Western Graduate&PostdoctoralStudies

Western University
Scholarship@Western

Electronic Thesis and Dissertation Repository

4-21-2020 10:30 AM

MHCherryPan, a novel model to predict the binding affinity of pan-specific class I HLA-peptide

Xuezhi Xie
The University of Western Ontario

Graduate Program in Computer Science
A thesis submitted in partial fulfillment of the requirements for the degree in Master of Science
© Xuezhi Xie 2020

Follow this and additional works at: https://ir.lib.uwo.ca/etd

Part of the Bioinformatics Commons

Recommended Citation

Xie, Xuezhi, "MHCherryPan, a novel model to predict the binding affinity of pan-specific class I HLA-peptide" (2020). *Electronic Thesis and Dissertation Repository*. 6971.
https://ir.lib.uwo.ca/etd/6971

This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact wlswadmin@uwo.ca.

Supervisor
Zhang Kaizhong
The University of Western Ontario

This dissertation/thesis is available at Scholarship@Western: https://ir.lib.uwo.ca/etd/6971

Abstract

The human leukocyte antigen (HLA) system or complex plays an essential role in regulating the immune system in humans. Accurate prediction of peptide binding with HLA can efficiently help to identify those neoantigens, which potentially make a big difference in immune drug development. HLA is one of the most polymorphic genetic systems in humans, and thousands of HLA allelic versions exist[6]. Due to the high polymorphism of HLA complex, it is still pretty difficult to accurately predict the binding affinity. In this thesis, we presented a new algorithm to combine convolutional neural network and long short-term memory to solve this problem. Compared with other current popular algorithms, our model achieved the state-of-the-art results.

Keywords: Bioinformatics, deep learning, health informatics, machine learning, HLA

Summary for Lay Audience

In recent years, deep learning has witnessed many significant breakthroughs in different areas and achieves encouraging results. Various machine learning methods have also been applied in bioinformatics fields. Due to the importance of major histocompatibility complex (MHC) in immunity, many models have been developed for MHC-peptide binding prediction. However, it is still difficult to accurately predict the MHC-peptide binding specification for all the MHC alleles now.

In this thesis, we proposed a novel model which combined convolutional neural network and long short-term memory to solve this problem. Our model has been tested with the experimental benchmark from IEDB and shows powerful performance compared with other currently popular algorithms.

Acknowlegements

To my supervisor, Dr. Kaizhong Zhang, I owe an immense debt of gratitude for his support, mentorship, scientific insights and contagious enthusiasm during my studies. His consistent, patient confidence in me was essential in the performance of the work described in this thesis. He provides much help since I came to London, I believe he considerably exceeded the expectations of his role as supervisor.

I would like to thank Yuanyuan Han for her encouragement and advice during this research and the life in Canada.

I would also like to show special gratitude to my parents for their constant support throughout my research and life.

Contents

Abstract **i**

Summary for Lay Audience **ii**

Acknowlegements **iii**

List of Figures **vii**

List of Tables **x**

1 Introduction **1**

1.1 Biological Background 1

1.2 Research Question 4

1.3 Difficulty 4

1.4 Machine Learning in MHC-binding Problem 5

1.5 Contributions 6

1.6 The Structure of this Thesis 7

List of Appendices **1**

2 Machine learning and related works **8**

2.1 Machine learning algorithms on binding predictions 8

2.1.1 Artificial Neural Networks 8

2.1.2 Feedforward Neural Network and Convolutional Neural Networks . . . 10

2.1.3 Recurrent Neural Networks 11

2.2 Models on HLA-peptide binding prediction 12

2.2.1 Allele-specific Models 13

2.2.2 Pan-specific Models 15

2.2.3 Consensus Models 18

2.3 Our Novel Design 19

3 Methodology **21**

3.1 Problem statement 21

3.2 Encoding 21

3.2.1 Genetic Codes for Amino Acids 22

3.2.2 Encoding Method 22

3.3 Model 24

3.3.1 Peptide Feature Extraction Module 24

3.3.2 HLA Feature Extraction Module 26

3.3.3 Affinity Prediction Module 27

3.4 Pipeline 28

3.5 Model Training 29

4 Experiment setup **31**

4.1 Dataset 31

4.1.1 EBI 31

4.1.2 IEDB 32

4.2 Data pre-processing and pre-analysis 33

4.2.1 Peptide-HLA data pre-processing 33

4.2.2 HLA Sequences 34

4.2.3 Data pre-analysis and visualization 35

4.2.4 The Amino Acid Enrichment for peptide-hla binding data 38

4.2.5 Encoding . . . 38
4.3 Model Learning . . . 41
4.3.1 Model training and preventing overfitting . . . 41
4.3.2 Implementation . . . 42
4.4 Performance Evaluation . . . 42
4.4.1 Test Benchmark Dataset . . . 43
4.4.2 Metrics and Mathematical Function . . . 43
4.5 Technologies and Tools . . . 47

5 Result and discussion 49
5.1 Experimental results on different encoding metrics . . . 49
5.1.1 BLOSUM Matrix . . . 49
5.1.2 Results on different blossom matrix . . . 51
5.2 Experimental results on Benchmark testing dataset . . . 52
5.2.1 Average AUC and SRCC . . . 55

6 Conclusion 56
6.1 Conclusion . . . 56
6.2 Future Work . . . 57

Bibliography 58

A MHCherryPan Model Summary 65

Curriculum Vitae 67

List of Figures

1.1 **MHC Class I Processing.**[38]: Proteins are degraded into peptides by the proteasome. peptides convey into the endoplasmic reticulum, where they combined with MHC-I molecules. MHC-I/peptide complexes enter Golgi apparatus, are glycosylated, enter secretory vesicles, fuse with the cell membrane, and externalize on the cell membrane interacting with T lymphocytes 2

1.2 **The structure of MHC class I molecules.**[38] 3

1.3 **Polymorphism and Polygeny of MHC molecules.** Taken from the source: Immunobiology, Garland Science[13]. Picture 1.3a(polymorphism) shows the expression of MHC alleles is codominant and most individuals are likely to be heterozygous at each locus. Therefore, two combinantions (shows as the colors) can be found as the result. Picture1.3b indicates the polygene where three genes could leads to 3 combination. Picture 3c (Polymorphism and Polygeny) represents that after combining polymorphism and polygenes, it could lead to 6 different combinantions. 5

1.4 **Application of deep learning in bioinformatics.** taken from the source: Deep learning in bioinformatics, Min et al. [24] 6

2.1 **Artificial Neural Networks[41]** 10

2.2 **The illustration of Feed Forward Neural Network and Convolutional Neural Network. Reproduced from original paper[16]**: (A) Feed forward network. (B) Convolutional neural network. (C) A filter can be visualized as a sequence motif. 11

