Comparing the explanations DeepCover produced when using the four different fault localisation measures included in the tool: Tarantula, Ochiai, Wong-II and Zoltar.
- **Frequency and Size of Minimal Explanations** How well the minimal explanations worked for our protein application, evaluated by how often minimal explanations could be produced, the size of the minimal explanations and whether the positions included in the minimal explanations agreed with the full explanations.

Part 2: Comparison Against LIME and DC-Causal. The second evaluation was a comparison of the explanations produced by DeepCover against two other perturbation-based explanation methods. The other explanation methods were DC-Causal, which is also a part of the DeepCover tool, and LIME. The comparison was performed on two levels, comparing the rankings across instances and per instance:

- **Position Rankings Across Instances.** Across instances, the rankings produced by each explanation method were plotted and the trends in which positions were deemed the most important by the different methods compared.
- **Position Rankings per Instance.** All three explanation methods are local explanation methods, producing one explanation per instance. The second level of

comparisons focused on a statistical comparison of the instance-level explanations. The instance-level comparison also included the explanation coverage of DC-Causal and LIME.

Part 3: Runtime Efficiency. The third evaluation regarded efficiency and involved measuring the runtime of DeepCover, comparing it to the runtime of LIME and evaluating the runtime improvement from the batch parallelisation applied to DeepCover.

The following sections describe the experiment setup to perform these evaluations.

4.1 Data

The explanation experiments were run for the 12 MHC-I alleles included in the test set, provided and split by Gasser [4]. In the experiments, DeepCover was run on 500 positive instances and 500 decoy instances for each MHC allele to mirror the explainability experiments performed by Gasser [4, 5]. The ImmunoBERT model was built with peptides of length 7-15 amino acids [4]. To simplify creating aggregate results, all instances chosen for this project had peptides of length 9 amino acids, which is one of the preferred peptide lengths for binding to the MHC molecules [50]. All the MHC pseudo sequences were of the same length in this set (34 amino acids), but the flanks varied in existence and size. The varying size and existence of the flanks was not a problem as they were included when the instances were given to the model for prediction, but were not mutated during the experiments or included in the explanations.

4.2 ImmunoBERT Model

Hyperparameters. The ImmunoBERT model used for the experiments was fine-tuned as part of the original ImmunoBERT paper [4]. No training or fine-tuning was performed in this project. It was fine-tuned by Gasser [4] for 5 epochs with learning rate 1e-05, 19 decoys per positive instance and using the classification token's output as the pooling, all decided through Gasser's experiments. No further hyperparameters were set for the experiments, besides turning the presentation score into a classification.

Classification. ImmunoBERT predicts a presentation score, representing how likely it was that the given MHC-I allele presented the given peptide. The score was passed through a sigmoid function, as in Gasser's code [51], to map the score to the range [0,1]. Unlike LIME and SHAP used by Gasser [4, 5], DeepCover needed a class label prediction, so the score was converted to a class label using a threshold of 0.5 to evenly divide the range into the two classes. This made a presentation score of 0.5 or higher the positive class "presented" and lower than 0.5 the negative class "not presented".

4.3 DeepCover

DeepCover Parameters. DeepCover created one explanation per instance by generating and computing the SFL measures on one test set per instance. The test set was created by mutating the instance 2000 times, resulting in a test set of 2000 mutated instances.

Only the peptide and MHC parts of the input were mutated as Gasser already concluded that the flanks were the least important parts of the input [4, 5]. The number of mutations were 1, 2 or 3 positions in non-continuous regions and randomly chosen. The experiments were run for all four SFL measures, Tarantula, Ochiai, Wong-II and Zolta, for 1000 instances for all 12 MHC-I alleles.

Minimal Subset Explanations. The minimal subset explanations for the ImmunoBERT explanations were produced using the minimal subset explanation algorithm described in Section 2.3.1.6 in the related work covering the DeepCover tool. When applying the minimal explanation algorithm to ImmunoBERT, positions were masked by setting the corresponding positions of the input mask to 0. The positions were then unmasked in the order of the full explanation given by DeepCover using the Tarantula SFL until the original prediction was recovered, producing an explanation that consisted of a subset of the original input sequence.

4.3.1 Parallel Execution Setting

The experiments were parallelised with a GNU parallel bash script [49]. The new batch arguments were used to split the instances into 25 batches of size 40 instances during the experiments. Eight batches were run in parallel on a GPU server where they were assigned one GPU per running DeepCover batch.

4.4 LIME

LIME is a commonly used explainable AI method. The objective of comparing the explanations produced by DeepCover to those produced by LIME was to decide whether they uncover the same positions as important. While complete agreement was not the the goal, some similarity between the explanations from the two methods can indicate truly important positions

Across-Instance Comparison. The comparison with LIME was done on two levels, on an across-instance level per allele and on a local per-instance level. For the across-instance view, the figures had already been created by Gasser [4]. They included rankings of the positions of the peptide, the flanks of the peptide which were the 15 amino acids on either side of the peptides in the original protein, before it was sliced into peptides, and the MHC pseudo sequence.

Instance-wise Comparison. For the local, instance-wise comparison, the LIME explanations had to be recreated and so there was an opportunity to design the experiment setup. As the flanks were deemed the least important in the original ImmunoBERT work [4] and DeepCover was used to produce explanations for only the peptide and MHC part of the input, the LIME explanations produced for the instance-wise comparison included only the peptide and MHC parts of the input. LIME explanations were produced for the same 1000 instances for each allele as DeepCover's explanations and was based on 2000 mutations per instance.

4.5 DC-Causal

DC-Causal was developed by the authors of DeepCover [44] and is part of the Deep-Cover tool. DC-Causal is a perturbation-based explanation technique which assigns an importance to an iteratively refined subset of input positions, see the original paper [44] or Section 2.3.1.6. The explanations produced by DC-Causal were obtained from another student at the University of Edinburgh (Linda Mazánová) who modified DC-Causal for explaining ImmunoBERT's peptide presentation decisions. It was run on the same 1000 instances per allele as DeepCover with SFL and LIME and the explanations were included in the instance-wise comparison with DeepCover with SFL.

4.6 Rank Biased Overlap for Instance-wise Comparisons

Rank Biased Overlap (RBO) is a measure to compare ranked lists [52]. RBO scores are in the range [0,1], where 0 means completely disjoint and 1 means identical ranked lists. Items that are ranked higher contribute more to the RBO score and it allows us to calculate how much weight was given to the top items.

The RBO score was used to compare the explanations produced by different methods instance-wise. The instance-wise comparison was performed for the different SFL measures implemented in DeepCover and to compare DeepCover to other explainable AI techniques LIME and DC-Causal. The RBO score was computed where both methods produced an explanation for the instance. The score was calculated on the full explanation containing the peptide and MHC ranking but also on just the rankings of the peptide positions, as these are the most important regions of the inputs [4, 5].

When computing the RBO score of the MHC and peptide combined, the parameter p, which weights the top positions in the lists higher, was set to 0.9. This assigned 86% of the score to the top 10 items of the MHC and peptide combined. This setup was chosen because it assigned a clear majority of the weight to the number of top positions that could accommodate the whole peptide, with a small margin, which tends to have the most important positions. During the peptide-only comparison, p was set to 0.55 to enable comparison between the scores: the same percentage of the weight was given to the top 2 items, which makes up the same fraction as the top 10 positions for the combined rankings.

4.7 Runtime

The final experiment measured the runtime of LIME and DeepCover when producing explanations. Both explanation methods were run for 50 instances on an interactive SLURM session with 4 CPUs and 1 GPU for performing the predictions. The per-instance runtime was then computed as the average over the 50 input instance.

The execution of DeepCover was parallelised using GNU's parallel [49] and batches, as described in the methods (Chapter 3). To include this "real-life" use of DeepCover, a runtime experiment with 8 parallel batches was run for the 50 instances. It was run as a SLURM batch job with one CPU and one GPU per parallel batch.

Chapter 5

Results

This chapter presents the results of the experiments. The presentation starts with evaluation part 1 concerning the coverage and biology of the explanations produced by DeepCover. This is followed by part 2 evaluating how the explanations compare to other explanation techniques and finally there is a short runtime evaluation for part 3.

5.1 Explanations Produced by DeepCover

The figures presenting the across-instance views of the explanations produced by DeepCover were formatted to correspond to figures created by Gasser [4] to facilitate comparison. DeepCover ranks the positions of the input for each instance. The figures show how often each position of the peptide and MHC input was ranked in different ranking positions. DeepCover was implemented with four fault localisation measures, Tarantula, Ochiai, Wong-II and Zoltar. The analysis first focuses on Tarantula and then compares the results against the other measures, because of their similarity.

5.1.1 Across-Instance View and Biological Interpretation

The evaluation of DeepCover using Tarantula SFL follows the first two parts of evaluation part 1 outlined in Chapter 4: the coverage of DeepCover explanations and the quality of the produced explanations compared to biological properties. The biological properties refer to the topics outlined in the biological background in Section 2.1.

Because changes in the MHC molecule largely impacts which peptides it will present, the analysis and modelling of peptide presentation is customarily done on a per-allele basis. The original project by Gasser [4] focused on the explanation of three representative alleles: HLA-A*33:01, HLA-B*54:01 and HLA-C*01:02 and presented the rest in the appendix. Where there is not enough space to present the results from all alleles in the main body of this report, the comparison will be presented in the same way here and in upcoming sections of the results.

5.1.1.1 Coverage

Coverage. Table 5.1 shows the coverage of the explanations produced by DeepCover. DeepCover was given 1000 instances (500 positive and 500 negative) per allele to explain. As presented in Table 5.1, DeepCover failed to explain the majority of instances for all alleles except HLA-A*33:03 which had explanations for just over 50% of the 1000 instances. From Table 5.1, it is clear that most of the explained images were the positive instances while very few of the decoys were explained. DeepCover relies on the predicted label switching while mutating to be able to compute the SFL measures. The imbalance between the number of positive and decoy instances that were explained was likely caused by the positive instances being easier to switch to the opposite label than negative instances.

Table 5.1: The coverage of DeepCover's explanations, as a percentage of the 1000 instances per MHC-I allele and for the instances labelled positive and negative.

Allele	Explanations Produced by DeepCover	Percentage Explained	Presented Instances Explained	Decoys Explained
HLA-A*33:01	458	45.8%	452	6
HLA-A*33:03	505	50.5%	476	29
HLA-A*36:01	410	41.0%	384	26
HLA-A*74:01	495	49.5%	475	20
HLA-B*37:01	206	20.6%	188	18
HLA-B*46:01	246	24.6%	224	22
HLA-B*54:01	465	46.5%	426	39
HLA-B*58:01	382	38.2%	343	39
HLA-B*58:02	340	34.0%	319	21
HLA-C*01-02	445	44.5%	417	28
HLA-C*15-02	291	29.1%	269	22
HLA-C*17-01	467	46.7%	436	31

Binary Class Issue. The main causes of the difficulties of switching the labels through mutations were the binary class nature of the model, an ill-chosen hard threshold and ImmunoBERT being robust. ImmunoBERT being robust means that the model itself was not swayed by perturbations in the input instances. The experiments performed as part of the original DeepCover paper involved many classes [8]. When applied to a binary class problem, there is less instability as there is only one other class to switch to and switching the label becomes harder.

5.1.1.2 HLA-A*33:01

The across-instance results for allele HLA-A*33:01 are displayed in Figure 5.1. They show that the positions of the peptide are generally ranked higher than the MHC positions (beware of the colour difference between the peptide and MHC sections in the figures). The most important positions were the 2nd and 9th position of the peptide and the 63rd of the MHC pseudo sequence.