2.3 **Basic recurrent neural network[41]** 12

2.4 **MHCflurry Flow Network. Reproduced from original paper [28]**: Neural network architecture 14

2.5 **The example of insertion and deletion for peptide. Reproduced from original paper [2]**: Left(a) is the insertion example for peptide ”AILDFTHL” and the right(b) is the deletion example for peptide ”FYGERPLTRY” 15

2.6 **Interaction map of the HLA pseudo sequence in NetMHCPan. Reproduced from original paper[19]** 16

2.7 **PickPocket Flow Network. Reproduced from original paper [53]** 18

3.1 **BLOSUM62 Matrix**: The BLOSUM62 matrix is used to encode peptide and HLA sequences. 24

3.2 **Peptide Feature Extractor - RNN** 25

3.3 **HLA Feature Extractor - CNN** 26

3.4 **MHCherryPan Network Architecture**: (I) Peptide Feature Extraction. (II) MHC Feature Extraction. (III) Affinity Prediction. (IV) Components of a convolutional Block 28

3.5 **Model Training**: Forward and Backward Propagation, replicated from original article[15]. 29

4.1 **MHC Training Data** 33

4.2 **The statistics summary for training dataset** 34

4.3 **Pie chart of binding or not for MHC Training Data.**[38]: The blue part is for ”not binding” data and the orange part is for ”binding” data. 35

4.4 **Training data distribution by peptide length.** 36

4.5 **Binding Affinity Distribution by Alleles** 37

4.6 **Motif analysis for 9-mer peptides**[38]: showing the percentage of enriched amino acid at each position for hla-peptide binding data. 39

4.7 **Motif analysis for 10-mer peptides** showing the percentage of enriched amino acid at each position for hla-peptide binding data. 40
4.8 **Job script for Model training** 42
4.9 **AUC Curve.**: Taken from Receiver operating characteristic - Wiki [40] 44

5.1 **Blosum Matrix:** from top to bottom: blosum 60, blosum 62, blosum 70, blosum 80, and blosum 90. The matrix is downloaded from `ftp://ftp.ncbi.nlm.nih.gov/blast/matrices/` 50
5.2 **Average AUC and SRCC Bar Chart** 55

A.1 Summary for MHCherryPan Model 66

List of Tables

2.1 Summary of related works for predicting MHC-peptide ligands. 19

5.1 Validation Experiment results for the MHCherryPan model trained on IEDB dataset. 51

5.2 Performance Evaluation on IEDB Benchmark Database 53

5.2 Performance Evaluation on IEDB Benchmark Database 54

Chapter 1

Introduction

In this chapter, we introduce the background of major histocompatibility complex, especially its function in immunology, the importance of the research question, and our contribution to solve it. The thesis layout is illustrated at the end of this chapter.

1.1 Biological Background

The major histocompatibility complex (MHC) proteins are critical for the adaptive immune system recognizing foreign molecules in most mammalian species. MHC are the chaperones for endogenous peptides which are transferred to the cell surface through binding with MHC (the MHC-peptide complex) as the antigens for T cell recognition. Essentially, T cells would tolerate the auto-antigens, but activate when the allo-antigens appear [38]. Those MHC proteins are called human leukocyte antigen (HLA) complex in human beings, and they help our immune system to distinguish pathogens (foreign molecules) through binding to peptide antigens and exposing them to the surface of cells for recognition by certain T cells. Research on the canonical mechanism of MHC-peptide binding can benefit our comprehension of which peptides bind to MHC (potentially resulting in an immune response). This could be in favor for cancer treatments. Specifically regarding the mutated or tumor cells, some peptides originates from altered tumor proteins and they are known as neoantigens. Those neoantigens would be the ideal antigens for the development of protein-based vaccines and drugs to treat tumor patients[22]. Recent advances in cancer immunotherapy have confirmed this new treatment boosts T cell activity to destroy cancer cells in both mouse models and human patients[20].

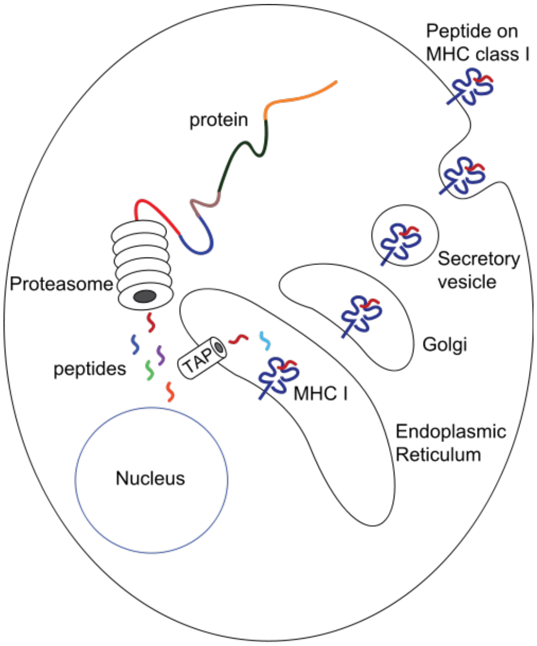


Figure 1.1: **MHC Class I Processing.**[38]: Proteins are degraded into peptides by the proteasome. peptides convey into the endoplasmic reticulum, where they combined with MHC-I molecules. MHC-I/peptide complexes enter Golgi apparatus, are glycosylated, enter secretory vesicles, fuse with the cell membrane, and externalize on the cell membrane interacting with T lymphocytes

MHC can be further divided into three subgroups: MHC class I, MHC class II, and MHC class III. Among those, MHC class I and MHC class II attract more research attention due to they directly involve in antigen presentation of immunology[14]. Class I MHC binds to peptides from intracellular proteins, while class II MHC binds to extracellular peptides that are brought inside the cell. Figure 1.1 illustrates the MHC class I pathway. MHC class I plays an irreplaceable role in cell-mediated immunity, which could resolve in intracellular pathogens such as viruses and some bacteria. MHC class I mediates the destruction of infected or malignant host cells (cellular immunity) by interacting with CD8 molecules on surfaces of cytotoxic T cells. In human beings, the MHC complex is also called the human leukocyte antigen (HLA) system. There are three class I in humans: HLA-A, HLA-B and HLA-C.

As shown in Figure 1.2, MHC class I molecules are heterodimers, which are made up of two polypeptide chains - α and β_2-microglobulin (b2m) and the two chains are linked through the interaction between b2m and the $\alpha 3$ domain[38]. Regarding those chains, the α chains are polymorphic and encoded by MHC genes, while the b2m subunit is not polymorphic and encoded by the β_2-microglobulin gene[38]. The $\alpha 3$ domain is plasma membrane-spanning and interacts with the CD8 co-receptor of T-cells. The $\alpha 3$-CD8 interaction holds the MHC I molecule in place while the T cell receptor (TCR) on the surface of the cytotoxic T cell binds its $\alpha 1$-$\alpha 2$ heterodimer ligand, and checks the coupled peptide for antigenicity[38]. The $\alpha 1$ and $\alpha 2$ heavy chain domains fold to form the active binding pocket for short antigenic peptide peptides to bind, and the peptides binding to MHC class I molecules are mostly in 8-10 amino acid in length[5]. Those $\alpha 1$, $\alpha 2$, and $\alpha 3$ domains create a globular protein in which a β-pleated sheet forms the floor of the peptide-binding groove and is bounded by two helical regions that form the sides of the groove[8].

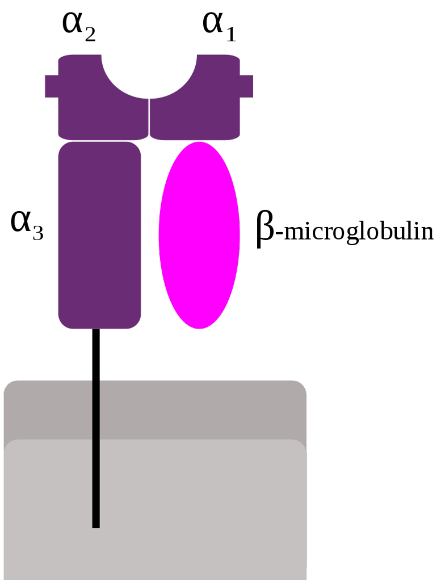


Figure 1.2: **The structure of MHC class I molecules.**[38]

1.2 Research Question

In this thesis, we concentrate on predicting peptide binding to MHC/HLA through deep learning methods. In particular, we try to solve peptide-MHC binding prediction problems by using Recurrent Neural Network and VGG-based Convolutional Neural Network, and then compare our model performance with other currently popular models.

1.3 Difficulty

Accurate prediction of peptides binding to MHC still faces a big challenge. The difficulty could refer to two unique properties of MHC: Polygenes and polymorphism. Polygenes mean each individual possesses several different MHC genes, which results in diverse range of peptide-binding specificities. The polymorphism of MHC stands for that the organism differs diversely from each other within the same species[14]. Many different alleles exist inside a population[38]. The MHC genes are the most polymorphic genes, as we know so far. The polymorphism is so high, no two individuals have exactly the same set of MHC molecules except for the twins[38]. Those properties lead to experimental characterization for peptide-MHC binding costly and time-consuming. Therefore, in silico computational approaches have been proposed to study this research topic, especially when large amounts of in vitro binding affinities data are available in public databases like the Immune Epitope Database (IEDB)[9, 32]. Although there are over a half million of experimentally characterized binding affinities data in the IEDB database, they are not evenly distributed across the alleles. The experimental data could be as many as 10000 characterized peptide-MHC binding affinities for common alleles, while it may be as few as 100 for other rare alleles. Besides, most peptides in MHC class I ligands data represented as 9 amino acids (aa) in length, but it also varied critically among MHC alleles. For example, the H-2Kb allele has a preference for 8 aa-long peptides, while HLA-A*01:01 has nearly one-third of MHC-presented peptides which have a length of 9 aa.

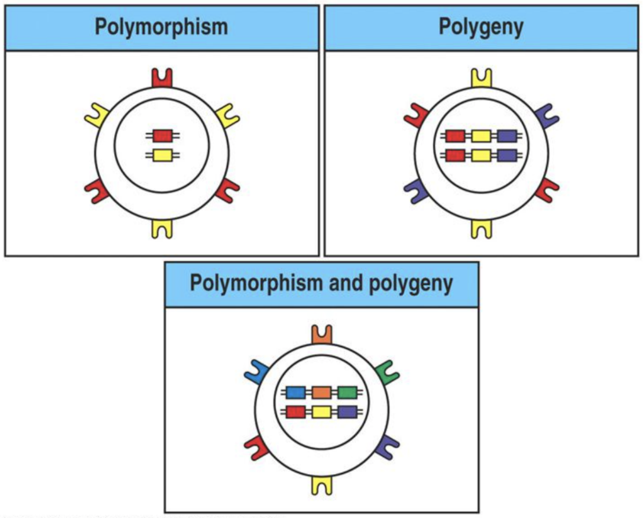


Figure 1.3: **Polymorphism and Polygeny of MHC molecules.** Taken from the source: Immunobiology, Garland Science[13]. Picture 1.3a(polymorphism) shows the expression of MHC alleles is codominant and most individuals are likely to be heterozygous at each locus. Therefore, two combinantions (shows as the colors) can be found as the result. Picture1.3b indicates the polygene where three genes could leads to 3 combination. Picture 3c (Polymorphism and Polygeny) represents that after combining polymorphism and polygenes, it could lead to 6 different combinantions.

1.4 Machine Learning in MHC-binding Problem

In recent years, various machine learning methods have been applied in bioinformatics fields. Figure 3.3 indicated the general process of deep learning in bioinformatics. Inside those methods, due to the importance of MHC in immunity, many models have been developed for MHC-peptide binding prediction to either predict the binding affinity (a regression problem) or to predict whether a peptide could bind to the MHC allele (a classification problem)[3]. Among all of the models, studies on predicting peptide binding to HLA class I molecules have been more active due to the availability of the large-scale high-quality peptide-HLA binding data as provided by IEDB database, and the automated benchmark platform for comparing, tracking, and scoring the progress of several current popular online computational methods[34]. There are mainly three categories of models for HLA-peptide binding prediction: allele-specific, consensus, and pan-specific models[27, 29, 53, 11].

Most of these models make full use of machine learning algorithms, such as artificial neu-

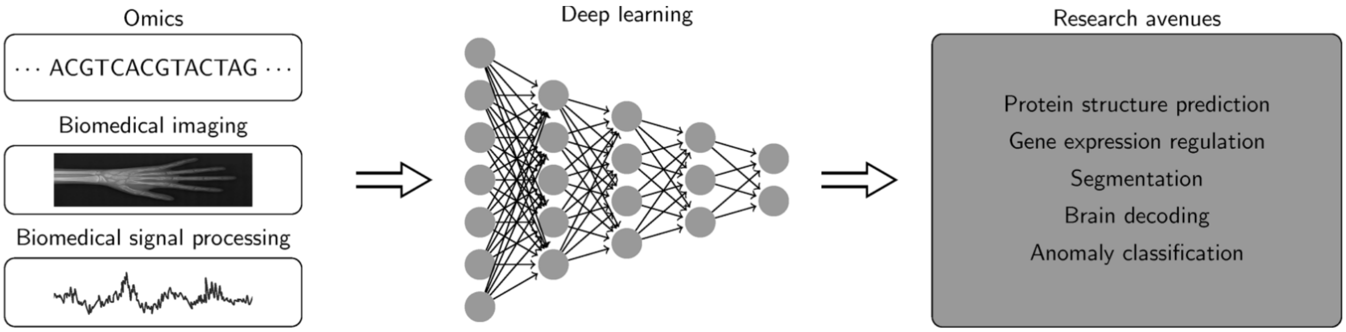

Omics
· · · ACGTCACGTACTAG · · ·
Biomedical imaging
Biomedical signal processing
Deep learning
Research avenues
Protein structure prediction
Gene expression regulation
Segmentation
Brain decoding
Anomaly classification