Peptide Importance. The most important position in the peptide and in the whole input for MHC allele HLA-A*33:01 was position 9, being ranked as the most important

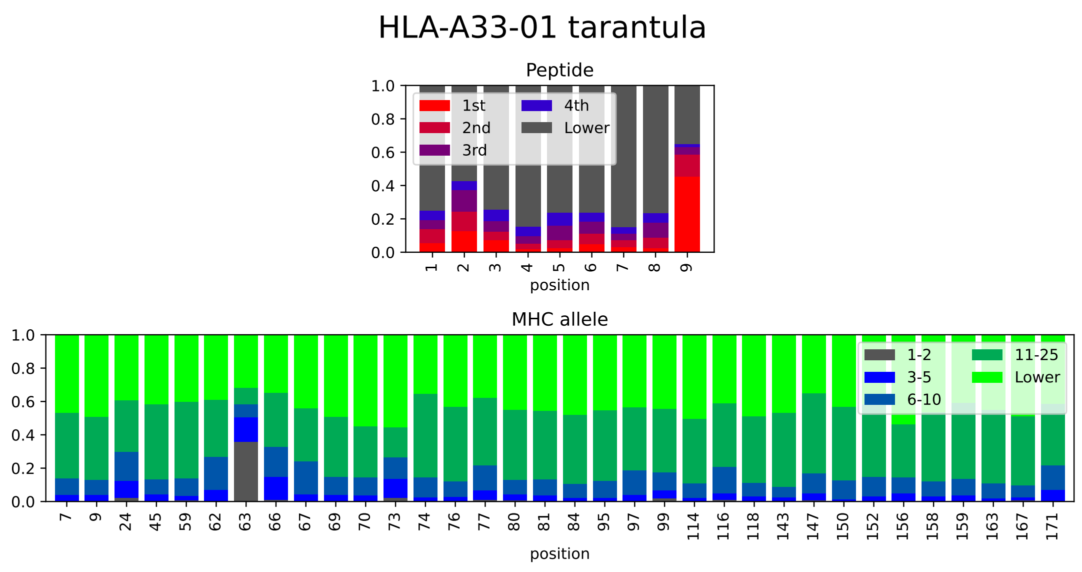


Figure 5.1: Across-instance view of the rankings produced by DeepCover with Tarantula SFL for allele HLA-A*33:01 for the peptide (top) and the MHC pseudo sequence (bottom).

position in almost half of the cases. There was also a spike at position 2 of the peptide. These two positions are likely the anchor residues, the amino acids on either side of the peptide that bind to the groove of the MHC molecules, discussed more thoroughly for allele HLA-B*54:01 in Section 5.1.1.3.

MHC Importance. The most important position in the MHC input was position 63, being ranked in the top 2 in almost 40% of the instances. In importance, this position is comparable to position 2 of the peptide. Interestingly, position 63 in the MHC allele was the position whose peptide-binding repertoire changed the most upon single-amino acid substitution in previous experiments [13]. They concluded that this means that position 63 is crucial for peptide binding according to their model, NetMHCpan, as well [13]. Position 63 is located in pockets A and B [13], where pocket A likely houses the N-terminus (start of the peptide, left-hand side in the figures) of the peptide.

Other slightly more important positions in the MHC pseudo sequence according to DeepCover include positions 24 in pocket B, 66 in pockets A and B, 73 in pocket C and 171 in pocket A [13]. Largely, being in the same pockets as position 63, they might play a supporting role in binding the peptide. Pocket F usually houses the C-terminus of the peptide. The most important position in pocket F is 77 but it does not stand out. There is not enough clarity in these results to conclude where the C-terminus (end of peptide, right-hand side in the figures) is housed in the MHC allele.

5.1.1.3 HLA-B*54:01

Peptide Importance. The rankings produced by DeepCover with Tarantula for allele HLA-B*54:01 are shown in Figure 5.2. Again, positions 2 and 9 of the peptide were the most important. However, in HLA-B*54:01, position 2 was more important than position 9 with position 2 being the most important in 40% of the cases while position 9 was the most important position in about 30% of the instances.

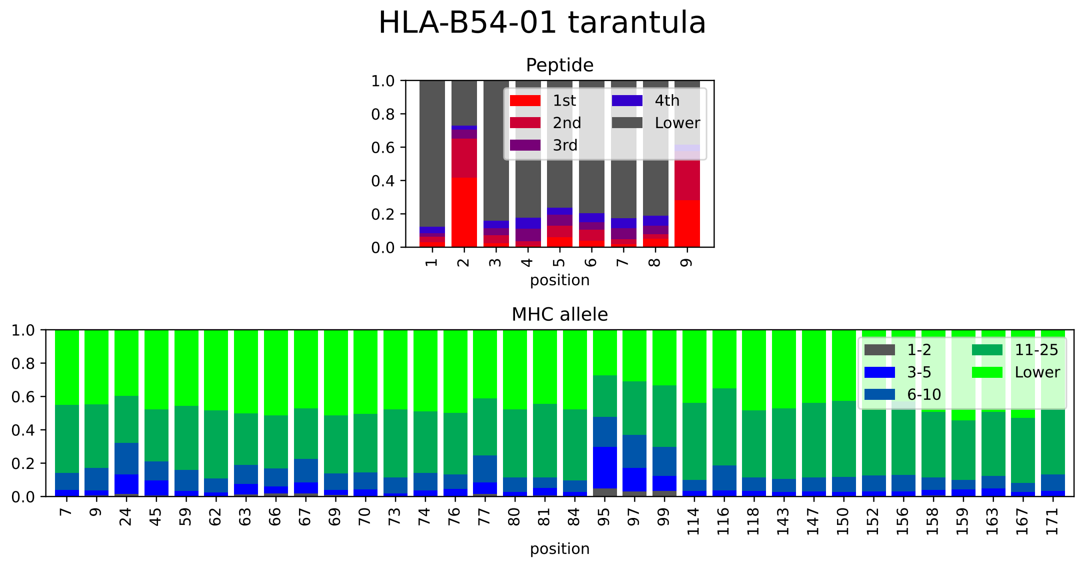


Figure 5.2: Across-instance view of the rankings produced by DeepCover with Tarantula SFL for allele HLA-B*54:01 for the peptide (top) and the MHC pseudo sequence (bottom).

For a few variations of HLA-B alleles, it has been experimentally shown that the C-terminus of the peptide was more conserved than the N-terminus, which was more flexible and varied in anchor position and amino acids [50]. They found that the anchor position of the C-terminus was mostly located at position 9 of the peptide and the amino acid sequence was more conserved at that position [50]. They also showed that position 2 of the peptide was a common N-terminus anchor site and also tended to have a distinct motif with less variation in amino acids than non-anchor positions of the peptide [50].

The rankings in Figure 5.2 agree that position 9 of the peptide was important. In terms of ImmunoBERT mutations, this means that the conservation of the C-terminus anchor residue has been captured by the model where mutating position 9 at the C-terminus causes the peptide to no longer be predicted as presented. The most important position in the N-terminus according to DeepCover was position 2, which was identified as a common anchor position by Prilliman et al. [50]. For this specific allele, HLA-B*54:01, the N-terminus housing pocket of the MHC molecule likely requires specific properties of the peptide to bind it which seems to include an inflexible N-proximal anchor residue. Similar properties were also present in HLA-A*33:01, which mainly highlighted position 9 but also somewhat position 2, indicating a more flexible N-terminus binding site.

MHC Importance. For the MHC pseudo sequence, the most important positions were 24, 95, 97 and 99. These positions are located in a variety of pockets. Position 24 is located in pocket B, 95 in pocket F, position 97 in pockets C and E and position 99 in pockets A, B and D [13].

The high importance of position 9 in the peptide is likely connected to position 95 which has the highest importance of the MHC molecule and is located in pocket F, which commonly houses the C-terminus. The N-proximal anchor site (position 2) of the peptide was also found to be very important. There have been indications that the

N-terminus can bind to pocket B of the MHC molecule rather than pocket A [5, 53]. This means that there are possible interactions between position 2 of the peptide and the important positions 24 and 99 of the MHC pseudo sequence, which are located in pocket B.

5.1.1.4 HLA-C*01:02

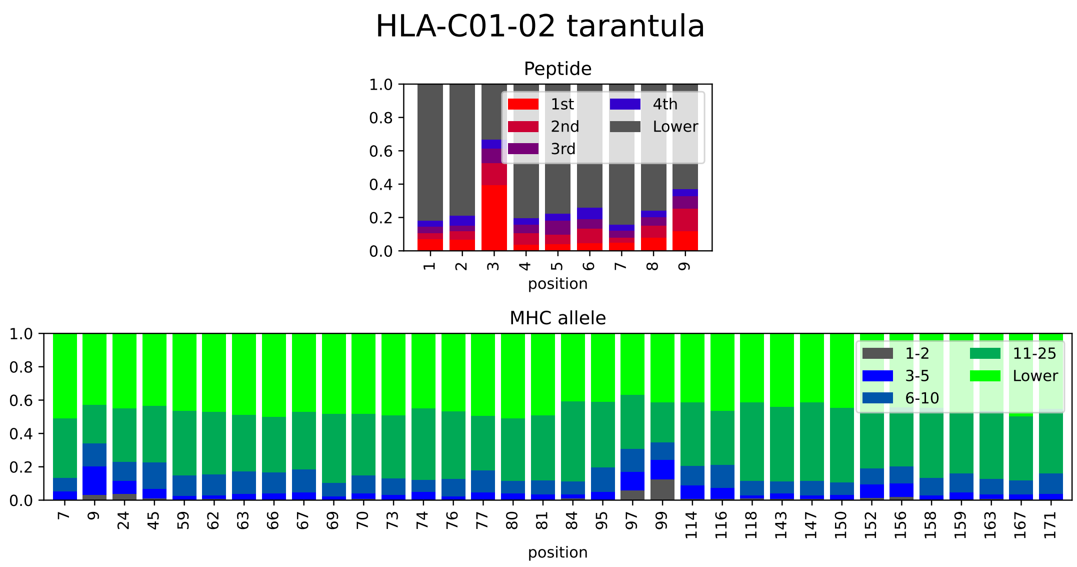


Figure 5.3: Across-instance view of the rankings produced by DeepCover with Tarantula SFL for allele HLA-C*01:02 for the peptide (top) and the MHC pseudo sequence (bottom).

Peptide Importance. The across-instance view of the rankings for HLA-C*01:02 by importance is shown in Figure 5.3. In HLA-C*01:02, the most important position of the peptide was position 3 with 40% of the highest rankings. Also position 9 was slightly more important than the other positions. Although explored for HLA-B alleles, Prilliman et al. [50] showed that position 3 can also be an N-proximal anchor position. Considering the importance assigned to position 3 by DeepCover, it is likely the anchor residue for HLA-C*01:02. The variation in most important position in the N-termini across alleles further supports the hypothesis of Prilliman et al. [50] that the position of the N-proximal anchor residue is flexible in these 9-length peptides. The slightly higher importance assigned to position 9 indicates that it again was the C-terminus anchor position, though it was more flexible to mutations than position 9 in the previous alleles.

MHC Importance. In the MHC part of the input, positions 9, 97 and 99 were the most important but still less important than most of the peptide positions. Position 9 is located in pockets B and C, 97 in pockets C and E and position 99 in pockets A, B and D [13]. The most important region was positions 97-99, which are adjacent in the pseudo sequence but located in different pockets. These are likely important in the binding as mutations in these locations led to changes in which peptides were predicted as presented but, being in different pockets, it is hard to draw conclusions about how they interact with the peptide. The flexibility of the C-terminus of the peptide might be connected to the less defined important regions in the MHC allele.

5.1.2 Comparison of Rankings Using Different Fault Localisation Measures

DeepCover implemented four fault localisation measures, Tarantula, Ochiai, Wong-II and Zoltar. The results from Tarantula have already been presented and this section will look at how the results produced by Tarantula compare to those produced by the other fault localisation measures.

5.1.2.1 Across-Instance Comparison

The aggregate figures produced using the other fault localisation measures for the first allele, HLA-A*33:01 are shown in Figures 5.4 - 5.6. Comparing the figures between themselves and to Tarantula in Figure 5.1, they produce very similar results with spikes at positions 2 and 9 in the peptide, a large spike at position 63 of the MHC pseudo sequence and smaller spikes at positions 24, 116 and 171. The other alleles (not included in the report) also look very similar between fault localisation measures.

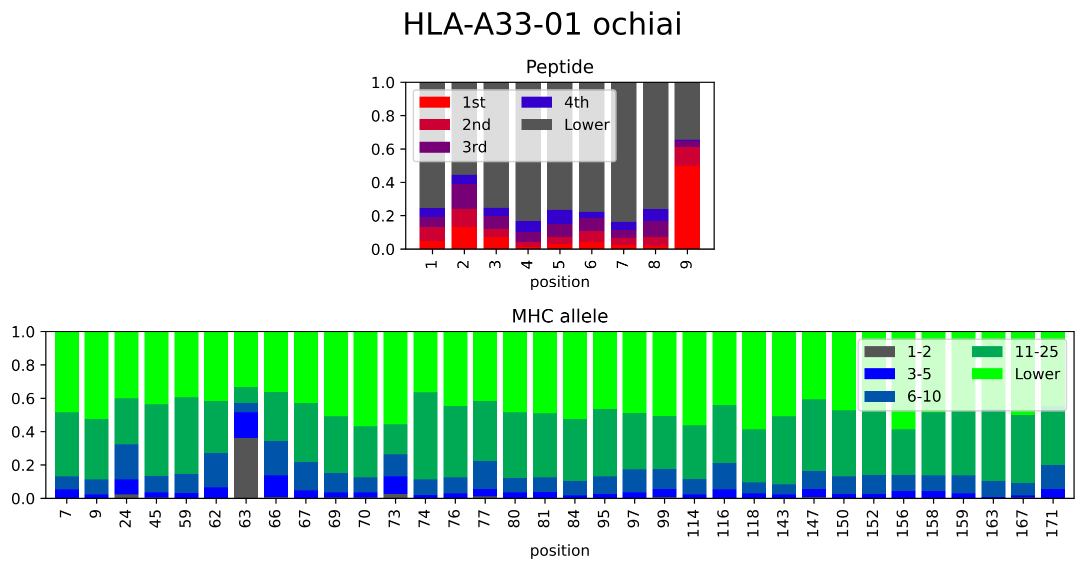


Figure 5.4: The rankings produced by Ochiai SFL for allele HLA-A*33:01.

5.1.2.2 Instance-wise Comparison

To fully evaluate how much the different SFL measures agree, an instance-wise comparison using RBO scores was performed. RBO is a measure that compares two ranked lists where 0 is disjoint and 1 is identical lists. Table 5.2 shows the average RBO scores for each pair of SFL measures.

Table 5.2: The average RBO scores between the four SFL measures across all 12 alleles.

	Ochiai	**Zoltar**	**Wong-II**
Tarantula	0.911	0.897	0.644
Ochiai	-	0.984	0.699
Zoltar	-	-	0.688

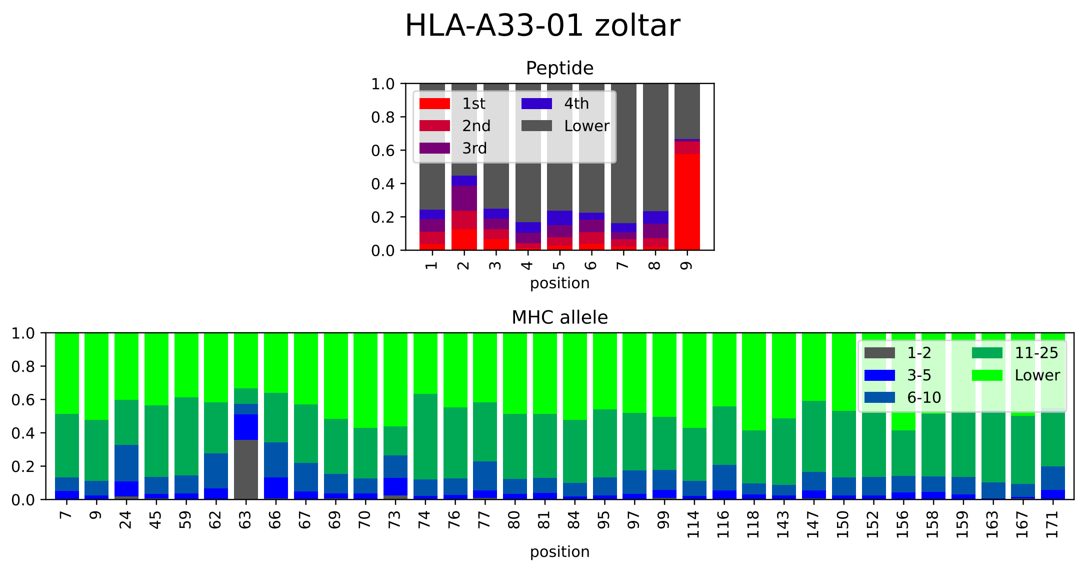


Figure 5.5: The rankings produced by Zoltar SFL for allele HLA-A*33:01.

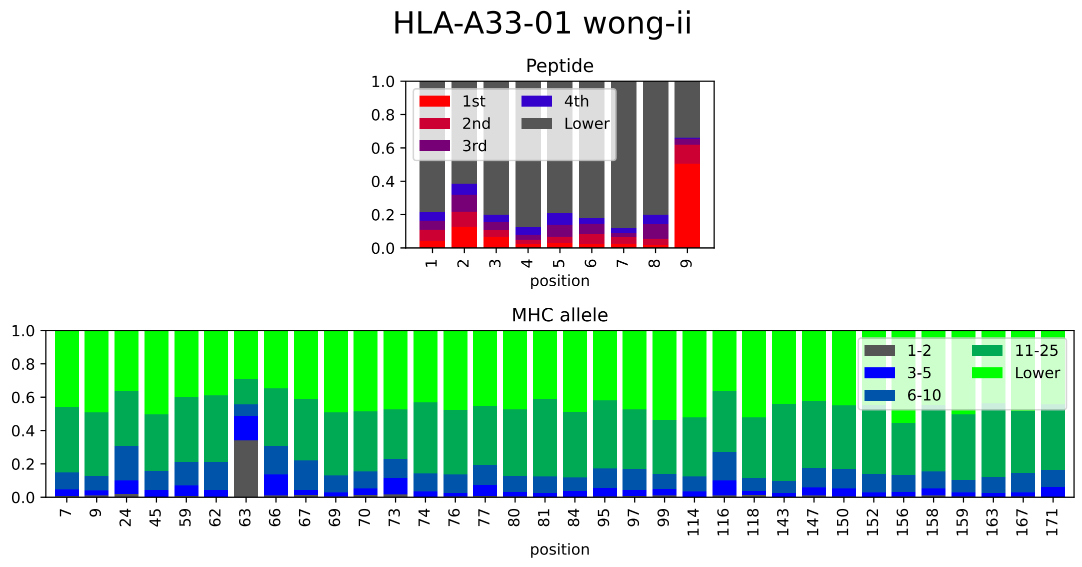


Figure 5.6: The rankings produced by Wong-II SFL for allele HLA-A*33:01.

From the instance-wise comparison in Table 5.2, we can see that the agreement between the measures varied between 0.644 and 0.984. Wong-II agreed the least with the other measures. While the disagreement of Wong-II is not as clear in the aggregate figures, it is apparent from the lower RBO scores. Wong-II is the only measure that does not scale the counts, see Equations 2.2. Additionally, it only considers the cases where a position was not mutated and simply subtracts the number of “passing” (where mutations in other positions preserved the original label) from the number of “failing” (where mutations in other positions caused the label to switch). The other SFL measures had a very high agreement because they are based on the same statistics and are all a scaled measure of a^s_{ef}, the number of times position s was not mutated but the predicted label was switched by mutations elsewhere in the sequence.

5.1.3 Minimal Subset Explanations

The final part of the evaluation of DeepCover in isolation was how well the minimal subset explanations introduced by Sun et al. [8] apply to the peptide presentation domain and the ImmunoBERT model. The idea of the minimal explanation is to use the full explanation to choose a subset of the original input instance that is sufficient to explain the original prediction.

Minimal Explanations for ImmunoBERT. The minimal explanations for ImmunoBERT were created by masking the entire input and adding positions in the order that they were ranked by DeepCover and Tarantula SFL in the full explanation until the original label was recovered. When masking the whole input, the peptides were predicted as not presenting by ImmunoBERT. As the minimal explanations depend on the label changing from the starting label, minimal explanations could only be produced for instances that were assigned the positive label, presented, in their original form. This is a consequence of the a binary class problem, presented or not presented by the MHC molecule. In the original DeepCover paper, it was intended for use on a multi-class problem where minimal explanations can be produced for many more labels [8].

Coverage. Table 5.3 shows how many of the 1000 instances per allele could be explained by DeepCover and how many of those resulted in a minimal explanation. The column “Percentage Created” shows how many of the DeepCover explanations lead to a minimal explanation, to clearly illustrate how many minimal explanation that could not be created because of the constraint on predicted labels. The percentage ranges from 38% to 86% showing that the problem is severe for some alleles and acceptable for some. The minimal explanations were produced for all of the instances that were predicted as positive in their original form, because of the reasons outlined in the previous paragraph. The table also presents the average size of the minimal explanations. This shows how many amino acids, on average, were needed by ImmunoBERT to be able to predict the original positive label and shows that the minimal explanations that were created were smaller than the full explanations.

Table 5.3: The number of explanations and minimal explanations that were produced by DeepCover for each allele. The column “Percentage Created” is the percentage of explanations that led to a minimal explanation (based on the previous two columns).

Allele	Explanations Produced by DeepCover	Minimal Explanations Produced	Percentage Created	Average Size of Minimal Explanations (out of 43)
HLA-A*33:01	458	289	63%	20.3
HLA-A*33:03	505	336	67%	20.8
HLA-A*36:01	410	349	85%	17.5
HLA-A*74:01	495	367	74%	25.9
HLA-B*37:01	206	78	38%	20.2
HLA-B*46:01	246	162	66%	23.7
HLA-B*54:01	465	335	72%	22.2
HLA-B*58:01	382	328	86%	21.6
HLA-B*58:02	340	277	81%	24.2
HLA-C*01-02	445	235	53%	22.7
HLA-C*15-02	291	115	40%	22.8
HLA-C*17-01	467	316	68%	19.7

Summary View of Frequent Positions. Just like with the rankings, one minimal explanation was produced per instance. Figures 5.7-5.9 summarise these minimal explanations for the three representative alleles. These figures show how many times each amino acid position in the peptide and MHC pseudo sequence was included in a minimal explanation. Intuitively, we expect these figures to mirror the Tarantula ranking figures as that is the order they were added to the minimal explanations.

HLA-A*33:01. Looking at Figure 5.7, the positions that stand out for HLA-A*33:01 are mainly 9 in the peptide and 24, 62, 63, 66, 77 and 147 in the MHC input. In the ranking plot, position 9 of the peptide and 63 of the MHC are clearly the most important. However, the other positions highlighted here in the minimal explanation summary have small but detectable higher importances than their neighbours in the Tarantula ranking plot. Position 2 was quite important in the ranking plot being the 3rd most important position overall. However, here in the minimal explanation summary plot, it is a frequent position but does not stand out as much as in the ranking plot when compared to the frequency of the other peptide positions.

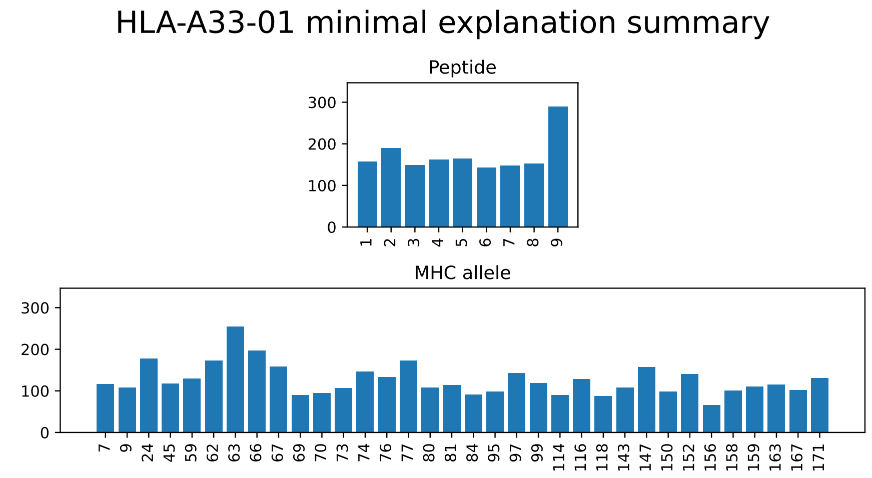


Figure 5.7: A summary of the minimal explanations produced for allele HLA-A*33:01 where the bars show how many times each position appeared in a minimal explanation.