Figure 1.4: **Application of deep learning in bioinformatics.** taken from the source: Deep learning in bioinformatics, Min et al. [24]

ral networks (ANN), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN), to extract the motifs features. Machine learning algorithms can build complex nonlinear models to achieve outstanding prediction performance in the HLA-peptides binding affinity prediction[51].

1.5 Contributions

In this thesis, we propose a new model to solve HLA-peptide binding prediction problems. Our model is developed based on pan-specific models and combines the strengths of CNN and RNN architectures. Compared with other currently popular models, including NetMHCPan3.0, NetMHCPan4.0, PickPocket, SMM, NetMHC3.4, NetMHC4.0, ARB, MHCflurry, AMMPM-BEC, IEDB Consensus and netMHCcons, our model achieved state-of-the-art results on various HLA alleles with excellent generalization performance.

The main contribution of this thesis is as follows:

- We proposed a well-performed binding probability prediction model called MHCherry-Pan, which achieved the state-of-the-art results on a large number of HLA alleles.
- The novel design which combines CNN-based and RNN-based extractors in one pan-specific model enables our model to deal with any length peptide sequences and also have good generalization capability.

- Our model takes raw sequential data as input, which could be potentially used in any RNA-protein, DNA-protein, and RNA-DNA binding prediction.

1.6 The Structure of this Thesis

The rest of the thesis is organized as follows. In Section 2, we provide an overview of the related work. In Section 3, we describe our new approach - MHCherryPan for predicting HLA-peptide binding. In Section 4, we discuss the experiment setup, including the datasets that we use and some experiment details. In Section 5, we compare our new model with other popular models. Section 6 concludes this paper.

Chapter 2

Machine learning and related works

In this section, we first introduce the frequently-used machine learning algorithms in the binding-prediction problems and then we would focus on the previous work done for HLA-peptide binding prediction, which can be classified into allele-specific, pan-specific, and consensus models. In the end, we will summary the related works and introduce our model.

2.1 Machine learning algorithms on binding predictions

Currently, the neural network is the dominant machine-learning algorithm for peptide-binding prediction problems [23]. Among different fundamental architectures of neural networks for binding prediction algorithms, Artificial Neural Networks, Convolutional Neural Networks and Recurrent Neural Networks are the most popular deep learning methods which have all been applied and achieved inspiring results.

2.1.1 Artificial Neural Networks

Deep learning, as a member of the broader family of machine learning methods, is based on artificial neural networks with representation learning [44]. The reason why it is so-called is due to the learning complexity. Generally speaking, deep learning model contains much deeper layers such as two or more hidden layers whereas shadow learning usually only contain one layer. Deep learning have been applied to fields such as speech recognition, computer vision, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced inspiring results which could be comparable or surpassing human performance[44].

Artificial Neural Network is an information processing model acquire, store and utilize experimental knowledge and it is is inspired by the way biological nervous systems, such as the brain, process information[55] . Artificial Neural Networks (ANN), as one of the most popular deep learning algorithms, have been successfully applied on several significant bioinformatic topics such as DNA-binding and RNA-binding prediction as well as HLA-peptides binding predictions[1]. An ANN is usually consisted of neurons that are connected to each other, and ANN has three general types of layers: the input layer, the hidden layer, and the output layer, as shown in figure 2.1. The input layer brings the initial data such as variables or features of a given problem into the system for further processing by subsequent layers. These inputs are then propagated through the hidden layers to the output layer. The hidden layers perform a series of (non)linear functions that convert the input data into data that the output layer can use. This conversion is often called feature extraction since the network learns which input data is more important than others.

ANN can capture the complex inter-relationships in the non-linearity among data variables and has been used in classification and recognition tasks as well as motifs feature extraction[51]. Most of the ANN-based prediction models on HLA-peptides binding affinity prediction, such as netMHC [21], and MHCflurry[28], only include one or two full-connected layers, with the optimizing network structures and parameters for different MHC alleles. Although ANN-based approaches are remarkable for its accuracy, they lose flexibility due to they limit the input length of peptide epitopes (8-15 amino acids for MHCflurry) and support only specific MHC alleles that the models have been trained for[16].

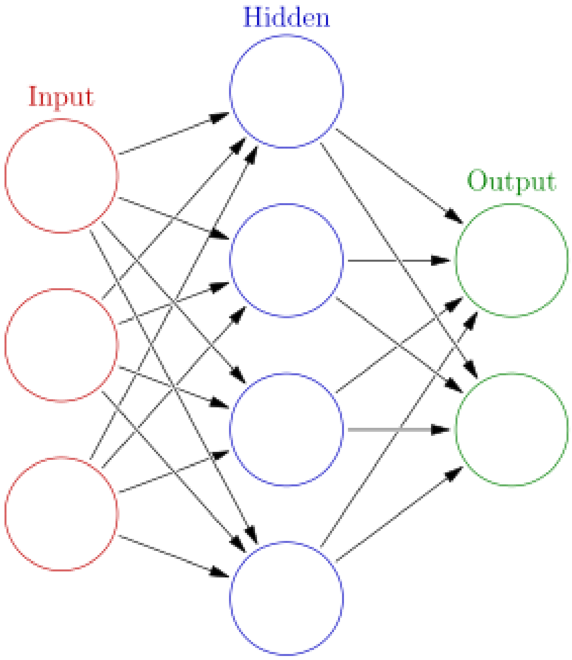


Figure 2.1: **Artificial Neural Networks[41]**

2.1.2 Feedforward Neural Network and Convolutional Neural Networks

A feedforward neural network is an artificial neural network in which connections between the nodes do not form a cycle[52]. As the name indicates, it is a network where the information flows in one single direction. Information spreads from one layer to the next, starting at the input nodes, through the hidden nodes, and ending in the output nodes. A non-linear so-called activation function, most often tanh or sigmoid, is applied to give the output of a neuron given its input. Figure 2.2A indicated the architecture of the feedforward neural network. The hidden layer of a feedforward neural network is also referred to as a dense layer because it is fully connected to the previous layer.