HLA-B*54:01. As shown in Figure 5.8, HLA-B*54:01 has spikes at locations 2 and 9 of the peptide which were also clearly the most important positions in the allele's Tarantula ranking plot. As for the MHC molecule, the most frequent positions in the minimal explanations were 24, 77, 95 and the neighbouring positions on its right, creating a stair-like shape. Like before, the ranking plot shows the same patterns.

HLA-C*01:02. Figure 5.9 presents a summary of the minimal explanations for allele HLA-C*01:02. The positions that stand out as the most frequent in this figure include positions 3 and 9 of the peptide and 9, 45 and 99 of the MHC part. While the ranking plot looks very similar in terms of spikes, they are quite different in terms of relative size. For instance, position 3 of the peptide is much more highly ranked than peptide position 9 but they appear in approximately the same number of minimal explanations because of the size of the minimal explanations.

Generally, the relative differences between lowly and highly ranked positions is smaller in the minimal explanation summary plots. This is likely because the minimal expla-

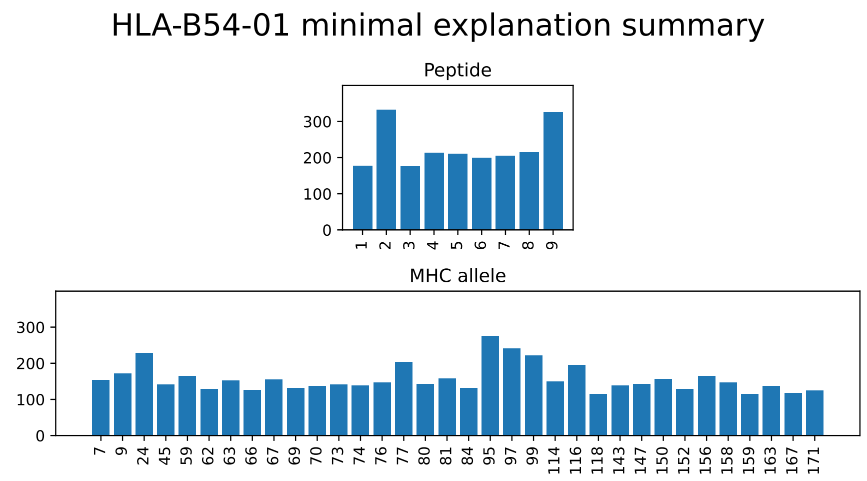


Figure 5.8: A summary of the minimal explanations produced for allele HLA-B*54:01 where the bars show how many times each position appeared in a minimal explanation.

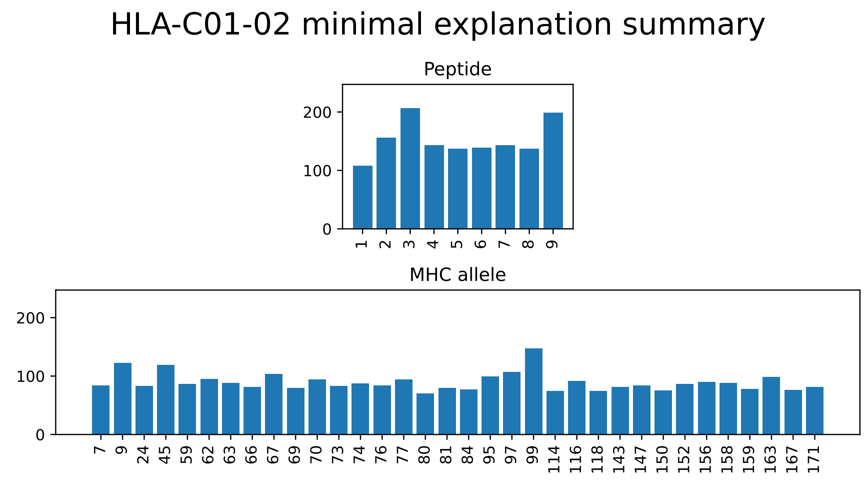


Figure 5.9: A summary of the minimal explanations produced for allele HLA-C*01:02 where the bars show how many times each position appeared in a minimal explanation.

nations are quite large on average, as seen in Table 5.3. Large minimal explanations mean that more weight is given to lower ranked positions which introduces more noise in those lower ranked positions. Minimal explanations work better when used as a local explanation method. DeepCover was made for images and their minimal explanations were certain regions of an image, to show what the model paid the most attention to in its decision-making. Here, we were more concerned with the overviews, such as the per-allele view. In these situations, the minimal explanations provide a different and less detailed way of viewing the importance rankings rather than providing a new perspective on the model's decision-making.

5.2 Comparison with LIME and DC-Causal

This section contains the second part of the evaluation, comparison against the other explanation techniques LIME and DC-Causal. The evaluation starts with a qualitative comparison of the across-instance ranking figures based on the explanations produced by the different explanation techniques. This is followed by an instance-wise comparison of agreement between DeepCover with Tarantula SFL, LIME and DC-Causal. To simplify the discussion about DeepCover with SFL modified in this project and the DeepCover variant DC-Causal, they are referred to as DC-SFL and DC-Causal, respectively, when comparing the two.

5.2.1 Across-Instance Comparisons

The first level of comparison between the explanation techniques is comparing them across instances, per allele, in figures. The DC-SFL and DC-Causal figures have been designed to look similar to the LIME-based ranking figures, produced for the MSc project that developed ImmunoBERT [4], to simplify comparisons. As the LIME figures included the flanks, the colours of the MHC pseudo sequences in the DC-SFL and DC-Causal figures were adjusted to make the figures more comparable.

5.2.1.1 LIME

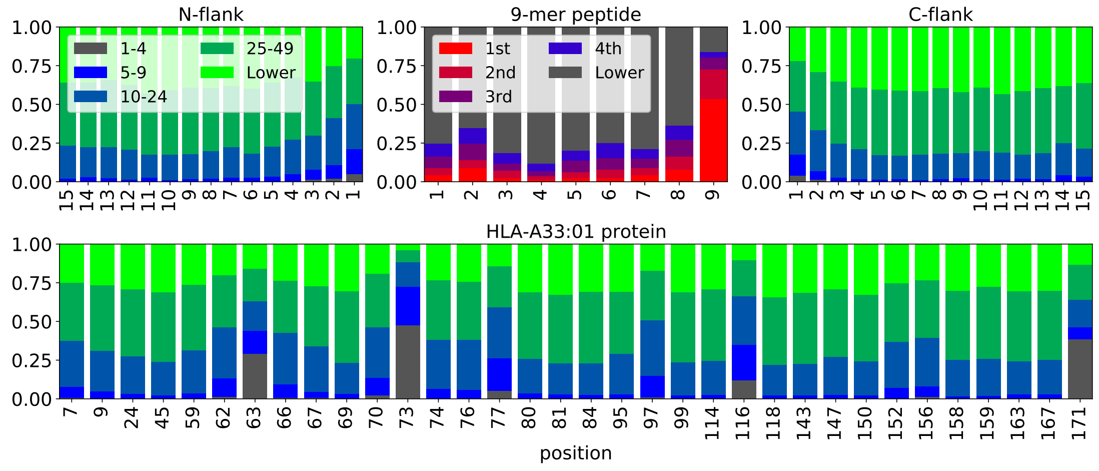


Figure 5.10: Across-instance view of the rankings for the peptide (top center), flanks (top left and right) and MHC pseudo sequence (bottom) of allele HLA-A*33:01 given by LIME. Figure from Gasser [4].

HLA-A*33:01 Peptide Importance. The rankings produced by LIME for allele HLA-A*33:01 are shown in Figure 5.10. Comparing this to Figure 5.1, produced by DeepCover with Tarantula, the peptide rankings were very similar. Position 9 was by far the most important position in the peptide in both figures, being ranked first in about half of the instances by both DeepCover and LIME. Additionally, position 2 was the second most important and position 4 was the lowest ranked in the peptide for both explanation methods.

HLA-A*33:01 MHC Importance. Both techniques agreed that position 63 of the MHC pseudo sequence was very important and highly ranked in many instances. However, LIME assigned higher importance to additional positions, such as 73, 116 and 171. These positions were slightly higher than their neighbours in DeepCover, creating a similar overall shape in the MHC input, but did not stand out as much as in the LIME figures. A few other positions, such as 24, had a similarly small spike among their neighbours in DeepCover but no spike in the LIME plot.

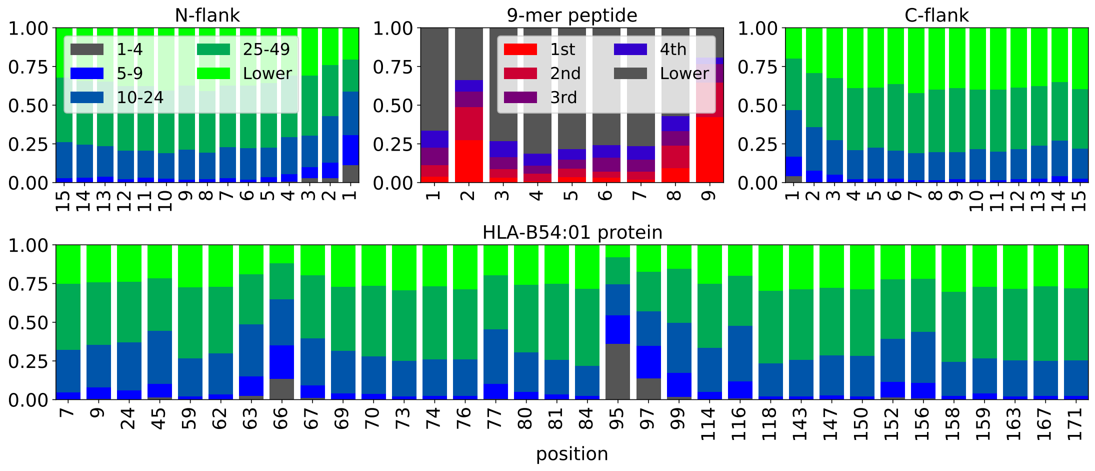


Figure 5.11: Across-instance view of the rankings for the peptide (top center), flanks (top left and right) and MHC pseudo sequence (bottom) of allele HLA-B*54:01 given by LIME. Figure from Gasser [4].

HLA-B*54:01 Peptide Importance. For allele HLA-B*54:01, the rankings produced by LIME are shown in Figure 5.11. In the peptide, both methods agreed that the most important positions were 2 and 9. LIME assigned a slightly higher importance to position 8 of the peptide which DeepCover did not.

HLA-B*54:01 MHC Importance. In the MHC input, there was a clear agreement around position 95. Both LIME and DeepCover highlighted position 95 as the most important in the MHC input, had a stair-like shape in the three positions to the right and a little spike again at position 116. They both picked up position 77 but position 66 was only high in LIME and not in DeepCover. LIME again had larger relative differences in the MHC part of the input.

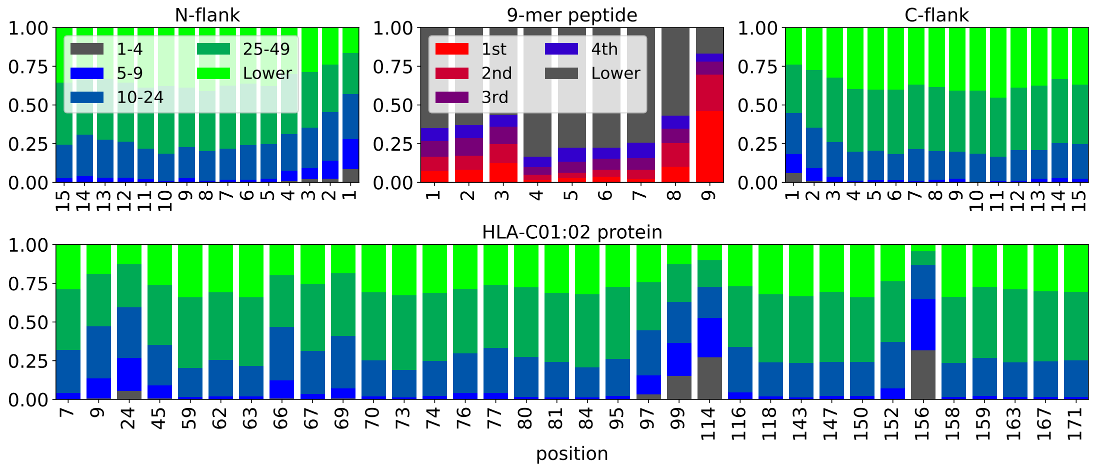


Figure 5.12: Across-instance view of the rankings for the peptide (top center), flanks (top left and right) and MHC pseudo sequence (bottom) of allele HLA-AC*01:02 given by LIME. Figure from Gasser [4].

HLA-C*01:02 Peptide Importance. Figure 5.12 presents the results of LIME for allele HLA-C*01:02. For this allele, there was less agreement between the two techniques. In the peptide, position 3 was the most important in DeepCover, followed by position 9. LIME instead set position 9 as the most important and position 3 as the second most important, with a substantial difference between the sizes of the bars. However, the methods did agree that these two positions were the most important part of the peptide.

HLA-C*01:02 MHC Importance. Position 156 was the most important in the MHC input according to LIME while DeepCover ranked position 156 and its neighbour, position 152, only slightly higher than their other neighbours. Following position 156, LIME ranked position 114, 99 and 97, in decreasing order. While DeepCover did not highlight position 114, it did rank the neighbours on the left, 97 and 99 as the highest in the MHC input. Both methods also had a small spike for positions 9 and 24. While there were some similarities to be found, they were much more subtle than observed for the previous two alleles.

Conclusion. Overall, the two methods uncover similar important positions in the peptides but LIME identified more highly ranked positions in the MHC input. The agreement between LIME and DeepCover with Tarantula for these three alleles suggests that DeepCover does work as an explanation method. As the methods agree about the importance of the peptides, we can be fairly certain that these regions are important. The figures for the rest of the alleles can be found in appendix A for DeepCover with Tarantula and in the appendix of the original ImmunoBERT MSc project for the LIME figures [4] for further comparison.

5.2.1.2 DC-Causal

Figures B.1-B.3 in Appendix B present the across-instance view of the rankings produced by DC-Causal for the three alleles HLA-A*33:01, HLA-B*54:01 and HLA-C*01:02. They were again formatted to correspond to the figures created by Gasser [4] to facilitate comparison.

HLA-A*33:01 Peptide Importance. Both DC-SFL and DC-Causal recognised position 2 and 9 as important but DC-Causal ranked position 1 highest more often than position 9. Moreover, DC-Causal placed a larger emphasis on the positions in the start of the peptide.

HLA-A*33:01 MHC Importance. In the MHC section, both DC-SFL and DC-Causal ranked position 63 the highest. Additionally, they both ranked positions 73 and 77 slightly higher than their neighbours but the other slight fluctuations between the position importances were not the same between the two explanation methods. DC-Causal again assigned higher importances to the start of the sequence.

HLA-B*54:01 Peptide Importance. Both methods agreed that position 2 was important but DC-SFL ranked position 9 very high while DC-Causal found position 9 one of the least important position of the peptide. This is a significant disagreement as the magnitudes are substantial.

HLA-B*54:01 MHC Importance. Both methods highlighted positions 95, 97 and

99 in a stair-like shape but did not highlight the same positions of the MHC pseudo sequence otherwise. DC-Causal again assigned higher importances to the start of both sequences.

HLA-C*01:02 Peptide Importance. Both methods highlighted peptide position 3 as the most important. However, they disagreed on the other peptide positions where DC-Causal placed a higher emphasis on the earlier positions 1 and 2 while DC-SFL highlighted position 9 as the second most important.

HLA-C*01:02 MHC Importance. In the MHC pseudo sequence of HLA-C*01:02, positions 24, 99 and 156 were assigned a slightly above average importance ranking but the differences between the ranking of the positions of the MHC inputs were so slight that it is hard to draw any conclusions about similarity.

Total order. DC-Causal is a top-down explanation method which can lead to coarse explanations. In DC-Causal, the positions are iteratively partitioned and the whole partition is given the same importance score when the stopping conditions are met. For instance, consider a peptide of length 9 that is partitioned once into partitions of size 4 and 5. If those partitions cannot be further partitioned, the 4 positions in the first partition will receive the same importance score and the 5 positions in the second partition will receive the same importance score. When assigning a total ordering to the positions, a level of arbitrary ordering was introduced which was likely alphabetically skewed, resulting in early positions being given higher rankings. The coarse-grained explanations are a limitation of DC-Causal but the arbitrary ordering problem is only introduced when ranking the positions based on the given importance scores. While the plotting method allows for multiple positions to have the same ranking, only the total order rankings were available when creating the figures.

Incomplete rankings. Another consequence of the coarse-grained explanations is that the rankings produced by DC-Causal do not always include all positions of the input sequences. All positions not included in the DC-Causal rankings were assigned ranking 200 when creating the figures, setting them to the least important as mutations in their partition did not lead to sufficient label switches to be assigned an importance score. This does not contribute to the problem of earlier parts of the sequences being ranked higher.

Conclusion. DC-SFL and DC-Causal agreed that the peptide was the most important part of the input overall. Somewhat similar patterns could be found in the MHC rankings but the explanations produced by the two methods differed on the rankings of a few key peptide positions. The DC-Causal figures assigned higher rankings to the start of the sequences for all three alleles because of a bias introduces by the total ordering.

5.2.2 Instance-Wise comparison

While presenting the explanations as global explanations gives more intuitive presentations of the results, DeepCover with Tarantula SFL, LIME and DC-Causal are all local explanation methods. Therefore, this section contains an instance-wise comparisons of the explanations produced by the three techniques.

5.2.2.1 Coverage

Coverage. Table 5.4 shows how many of the 1000 instances each of the three explanation methods, LIME, DC-SFL and DC-Causal, managed to explain. As discussed in Section 5.1.1.1, the coverage of DC-SFL was mostly less than 50% and Table 5.4 shows that the coverage of DC-Causal was even lower. Both DeepCover variants rely on mutations of the input changing the predicted class label to produce explanations. DC-Causal is even further restricted than DC-SFL as it splits the input into partitions and masks subsets of these rather than mutating any subset of positions. LIME does not rely on switching the predicted class label. It instead uses the difference between the original and mutated instances' presentation scores to produce explanations and could therefore explain all instances.

Table 5.4: The coverage (instances that the methods could produce explanations for) of the three explanation methods for the same 1000 instances per allele. The overlap in the final column is the number of instances out of the 1000 that were explained by both DC-SFL and DC-Causal. DC-SFL refers to DeepCover with the Tarantula SFL measure.

Allele	**LIME Coverage**	**DC-SFL Coverage**	**DC-Causal Coverage**	**Overlap DC-SFL and DC-Causal**
HLA-A*33:01	100%	45.8%	39.5%	395
HLA-A*33:03	100%	50.5%	43.8%	438
HLA-A*36:01	100%	41.0%	32.3%	323
HLA-A*74:01	100%	49.5%	42.7%	427
HLA-B*37:01	100%	20.6%	14.2%	142
HLA-B*46:01	100%	24.6%	21.3%	213
HLA-B*54:01	100%	46.5%	42.8%	428
HLA-B*58:01	100%	38.2%	35.5%	355
HLA-B*58:02	100%	34.0%	29.3%	293
HLA-C*01-02	100%	44.5%	35.2%	352
HLA-C*15-02	100%	29.1%	22.9%	229
HLA-C*17-01	100%	46.7%	41.2%	412

Relationship Between Coverage of DC-SFL and DC-Causal. The last column of Table 5.4 shows how many of the 1000 instances were explained by both DC-SFL and DC-Causal. Comparing these numbers to the coverage of DC-Causal, it shows that all instances that were explained by DC-Causal were also explained by DC-SFL. In other words, the set of instances that DC-Causal could produce explanations for was a subset of the instances that DC-SFL could produce explanations for. This highlights that DC-Causal ran into difficulties for the same instances as DC-SFL because the predicted labels of these instances did not change upon mutations.

5.2.2.2 Instance-Level Agreement

Instance-wise Comparison of Explanations. Table 5.5 shows the results of instance-wise comparisons as the average RBO score per allele. Considering that an RBO score of 0 means that the lists are disjoint and 1 means that they are identical, we see that there is some agreement between DC-SFL and the other methods, especially LIME. Like the aggregate comparisons, this shows that the different methods agree on the trends but not on the details. Most importantly, it shows that DC-SFL has the potential to uncover positions deemed important by other explanation methods as well.

Table 5.5: Average RBO scores per allele. For the combined comparisons of peptide and MHC pseudo sequence, 85% of the scores was assigned to the top 10 positions. For the peptide-only scores, 85% of the weight was assigned to the top two positions. DC-SFL refers to DeepCover with the Tarantula SFL measure.

Allele	**LIME vs DC-SFL**	**LIME vs DC-SFL peptide only**	**DC-Causal vs DC-SFL**	**DC-Causal vs DC-SFL peptide only**
HLA-A*33:01	0.432	0.510	0.395	0.407
HLA-A*33:03	0.436	0.549	0.386	0.396
HLA-A*36:01	0.515	0.647	0.316	0.283
HLA-A*74:01	0.427	0.452	0.317	0.274
HLA-B*37:01	0.332	0.260	0.382	0.361
HLA-B*46:01	0.376	0.306	0.348	0.305
HLA-B*54:01	0.458	0.472	0.401	0.442
HLA-B*58:01	0.414	0.516	0.340	0.344
HLA-B*58:02	0.386	0.346	0.325	0.308
HLA-C*01-02	0.393	0.383	0.393	0.409
HLA-C*15-02	0.335	0.291	0.405	0.443
HLA-C*17-01	0.429	0.422	0.407	0.439

Comparison of RBO Scores and Across-Instance Rankings for LIME and DC-SFL. The highest RBO score between LIME and DC-SFL was 0.647 for HLA-A*36:01 and the lowest were 0.260 for HLA-B*37:01 and 0.291 for HLA-C*15-02. The across-instance figures produced by DC-SFL for these alleles can be found in appendix A and those produced by LIME are included in the appendix of the original ImmunoBERT report [4]. DC-SFL and LIME showed moderate across-instance agreement in the MHC sequence. Where the alleles with high and low RBO scores differed was in the peptide sequence. For the low RBO alleles, LIME highlighted the latter parts of the peptide, like C-terminal positions 8 and 9, while DC-SFL put a larger emphasis on the start of the peptide, like N-terminal positions 2 and 3, or had a more uniform distribution. Meanwhile, for the highest scored allele, HLA-A*36:01, peptide position 9 was clearly the most important position according to both methods. In conclusion, the alleles with lower agreement, identified by a lower RBO score, also had lower quality agreement on the across-instance level.

DC-Causal RBO scores and Comparison to Across-Instance Rankings. For most alleles, the RBO scores of DC-SFL compared to LIME were higher than DC-SFL compared to DC-Causal. As discussed in Section 5.2.1.2, DC-Causal produces coarse-grained explanations that had a level of arbitrary ordering when assigning a total order to the positions based on the importance scores. To apply RBO, the positions cannot share the same rank so total ordering was necessary. Looking back at the across-instance comparison of DC-SFL and DC-Causal, they both ranked the peptide positions as containing the most important positions compared to the MHC positions but they disagreed on the importance of a few key peptide positions, partially skewed by the bias introduced by the total ordering.

Peptide-Only RBO Scores. The ranking of the peptide is particularly important as RBO assigned 86% of the weight to the top 10 items and 52% to the top 3. To have a high RBO score it is therefore important that the explanation methods agree on the most important positions, which tend to be in the peptide. The peptide-only comparison also shown in Table 5.5 lead to a higher score in only about half of the alleles and in the other half it led to a lower score. This shows that the methods did not agree more on the peptide parts of the inputs overall. Out of the three alleles with the highest and lowest RBO scores between LIME and DC-SFL discussed previously, HLA-A*36:01, scored the by far highest peptide-only RBO score between LIME and DC-DFL while the lowest ranked alleles, HLA-B*37:01 and HLA-C*15-02, showed a decrease in peptide-only RBO score. This reflects the analysis of the across-instance figures.