Convolutional Neural Networks (CNNs) are a type of deep feedforward neural network. The name ”convolutional Neural Networks” is due to that the network utilized a mathematical operation called convolution, which is a specialized kind of linear operation [43] as shown in Figure 2.2b. Convolutional networks are neural networks that use convolution instead of the general matrix multiplication in at least one of their layers[43]. CNNs usually consist of

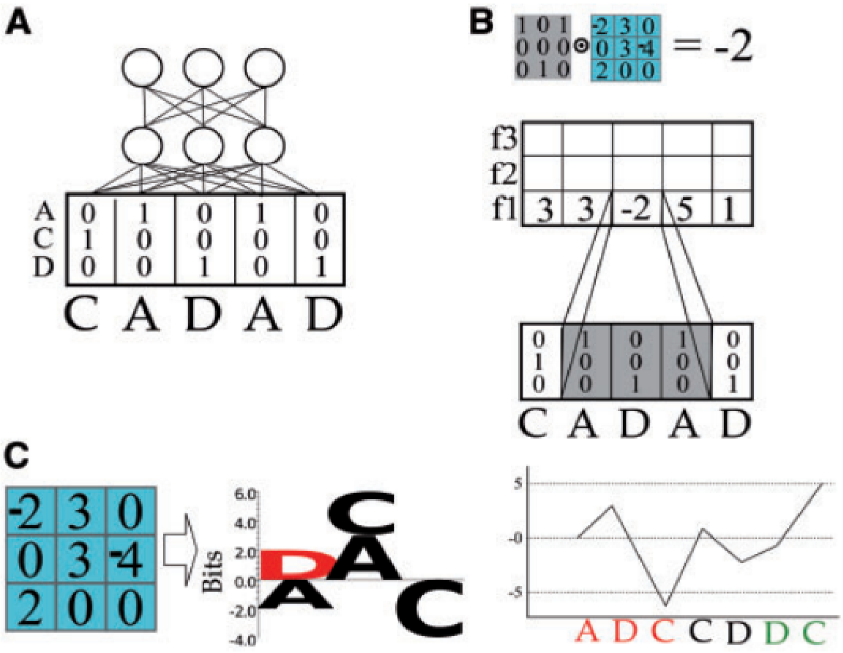


Figure 2.2: **The illustration of Feed Forward Neural Network and Convolutional Neural Network. Reproduced from original paper[16]**: (A) Feed forward network. (B) Convolutional neural network. (C) A filter can be visualized as a sequence motif.

multiple convolutional layers with many convolutional filters and pooling layers (the function is to downsample an input representation), which makes the model able to learn many highly abstract features[18]. CNN have achieved amazing result in some fields especially in image classification. In the field of HLA binding, the HLA-CNN[35] , which uses three convolutional layers and two fully- connected layers with word embedding for encoding, achieves good performance and outperforms all traditional prediction methods. DeepSepPan[19] utilized the VGG-liked deep CNN to extract abstract features from HLA. Although it also achieves competitive performance compared with other current algorithms, DeepSeqPan loses flexibility and only supports the peptides which have 9 aa length.

2.1.3 Recurrent Neural Networks

A Recurrent Neural Network (RNN), as shown in figure 2.3, is a special type of neural network because (unlike feedforward neural networks) it contains directed cycles. RNN layers use the

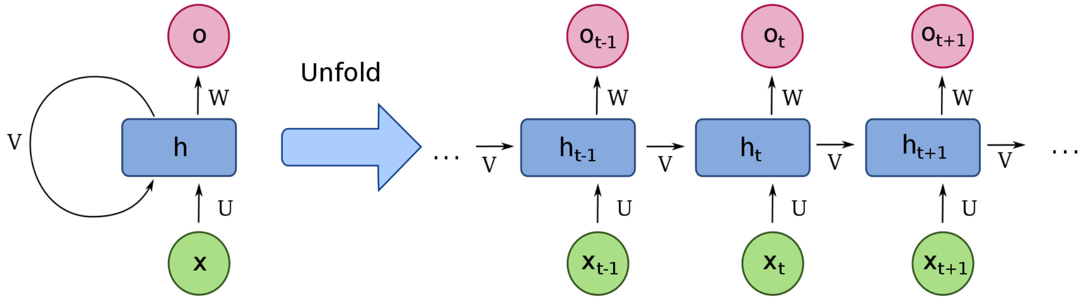


Figure 2.3: **Basic recurrent neural network[41]**

output of the network at a certain iteration as input for a next iteration. This recursive aspect of RNN make them extremely useful for sequential data, especially if the output of the network for a certain element depends on both that element and the output of the previous element(s). RNN has been used to encode time-series information in many tasks due to that RNN can better capture temporal relationships by remembering previous inputs[16]. A popular choice for RNN is the Long Short-Term Memory (LSTM), which alleviates the vanishing and exploding gradient problems presented in normal RNN[16]. MHCpred[10] with the structure of deep RNN, which applies three LSTM layers and adaptively learns the appropriate parameters to enable the model to learn the hidden features more efficiently. Another big merit of RNN is that there is no limit for the peptide length due to RNN utilizes the masking layers, so it doesn't require fixed-length inputs. In this case, those RNN-based prediction models could be very flexible in the input length.

2.2 Models on HLA-peptide binding prediction

In order to accurately predict peptides binding MHC and identify ideal neoanitgen for protein-based vaccine development, many models have been developed during the last several years. There are different ways to classify those models. For example, based on the result the model generates, they could be divided into regression model (predicting binding affinity) or classification models(predicting binding or not). If we focus more on the algorithms or model themselves, most of the models for HLA-peptide binding prediction can be divided into three categories: allele-specific, consensus, and pan-specific models[27, 29, 53, 11].