Viability of Evaluation by Agreement. Neely et al. [10] performed a study where they compared an array of explanation methods presented in the related work section (Section 2.3) of this report: LIME, Integrated Gradients, DeepLIFT, GradSHAP, DeepSHAP and attention-based explanations. They produced explanations for 500 instances for an LSTM model and a light-weight BERT model for natural language applications. To compare the explanations, they computed Kendall's τ between each pair of explanation methods for the two models. They found that the explainable AI methods rarely correlated with each other which means that the method of evaluating a new explanation method by comparing the correlation with explanations produced by other explanation methods is likely not ideal.

5.3 Runtime

DeepCover vs LIME. Table 5.6 shows the per-instance runtimes of LIME and DeepCover computed by explaining 50 instances and averaging the runtime. Both DeepCover and LIME are perturbation-based explanation techniques, meaning that they perturb the input to create mutations and observe how that affects the predictions of a model. They were both run with 2000 mutations per instance and so both ran the ImmunoBERT inference 2000 times per given instance. DeepCover was substantially slower than LIME. This shows that the way perturbation-based explanation methods are implemented largely affects the runtime.

Table 5.6: Runtime per instance computed as the runtime averaged over the number of instances when explaining 50 instances.

Explanation method	Time per instance
LIME	13 seconds
DeepCover	85 seconds
DeepCover Parallel	16 seconds

Parallelisation. The DeepCover implementation provided by Sun et al. [8] was completely serial with nested for-loops that looped over the instances and each mutant of each instance. When computing the SFL scores, it looped over each SFL measure and then over each position of the input to compute the score per position. The internals of the DeepCover tool were not parallelised or vectorised by either the original authors or as part of the current project. However, batch options were added to allow for external parallelisation in a bash script. The last row in Table 5.6 shows that parallelisation resulted in a significant runtime improvement over the un-parallelised DeepCover and made the runtime per instance comparable to LIME. The runtime could be further improved by optimising the DeepCover code.

5.4 Threats to Validity

The main threats to validity of the experiments relate to the data generation, threshold choice, and lack of gold standard.

Subset of Alleles Explained. MHC alleles vary a lot across the population, as discussed in Section 2.1.1. Only a small subset of these were explained as part of the project. These were the subset consisting of 12 alleles chosen by Gasser [4] with a focus on the analysis of three of them. The subset of alleles that were chosen may not be representative of the varied nature of the alleles.

Decoy Generation. The decoys (negative instances) were generated by Gasser [4] by randomly sampling non-presented peptides from the set of proteins that the presented peptides originated from. This method of decoy generation has not been validated and the decoys may not always be biologically possible. This might have affected the low number of explanations that were created for the negative instances but likely did not skew the quality of the across-instance as few negative instances were included in these results. If more negative explanations were available, the presentation of the across-instance results should be split into negative and positive to explain the instances on a per-class basis.

Explaining the Absence of a Label. The next threat is an abstract point where the binary class problem that we are explaining can be viewed as the presence and absence of some characteristics that makes the peptide presented or not. Explaining the negative instances is useful as that can tell us what to avoid when aiming to create presented peptides but conceptually, DeepCover was designed to explain the inclusion in a class, rather than the exclusion from, or absence of, a class. This makes DeepCover more suitable to multi-class models. The problem of explaining an absence of class was particularly clear in the minimal explanation experiments where masking the whole input led to a negative class prediction, as the characteristics that make an MHC molecule present the peptide were missing.

ImmunoBERT Threshold. The choice of threshold for turning ImmunoBERT into a classification from a presentation score was arbitrarily set to 0.5. This likely contributed to the difficulties of explaining negative instances by being too high. A possible solution to setting the threshold to a biologically motivated value is discussed in Section 6.1.

Lack of Explanation Gold Standard. Lastly, the lack of gold standard for explanations means that the evaluation focused on motivating the produced explanations and comparing to explanations produced by other methods. Other explanation methods are not a gold standard, and the pitfalls of evaluation by agreement is discussed in Section 5.2.2.2. However, the input space of the explanations should be considered for the lack of gold standard as there likely exists no single true explanation because peptide presentation is a complex problem.

Chapter 6

Discussion

Summary of Results. The evaluation focused on three parts:

Part 1: Performance of DeepCover in Isolation. The maximum explanation **coverage** was 50%, because of the restriction on mutations causing label switches. DeepCover was mostly able to explain positive instances. For the explanations that were produced, the across-instance view highlighted positions that could be **biologically motivated** as important for peptide binding and presentation. The explanations produced by the different **fault localisation measures** mostly agreed very well. The **minimal explanations** could be produced only for the instances predicted as positive but where they were produced, they were smaller than the full explanations and included the important positions.

Part 2: Comparison Against LIME and DC-Causal. The explanation coverage of LIME was better while the coverage of DC-Causal was worse than for DeepCover. There was some agreement and some disagreement on important positions in both **across-instance** and **per-instance** comparisons.

Part 3: Runtime Efficiency. Data **parallelisation** of DeepCover greatly improved the runtime and made it competitive with LIME. Further optimisation is possible within the DeepCover code.

In conclusion, DeepCover was successfully modified to explain the amino acid input. The biological motivation of explanations and partial agreement with other explanation methods indicate that DeepCover uncovered valuable insights into the decisions made by the prediction model.

6.1 Limitations

The main limitations of the project and explanation method include the coverage of the full explanations, the coverage of the minimal subset explanations and the threshold used to convert ImmunoBERT's presentation score into a classification.

Binary Class Problem. DeepCover was originally developed for multi-class problems and relies on mutations in the input changing the predictions. The peptide presentation application discussed in this report is a binary class problem, where a peptide can be presented or not presented. In many cases, DeepCover could not produce explanations because the mutation did not lead to changes in predicted labels as frequently in the binary-class application as in the original multi-class application. This problem was especially clear in the decoys that were clearly under-represented in the produced explanations, as seen in Table 5.1. Adjusting the model threshold would improve the coverage of explanations but, conceptually, mutation-based methods explanation methods that require label switches fit better to multi-class problems.

Binary Class Problem for Minimal Subset Explanations. DeepCover could not create minimal explanations for instances with a predicted negative label. The problem was that masking all positions resulted in a negative label and adding positions in order of importance could not cause the label to switch from negative to negative. This was again a consequence of the binary class problem. The application in the minimal explanation experiments in the original DeepCover paper [8] was image classification with many classes meaning that masking all pixels changed the label and adding pixels in order of importance did cause the label to change back to the original label.

A possible way to adjust the minimal explanations to work better with negative instances is to start with a synthetic positive instance where all positions of the input have been mutated rather than masked. The minimal explanation would be created by "unmutating" the amino acids in the positions following the ranked order until the label changes back to negative. A possible downside of this alternative method is that the minimal explanation produced might depend too much on the mutated starting instance and it might require finding an agreement between minimal explanations produced from different starting points for each instance.

ImmunoBERT Threshold. The threshold used for ImmunoBERT to distinguish between predicted or not was arbitrarily set to 0.5 for the (sigmoid) presentation score which was in the range [0,1]. To demonstrate how the threshold could be selected in a more theoretically motivated way, we can compare to other models.

Binding affinity models predict only the most restrictive step of the peptide presentation pathway: the binding step when the MHC-I/peptide complexes are created [54]. Binding affinity models commonly predict the IC_{50} score, a measure of binding affinity. As the scale of the binding affinity score compares to a real concept and measurement, the threshold of the predicted IC_{50} score is often set to 500nM which is the value that has been experimentally decided as the threshold for peptide presentation through *in vivo* and *in vitro* experiments [55].

The creators of the well-known MHC class I-peptide binding model NetMHCpan re-evaluated the hard IC_{50} threshold as the fraction of positive predictions varied substantially for different alleles [56]. They found that ranking the predicted binding scores within each allele and assigning a percentile ranking score resulted in higher sensitivity and specificity in the receiver operating characteristic curve when varying the percentile rank compared to the binding affinity score.

A similar approach applied to ImmunoBERT would help by not being arbitrary and being fairer across alleles. It could involve scaling the score, activation function or threshold to make the predictions match the distribution of peptides that are presented in nature. For instance, 1/200 of random peptides bind to a given MHC-I molecule [57]. As binding is the most restrictive step for peptide presentation [54], it could be used as an approximation of the ratio of peptide presentation.

6.2 Further Work

Other Peptide Presentation Models. The first suggestion for future work concerns generalising the performance evaluation of DeepCover with SFL for the peptide presentation problem. Practically, that would involve performing similar experiments to the ones presented in this project for other peptide presentation models like NetMHCPan with the aim of evaluating how well DeepCover works in the problem domain, rather than for ImmunoBERT specifically.

Agreement Between Explanation Methods. It is not uncommon for different explainable AI techniques to disagree. To further investigate the effect of different explainable AI methods, a possible future avenue is to combine a collection of explainable AI methods for the peptide presentation problem, compare the agreement between them and produce scores of how much each method agrees with the other techniques or highlights unique positions. This would not produce the true explanations as there are likely many possible explanations for peptide presentation outcomes. However, having multiple explanations could cover a larger part of the explanation input space and where they agree we would be more confident of the importance. Additionally, knowing how the methods agree and relate to each other would be valuable to understand their performance.

Biological Gold Standard. To truly evaluate whether the explanations were useful, a biologically motivated gold standard dataset is needed. This could for instance be the results of *in vitro* experiments from vaccine development exploring which mutated peptides are presented by the MHC-I alleles. Such a dataset would allow us to see the impact of mutating certain positions, identify the truly biologically important positions of a few instances to compare to the explanations and assess the predictability of peptide presentation using ImmunoBERT.

Bibliography

[1] Peter J. Delves and Ivan M. Roitt. “The Immune System”. In: *New England Journal of Medicine* 343.1 (2000), pp. 37–49. DOI: 10.1056/NEJM200007063430107.

[2] J. D. Comber and R. Philip. “MHC class I antigen presentation and implications for developing a new generation of therapeutic vaccines”. In: *Ther Adv Vaccines* 2.3 (May 2014), pp. 77–89. DOI: 10.1177/2051013614525375.

[3] K. L. Rock, E. Reits, and J. Neefjes. “Present Yourself! By MHC Class I and MHC Class II Molecules”. In: *Trends Immunol* 37.11 (Nov. 2016), pp. 724–737. DOI: 10.1016/j.it.2016.08.010.

[4] Hans-Christof Gasser. “The BERT architecture for the prediction of peptide presentation by MHC class I proteins”. MSc thesis. School of Informatics, University of Edinburgh, 2021.

[5] Hans-Christof Gasser, Georges Bedran, Bo Ren, David Goodlett, Javier Alfaro, and Ajitha Rajan. *Interpreting BERT architecture predictions for peptide presentation by MHC class I proteins*. 2021. arXiv: 2111.07137.

[6] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. “Explainable AI: A Review of Machine Learning Interpretability Methods”. In: *Entropy* 23.1 (2021). DOI: 10.3390/e23010018.

[7] Andreas Holzinger, Chris Biemann, Constantinos S. Pattichis, and Douglas B. Kell. *What do we need to build explainable AI systems for the medical domain?* 2017. arXiv: 1712.09923.

[8] Youcheng Sun, Hana Chockler, Xiaowei Huang, and Daniel Kroening. “Explaining Image Classifiers using Statistical Fault Localization”. In: *European Conference on Computer Vision (ECCV)*. Vol. 12373. LNCS. Springer, 2020, pp. 391–406. DOI: /10.1007/978-3-030-58604-1_24.

[9] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “”Why Should I Trust You?”: Explaining the Predictions of Any Classifier”. In: KDD ’16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 1135–1144. DOI: 10.1145/2939672.2939778.

[10] Michael Neely, Stefan F Schouten, Maurits JR Bleeker, and Ana Lucic. *Order in the Court: Explainable AI Methods Prone to Disagreement*. May 2021. arXiv: 2105.03287.

[11] Gwenyth Rooijackers. “Recording Execution Traces through Java Bytecode Programs”. MInf (Part 1) Project. School of Informatics, University of Edinburgh, 2021.

[12] Foivos Tsimpourlas, Gwenyth Rooijackers, Ajitha Rajan, and Miltiadis Allamanis. “Embedding and classifying test execution traces using neural networks”. In: *IET Software* (Aug. 2021). DOI: 10.1049/sfw2.12038.

[13] 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 (June 2015), pp. 425–436. DOI: 10.1007/s00251-015-0849-y.

[14] Anette Stryhn, Lars Østergaard Pedersen, Arne Holm, and Søren Buus. “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. DOI: /10.1002/1521-4141(200011)30:11⟨3089::AID-IMMU3089⟩3.0.CO;2-5.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “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)*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[16] Morten Nielsen, Claus Lundegaard, Thomas Blicher, Kasper Lamberth, Mikkel Harndahl, Sune Justesen, Gustav Røder, Bjoern Peters, Alessandro Sette, Ole Lund, and Søren Buus. “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). Ed. by Esper Kallas. DOI: 10.1371/journal.pone.0000796.

[17] Zachary C. Lipton. “The Mythos of Model Interpretability: In Machine Learning, the Concept of Interpretability is Both Important and Slippery.” In: *Queue* 16.3 (June 2018), pp. 31–57. DOI: 10.1145/3236386.3241340.

[18] Mourad Oussalah. “AI Explainability. A Bridge Between Machine Vision and Natural Language Processing”. In: *Pattern Recognition. ICPR International Workshops and Challenges*. Ed. by Alberto Del Bimbo, Rita Cucchiara, Stan Sclaroff, Giovanni Maria Farinella, Tao Mei, Marco Bertini, Hugo Jair Escalante, and Roberto Vezzani. Cham: Springer International Publishing, 2021, pp. 257–273.

[19] Dan Ofer, Nadav Brandes, and Michal Linial. “The language of proteins: NLP, machine learning & protein sequences”. In: *Computational and Structural Biotechnology Journal* 19 (2021), pp. 1750–1758. DOI: https://doi.org/10.1016/j.csbj.2021.03.022.

[20] Sumanta. Basu, K. Kumbier, J. B. Brown, and B. Yu. “Iterative random forests to discover predictive and stable high-order interactions”. In: *Proceedings of the National Academy of Sciences (PNAS)* 115.8 (Feb. 2018), pp. 1943–1948. DOI: /10.1073/pnas.1711236115.

[21] Ashley Cliff, Jonathon Romero, David Kainer, Angelica Walker, Anna Furches, and Daniel Jacobson. “A High-Performance Computing Implementation of Iterative Random Forest for the Creation of Predictive Expression Networks”. In: *Genes* 10.12 (2019). DOI: 10.3390/genes10120996.

[22] Alfred Ultsch, Jörg Hoffmann, Maximilian Röhnert, Malte Von Bonin, Uta Oelschlägel, Cornelia Brendel, and Michael C. Thrun. *An Explainable AI System*

for the Diagnosis of High Dimensional Biomedical Data. 2021. arXiv: 2107.01820.

[23] Been Kim, Rajiv Khanna, and Oluwasanmi Koyejo. "Examples Are Not Enough, Learn to Criticize! Criticism for Interpretability". In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 2288–2296.

[24] Ethan R. Elenberg, Alexandros G. Dimakis, Moran Feldman, and Amin Karbasi. "Streaming Weak Submodularity: Interpreting Neural Networks on the Fly". In: NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 4047–4057.

[25] Muhammad Rehman Zafar and Naimul Mefraz Khan. *DLIME: A Deterministic Local Interpretable Model-Agnostic Explanations Approach for Computer-Aided Diagnosis Systems*. 2019. arXiv: 1906.10263.

[26] Kacper Sokol and Peter Flach. *LIMEtree: Interactively Customisable Explanations Based on Local Surrogate Multi-output Regression Trees*. 2020. arXiv: 2005.01427.

[27] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Anchors: High-Precision Model-Agnostic Explanations". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. Apr. 2018.

[28] Gregory Plumb, Denali Molitor, and Ameet S Talwalkar. "Model Agnostic Supervised Local Explanations". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018.

[29] Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. "Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations". In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. Barcelona, Spain: Association for Computing Machinery, 2020, pp. 607–617. DOI: 10.1145/3351095.3372850.

[30] David Harbecke and Christoph Alt. "Considering Likelihood in NLP Classification Explanations with Occlusion and Language Modeling". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. Online: Association for Computational Linguistics, July 2020, pp. 111–117. DOI: 10.18653/v1/2020.acl-srw.16.

[31] 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. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 4768–4777.

[32] Patrick Hall, Navdeep Gill, Megan Kurka, and Wen Phan. *Machine Learning Interpretability with H20 Driverless AI*. Sept. 2021. URL: https://docs.h2o.ai/driverless-ai/latest-stable/docs/booklets/MLIBooklet.pdf.

[33] Quinn Dickinson and Jesse G. Meyer. "Positional SHAP (PoSHAP) for Interpretation of machine learning models trained from biological sequences". In: *PLOS Computational Biology* 18.1 (Jan. 2022). Ed. by Ilya Ioshikhes. DOI: 10.1371/journal.pcbi.1009736.

[34] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic Attribution for Deep Networks". In: *Proceedings of the 34th International Conference on Ma-*

chine Learning. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 3319–3328.

[35] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. "Learning Important Features through Propagating Activation Differences". In: ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 3145–3153.

[36] Samira Abnar and Willem Zuidema. "Quantifying Attention Flow in Transformers". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 4190–4197. DOI: 10.18653/v1/2020.acl-main.385.

[37] Alexis Ross, Ana Marasović, and Matthew Peters. "Explaining NLP Models via Minimal Contrastive Editing (MiCE)". In: *Findings of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Aug. 2021, pp. 3840–3852. DOI: 10.18653/v1/2021.findings-acl.336.

[38] Enja Kokalj, Blaž Škrlj, Nada Lavrač, Senja Pollak, and Marko Robnik-Šikonja. "BERT meets Shapley: Extending SHAP Explanations to Transformer-based Classifiers". In: *Proceedings of the EACL Hackashop on News Media Content Analysis and Automated Report Generation*. Online: Association for Computational Linguistics, Apr. 2021, pp. 16–21.

[39] Scott Lundberg. *SHAP, Deep learning example with GradientExplainer (TensorFlow/Keras/PyTorch models)*. 2021. URL: https://github.com/slundberg/shap#deep-learning-example-with-gradientexplainer-tensorflowkeraspytorch-models (visited on 10/10/2021).

[40] Jiwei Li, Will Monroe, and Dan Jurafsky. *Understanding Neural Networks through Representation Erasure*. 2017. arXiv: 1612.08220.

[41] Lloyd S. Shapley. "A value for n-person games". In: *Contributions to the Theory of Games* 28 (1953), pp. 307–317.

[42] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. "Gradient-Based Attribution Methods". In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Ed. by Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller. Cham: Springer International Publishing, 2019, pp. 169–191. DOI: 10.1007/978-3-030-28954-6_9.

[43] Lee Naish, Hua Jie Lee, and Kotagiri Ramamohanarao. "A Model for Spectra-Based Software Diagnosis". In: *ACM Transactions on Software Engineering and Methodology* 20.3 (Aug. 2011). DOI: 10.1145/2000791.2000795.

[44] Hana Chockler, Daniel Kroening, and Youcheng Sun. "Explanations for Occluded Images". English. In: International Conference on Computer Vision (ICCV): Proceedings. IEEE, Feb. 2022. DOI: 10.1109/ICCV48922.2021.00127.

[45] Sarthak Jain and Byron C. Wallace. "Attention is not Explanation". 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. 3543–3556. DOI: 10.18653/v1/N19-1357.

[46] Sarah Wiegreffe and Yuval Pinter. "Attention is not not Explanation". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Pro-*

cessing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 11–20. DOI: 10.18653/v1/D19-1002.

[47] Michael R. Garvin, E. T Prates, M. Pavicic, P. Jones, B. K. Amos, A. Geiger, M. B. Shah, J. Streich, J. G. Felipe Machado Gazolla, D. Kainer, A. Cliff, J. Romero, N. Keith, J. B. Brown, and D. Jacobson. “Potentially adaptive SARS-CoV-2 mutations discovered with novel spatiotemporal and explainable AI models”. In: *Genome Biol* 21.1 (Dec. 2020), p. 304.

[48] Jesse Vig, Ali Madani, Lav Varshney, Caiming Xiong, Richard Socher, and Nazneen Rajani. *BERTology Meets Biology: Interpreting Attention in Protein Language Models*. June 2020. DOI: 10.1101/2020.06.26.174417.

[49] Ole Tange. “GNU Parallel - The Command-Line Power Tool”. In: *The USENIX Magazine* 36.1 (Feb. 2011), pp. 42–47. DOI: http://dx.doi.org/10.5281/zenodo.16303.

[50] Kiley R. Prilliman, Kenneth W. Jackson, Mark Lindsey, Jihua Wang, David Crawford, and William H. Hildebrand. “HLA-B15 Peptide Ligands Are Preferentially Anchored at Their C Termini”. In: *The Journal of Immunology* 162.12 (1999), pp. 7277–7284.

[51] Hans Gasser. *ImmunoBERT*. 2021. URL: https://github.com/hcgasser/ImmunoBERT (visited on 03/30/2021).

[52] William Webber, Alistair Moffat, and Justin Zobel. “A Similarity Measure for Indefinite Rankings”. In: *ACM Trans. Inf. Syst.* 28.4 (Nov. 2010). DOI: 10.1145/1852102.1852106.

[53] Jun Liu and George F Gao. “Major Histocompatibility Complex: Interaction with Peptides”. In: *eLS*. John Wiley Sons, Ltd, 2011. DOI: /10.1002/9780470015902.a0000922.pub2.

[54] Morten Nielsen, Massimo Andreatta, Bjoern Peters, and Søren Buus. “Immunoinformatics: Predicting Peptide–MHC Binding”. In: *Annual Review of Biomedical Data Science* 3.1 (July 2020), pp. 191–215. DOI: 10.1146/annurev-biodatasci-021920-100259.

[55] Alessandro Sette, A Vitiello, B Reherman, P Fowler, R Nayersina, W M Kast, C J Melief, C Oseroff, L Yuan, J Ruppert, J Sidney, M F del Guercio, S Southwood, R T Kubo, R W Chesnut, H M Grey, and F V Chisari. “The relationship between class I binding affinity and immunogenicity of potential cytotoxic T cell epitopes”. In: *The Journal of Immunology* 153.12 (1994), pp. 5586–5592.

[56] 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). DOI: 10.1186/s13073-016-0288-x.

[57] 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), pp. 51–88. DOI: 10.1146/annurev.immunol.17.1.51.

Appendix A

The Other Across-Instance Figures

A.1 HLA-A33-03

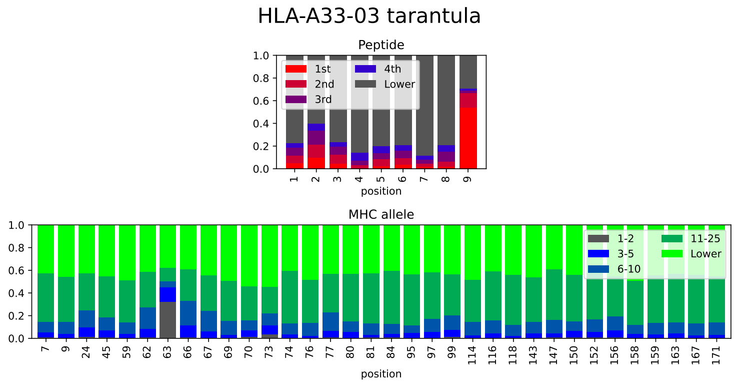


Figure A.1: Across-instance view of the rankings produced by DeepCover with Tarantula SFL for allele HLA-A33-03for the peptide (top) and the MHC pseudo sequence (bottom).