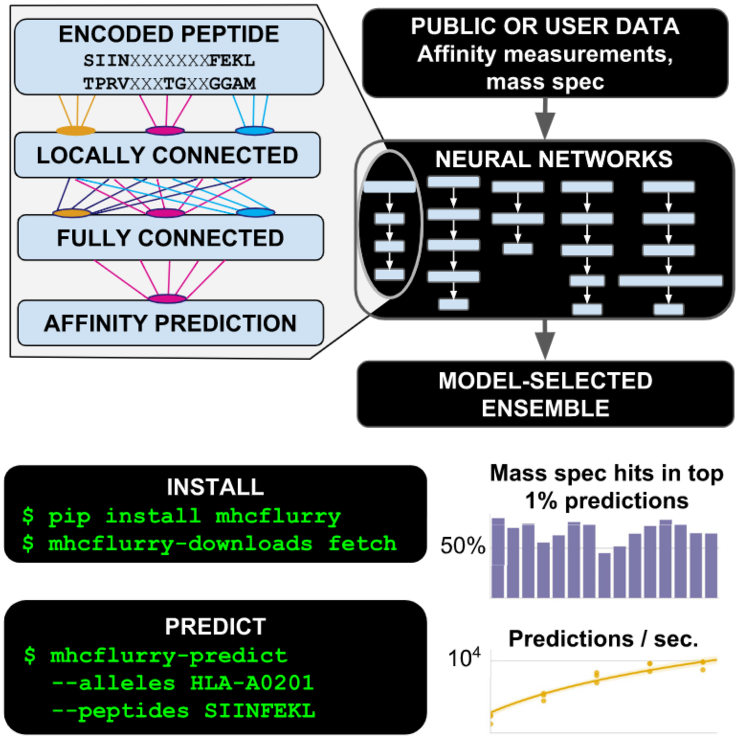
2.2.1 Allele-specific Models

Allele-specific models have been proposed to predict the specificity of one receptor (one HLA Allele) based on training data regarding this one HLA allele. In allele-specific models, each HLA allele would have a separate allele-specific prediction model. MHCflurry is a recent proposed allele-specific model that achieved state-of-the-art results compared with other algorithms[28].

MHCflurry

MHCflurry, as one of the most well-performed allele-specific model, is developed using separate artificial neural networks (ANNs) for each supporting allele, which consists of locally connected layers and a fully connected layer [28]. As mentioned previously that allele model need to have separate prediction model for each allele, MHCflurry designed an ensemble of 8-16 neural networks for each supported MHC allele trained on affinity measurements from IEDB and other sources[28].

Figure 2.4 indicates the flow chart for the MHCflurry model. As the figure 2.4 shows, peptide data sequences are firstly padding into 15 amino acids by adding a ‘no-residue’ character (‘X’). The reason why they padding through the mid of the peptide sequences instead of regular padding method (padding to the beginning or the end) is to maintain the position of the anchor residue (please also see in section 3.4 prepossessing). After padding, the peptide sequence is then encoded into a tensor with dimension size of 15 x 21 by using BLOSUM 62 matrix. Regarding the activation function for the model, MHCflurry uses the hyperbolic tangent function for the hidden layers and the sigmoid function for the final output layers. MHCflurry had separate networks trained for each supported MHC allele, and each network contains the locally-connected layers, fully-connected layers and an affinity predictor.



```
$ pip install mhcflurry
$ mhcflurry-downloads fetch
```

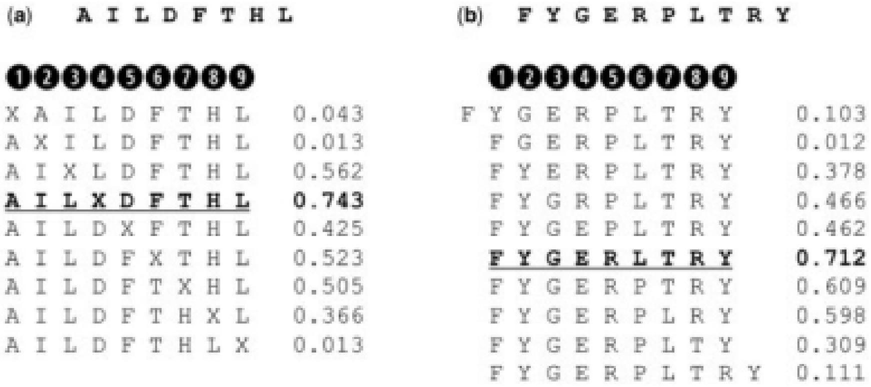
```
$ mhcflurry-predict
  --alleles HLA-A0201
  --peptides SIINFEKL
```

Figure 2.4: **MHCflurry Flow Network. Reproduced from original paper [28]**: Neural network architecture

NetMHC 4.0

NetMHC 4.0 is another popular allele-specific model, which is also developed using artificial neural networks (ANNs). NetMHC 4.0 develops an ensemble method to generate the feed-forward networks and assigns the binding core of a given peptide based on the majority vote of the networks in the ensemble [2]. NetMHC 4.0 uses both sparse encoding matrix (a matrix in which most of the elements are zero such as one-hot key) and BLOSUM62 (see explanation in 3.2.2) to encode the peptide sequences into nine amino acid-binding cores.

But unlike MHCflurry padding all peptide sequence into 15 amino acids, NetMHC 4.0 developed a novel deletion-and- insertion methods, which is used to reconcile peptide sequence into a core of nine amino acids. As figure 2.5 shows the example of insertion and deletion for peptide. Regarding the peptide which the sequence length is 8 or 10, the NetMHC would first reconcile into the 9-amino acid peptides by inserting or deleting at all possible positions, which results in multiple potential sequences. After that, it will select the highest predicted score with the current configuration of the neural network is taken as the optimal binding core[2] . In addition, NetMHC does not public their detailed implementation and code.



(a) A I L D F T H L

①②③④⑤⑥⑦⑧⑨	
X A I L D F T H L	0.043
A X I L D F T H L	0.013
A I X L D F T H L	0.562
A I L X D F T H L	**0.743**
A I L D X F T H L	0.425
A I L D F X T H L	0.523
A I L D F T X H L	0.505
A I L D F T H X L	0.366
A I L D F T H L X	0.013

(b) F Y G E R P L T R Y

①②③④⑤⑥⑦⑧⑨	
F Y G E R P L T R Y	0.103
F G E R P L T R Y	0.012
F Y E R P L T R Y	0.378
F Y G R P L T R Y	0.466
F Y G E P L T R Y	0.462
F Y G E R L T R Y	**0.712**
F Y G E R P T R Y	0.609
F Y G E R P L R Y	0.598
F Y G E R P L T Y	0.309
F Y G E R P L T R Y	0.111