A.2 HLA-A36-01

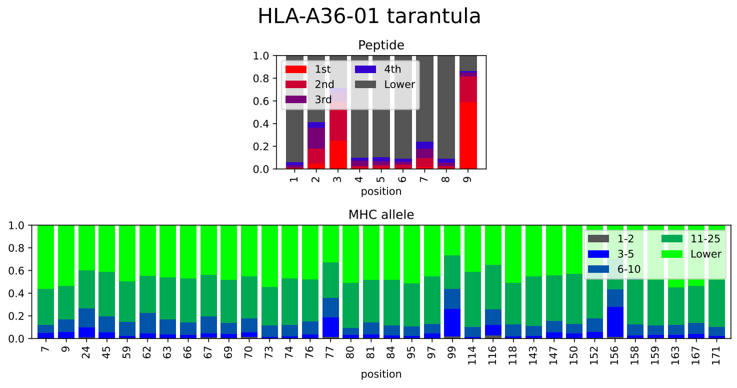


Figure A.2: Across-instance view of the rankings produced by DeepCover with Tarantula SFL for allele HLA-A36-01for the peptide (top) and the MHC pseudo sequence (bottom).

A.3 HLA-A74-01

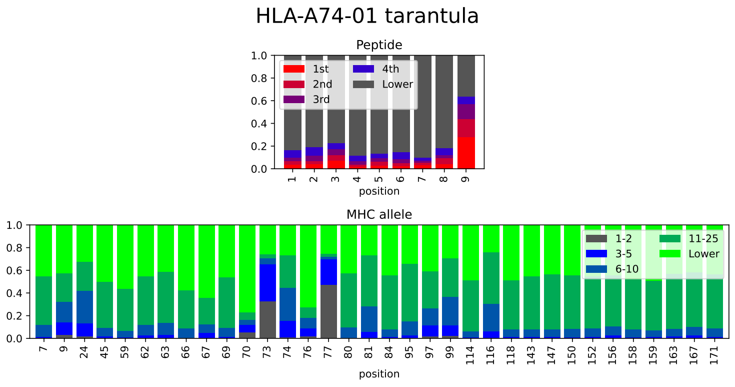


Figure A.3: Across-instance view of the rankings produced by DeepCover with Tarantula SFL for allele HLA-A74-01for the peptide (top) and the MHC pseudo sequence (bottom).

A.4 HLA-B37-01

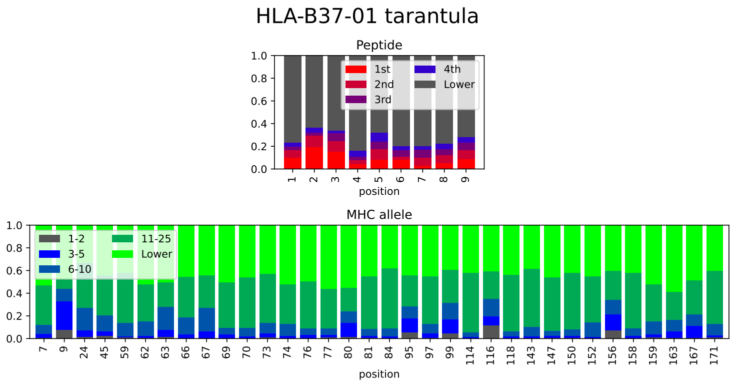


Figure A.4: Across-instance view of the rankings produced by DeepCover with Tarantula SFL for allele HLA-B37-01for the peptide (top) and the MHC pseudo sequence (bottom).

A.5 HLA-B46-01

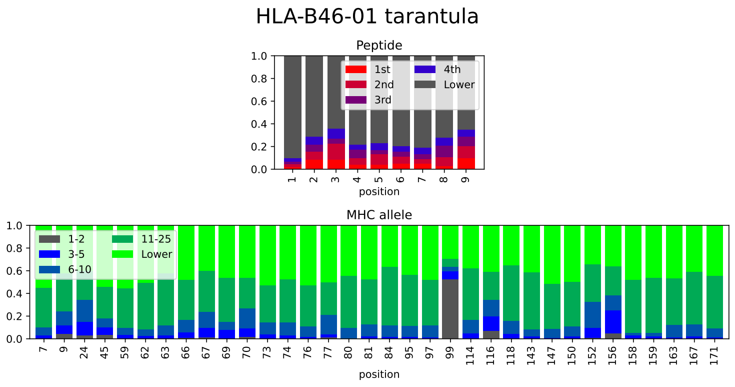


Figure A.5: Across-instance view of the rankings produced by DeepCover with Tarantula SFL for allele HLA-B46-01for the peptide (top) and the MHC pseudo sequence (bottom).

A.6 HLA-B58-01

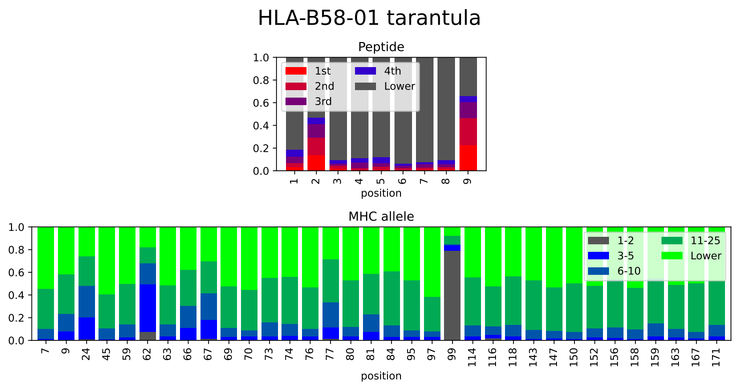


Figure A.6: Across-instance view of the rankings produced by DeepCover with Tarantula SFL for allele HLA-B58-01for the peptide (top) and the MHC pseudo sequence (bottom).

A.7 HLA-B58-02

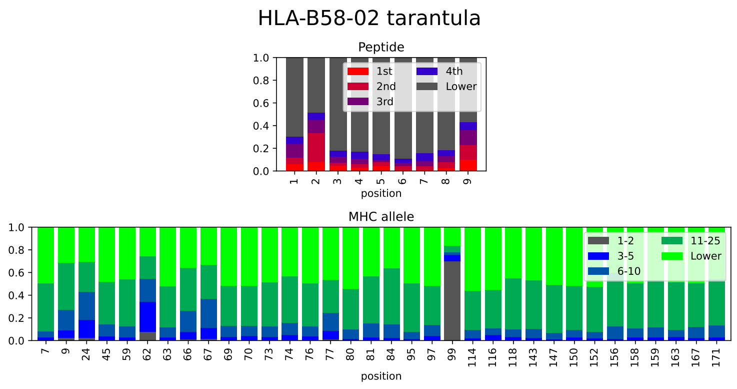


Figure A.7: Across-instance view of the rankings produced by DeepCover with Tarantula SFL for allele HLA-B58-02for the peptide (top) and the MHC pseudo sequence (bottom).

A.8 HLA-C15-02

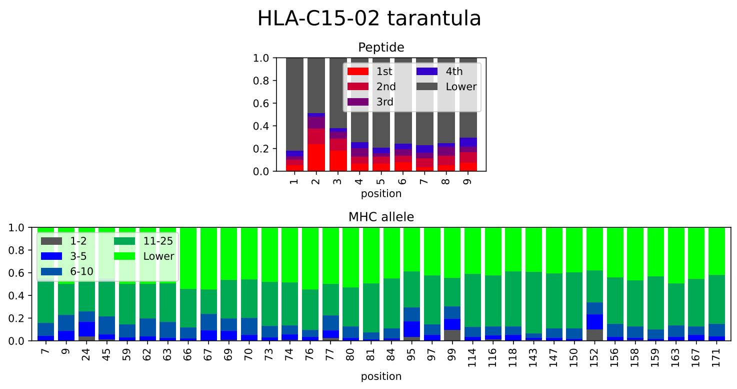


Figure A.8: Across-instance view of the rankings produced by DeepCover with Tarantula SFL for allele HLA-C15-02for the peptide (top) and the MHC pseudo sequence (bottom).

A.9 HLA-C17-01

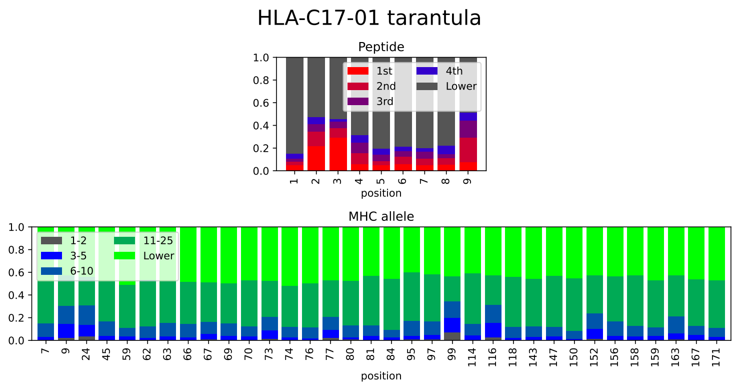


Figure A.9: Across-instance view of the rankings produced by DeepCover with Tarantula SFL for allele HLA-C17-01for the peptide (top) and the MHC pseudo sequence (bottom).

Appendix B

Across-Instance DC-Causal Figures

B.1 HLA-A33-01

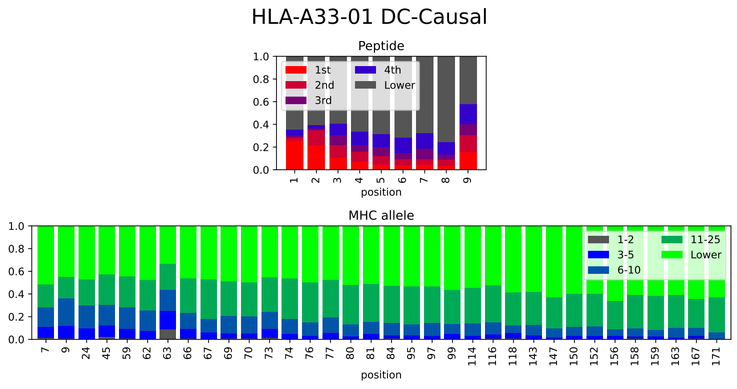


Figure B.1: Across-instance view of the rankings produced by Dc-Causal for allele HLA-A33-01for the peptide (top) and the MHC pseudo sequence (bottom).

B.2 HLA-B54-01

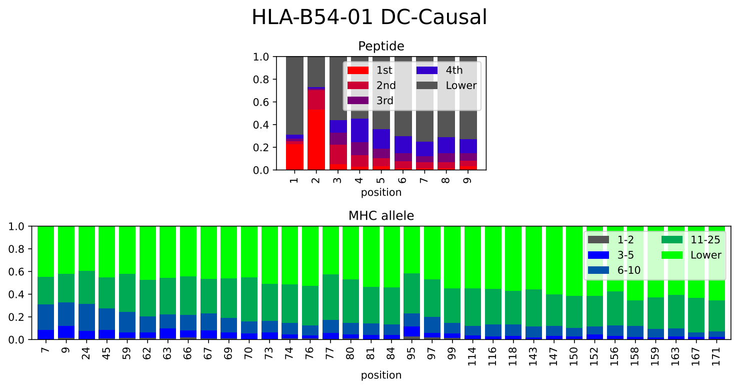


Figure B.2: Across-instance view of the rankings produced by Dc-Causal for allele HLA-B54-01for the peptide (top) and the MHC pseudo sequence (bottom).

B.3 HLA-C01-02

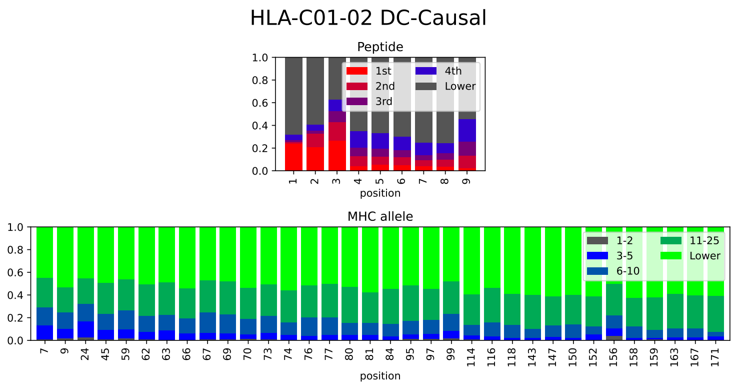


Figure B.3: Across-instance view of the rankings produced by Dc-Causal for allele HLA-C01-02for the peptide (top) and the MHC pseudo sequence (bottom).