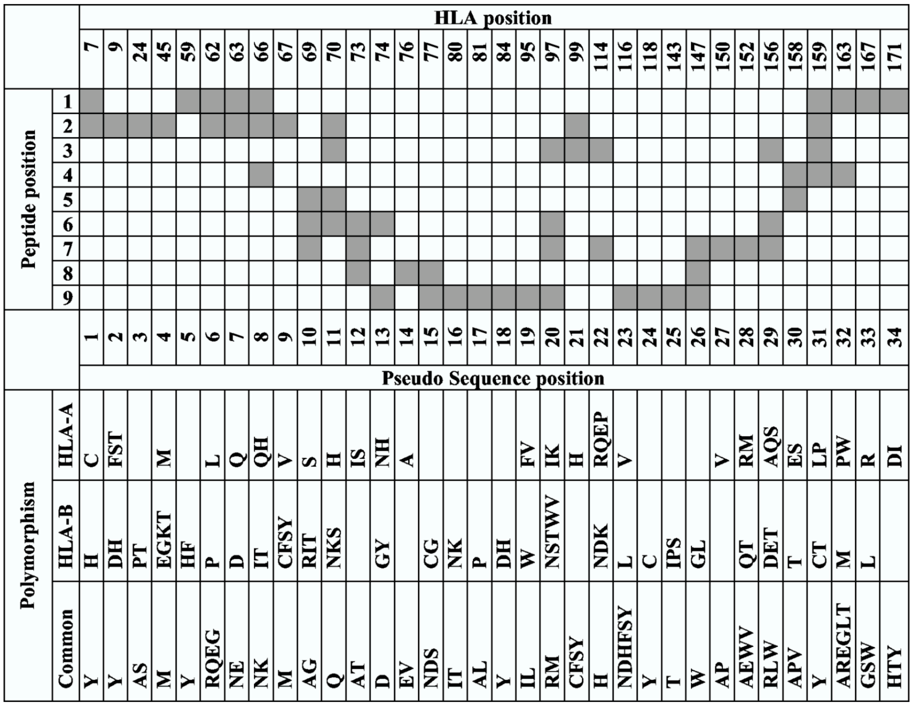
Figure 2.5: **The example of insertion and deletion for peptide. Reproduced from original paper [2]**: Left(a) is the insertion example for peptide ”AILDFTHL” and the right(b) is the deletion example for peptide ”FYGERPLTRY”

2.2.2 Pan-specific Models

Although those allele-specific models usually have high excellent performance on some common HLA alleles which have large amounts of experimental data but predict poorly on other alleles with few experimental data. This will cause a big issue since many alleles only have a few experimental data available. Therefore, researchers proposed the pan-specific models to solve this data scarcity issue.

The main idea of the pan-specific model is to use the data from all the alleles of HLA for model training so that it could predict the binding results of the HLA alleles, which may have less experimental data or do not appear at all. The pan-specific model would encode both the peptides and the HLA alleles protein as inputs. The peptide-HLA binding environment is represented in this way so the deep learning models can be trained on all peptide-HLA binding data[54].

NetMHCPan



		HLA position																																	
		7	9	24	45	59	62	63	66	67	69	70	73	74	76	77	80	81	84	95	97	99	114	116	118	143	147	150	152	156	158	159	163	167	171
Peptide position	1	▨				▨	▨	▨	▨																							▨	▨	▨	▨
	2	▨	▨	▨	▨		▨	▨	▨	▨		▨										▨										▨			
	3											▨									▨	▨	▨							▨		▨			
	4								▨																						▨	▨	▨		
	5										▨	▨																			▨				
	6										▨	▨	▨	▨							▨									▨					
	7										▨		▨								▨		▨				▨	▨	▨	▨					
	8												▨		▨	▨											▨								
	9													▨		▨	▨	▨	▨	▨	▨			▨	▨	▨	▨								
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
		Pseudo Sequence position																																	
Polymorphism	HLA-A	C	FST		M		L	Q	QH	V	S	H	IS	NH	A					FV	IK	H	RQEP	V				V	RM	AQS	ES	LP	PW	R	DI
	HLA-B	H	DH	PT	EGKT	HF	P	D	IT	CFSY	RIT	NKS		GY		CG	NK	P	DH	W	NSTWV		NDK	L	C	IPS	GL		QT	DET	T	CT	M	L	
	Common	Y	Y	AS	M	Y	RQEG	NE	NK	M	AG	Q	AT	D	EV	NDS	IT	AL	Y	IL	RM	CFSY	H	NDHFSY	Y	T	W	AP	AEWV	RLW	APV	Y	AREGLT	GSW	HTY

Figure 2.6: **Interaction map of the HLA pseudo sequence in NetMHCPan. Reproduced from original paper[19]**

NetMHCPan is the first pan-specific MHC-peptide binding prediction algorithm and it achieves state-of-the-art result in prediction performance [27]. One of the significant contributions of NetMHCPan is that they designed a novel pseudo sequence to stand for the HLA binding motifs. In their method, an HLA sequence is reduced to a pseudo amino acid sequence of length 34, and each amino acid in this pseudo sequence is selected if it is in contact with the peptide within 4.0 A [27].

The interaction map in Figure 2.6 is extracted based on a representative set of HLA structures with nonamer peptides, and this extracted 34-length pseudo sequence is a list of location indexes of amino acids in the HLA sequence[27]. Therefore, NetMHCPan represents each MHC sequence as a 34 corresponding residues, and an HLA-peptide binding sample is represented as a 43-length amino acid sequence (34 from the HLA and 9 from the peptide). The input is then used to train multiple feedforward neural networks, and the network with the highest prediction performance on the evaluation set was selected as the final prediction model[27].

PickPocket

PickPocket is another popular pan-specific model for predicting peptide-MHC binding. The main idea of PickPocket is to use statistical approach to calculate the binding score for each candidate peptide sequence. For each supported HLA alleles, amino acids that frequently occur at anchor positions (the positions where certain amino acids appear more) given the higher value. The less frequent amino acids are assigned with lower values [53]. The final score of a sequence is the sum of values at each position.

PickPocket constructs position-specific scoring matrices (PSSMs) for each MHC alleles, which are derived from peptide ligands in the training data. In order to support MHC alleles with insufficient data, PickPocket developed a novel pocket-specific binding method. These pockets are derived form specificity determining interactions with amino acid side chains distributed along the length of the ligand [53]. Therefore, pocket library could be built from MHC alleles for where a significant amount of peptide-binding ligands data was available to construct a PSSM for those rare HLA alleles which have insufficient experimental data. In order to calculate the similarity between the a rare MHC allele and the MHC allele with a calculated PSSM from the training dataset, the similarity function is used as following:

$$Sim(s_q, s_i) = \frac{S(s_q, s_i)}{\sqrt{S(s_q, s_q)S(s_i, s_i)}} \tag{2.1}$$

S(s_q, s_i) indicates the similarity score between the two pocket sequences Sq and Si.

By doing this, PickPocket could calculate a weighted average PSSM for those unknown or rare MHC alleles based on pocket similarities of all calculated HLA PSSMs from the training set. Figure 2.7 indicates the more-detailed flow network for PickPocket. PickPocket firstly extracts the position-specific vectors from the PSSMs in association with pseudo-sequence to construct a pocket library and each pocket library entry is characterized by nine pairs, where each pair consists of a specificity vector and a list of pocket amino acids[53]. Each number corresponds to the following steps: (1) Build up PSSMs from ligands data; (2) Extract pseudo-sequences for the pockets; (3) Extract the position-specific vectors to construct a pocket library;

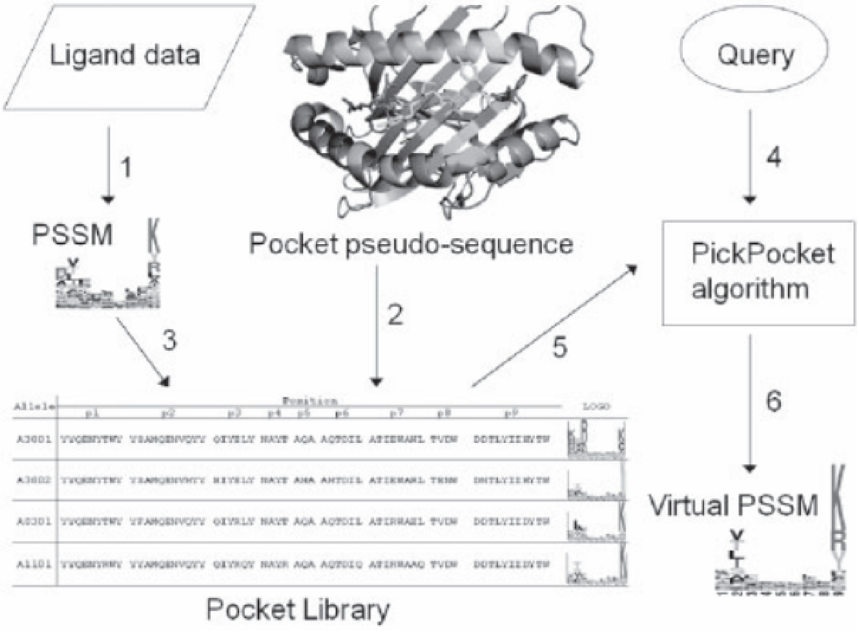


Figure 2.7: **PickPocket Flow Network. Reproduced from original paper [53]**

(4 and 5) Input a query MHC, the model retrieves the position-specific vectors and calculates mean vectors (6) The algorithm constructs a virtual PSSM for the target allele[53].

2.2.3 Consensus Models

The third one is called the consensus model. Due to some models may predict well for one given HLA molecule but perform poorly for others, a consensus model usually uses several different models and produces a prediction result based on all of them, which is usually defined as a simple average of those models[26]. The objective of consensus algorithms is to achieve an optimal combination of a series of prediction methods[17] for any given HLA molecule in an automatic manner[17]. There are some current popular consensus models, such as IEDB Consensus and netMHC-cons.

NetMHCcons

NetMHCcons is a popular consensus model for peptide-MHC binding prediction. In particular, two ANN-based MHC-peptide prediction models (NetMHC 3.4 and NetMHCpan) and one PSSM-based model (Pickpocket 1.1) have been utilized in this approach. NetMHCcons provides three options for peptide binding prediction to different MHC alleles. Each of the included models was first benchmarked separately and performance evaluated prior to the application of the consensus method [17]. In the end, the output result of NetMHCcons model is defined as following:

$$NetMHCcons = \begin{cases} NetMHCpan + Pickpocket & \text{if distance} >= 0.1 \\ NetMHC + NetMHCpan & \text{if distance} = 0 \\ NetMHCpan & \text{if } 0 < \text{distance} < 0.1 \end{cases} \tag{2.2}$$

where distance refers to the distance between the query HLA alleles and its nearest neighbour in the reference HLA allele list

2.3 Our Novel Design

Model Name	Category	Year	Special Design	Algorithm	Last Update
NetMHC4.0	Allele-specific	2014	Insertion and deletion	ANNs	October 2017
MHCflurry	Allele-specific	2018	Padding	ANNs	January 2019
NetMHCPan 4.0	Pan-specific	2017	HLA pseudo code features	ANNs	January 2018
PickPocket	Pan-specific	2009	Pocket Library	PSSM	January 2017
NetMHCcons	Consensus	2012	Sequence-based features	Consensus	January 2017

Table 2.1: Summary of related works for predicting MHC-peptide ligands.
The online link to access the above mentioned models are as following:
NetMHC 4.0: `http://www.cbs.dtu.dk/services/NetMHC/`;
MHCflurry: `https://github.com/openvax/mhcflurry/`;
NetMHCPan-4.0, `http://www.cbs.dtu.dk/services/NetMHCpan/`;
PickPocket 1.1: `http://www.cbs.dtu.dk/services/PickPocket/`;
NetMHCcons-1.1; `http://www.cbs.dtu.dk/services/NetMHCcons/`;

Table 2.1 is a summary of the related works as we discussed previously. Those model are still frequently used today and can be considered as the state of the art for peptide-MHC binding prediction problems. Currently, most of the prediction models are based on single machine learning algorithm, such as ANN-based, CNN-based and RNN-based model. But as explained in machine learning section, considering that CNNs are excellent in extracting motifs features from protein sequence and RNN performs well on sequential data, in this thesis, we propose a novel pan-specific model called MHCherryPan which makes full use of CNN and RNN architectures to predict the binding affinity between HLA class I and peptides by using large-scale public datasets. We compared our model with other current popular models, and the result proved our model performed very well on the HLA-peptide binding problem and achieved state-of-the-art results on a large number of HLA alleles with good generalization capability.

Chapter 3

Methodology

In this chapter, we propose a new model called MHCherryPan to solve HLA-peptide binding prediction problems, and then we would elucidate our approach thoroughly. Firstly, we demonstrate the problem that we want to solve, then we explain our model in detail, including encoding the input data, extracting peptide features, extracting HLA features, and predicting HLA-peptide binding affinity. Lastly, we would combine all the modules we discussed and introduce the pipeline for our model, which would give a big clear picture.

3.1 Problem statement

In this thesis, we build up a deep learning model to predict peptide binding to MHC/HLA through deep learning methods. In particular, we try to solve peptide-HLA binding prediction problems by using Recurrent Neural Network and VGG-based Convolutional Neural Network and then compare our model performance with other currently popular models.

3.2 Encoding

This section introduces the genetic codes for amino acids and how we transform them into the numeric representation by using the encoding matrix.

3.2.1 Genetic Codes for Amino Acids

The MHC/HLA and antigen peptides are composed of amino acids. There are around 500 amino acids known, but merely 20 amino acids appear in the genetic code[36]. Those 20 amino acids are used to represent the input data (peptides and HLA). Any one of these 20 amino acids has its name, but it also has the three-letter or one-letter representations which are commonly used in literature including this thesis:

alanine - ala - A
arginine - arg - R
asparagine - asn - N
aspartic acid - asp - D
cysteine - cys - C
glutamine - gln - Q
glutamic acid - glu - E
glycine - gly - G
histidine - his - H
isoleucine - ile - I
leucine - leu - L
lysine - lys - K
methionine - met - M
phenylalanine - phe - F
proline - pro - P
serine - ser - S
threonine - thr - T
tryptophan - trp - W
tyrosine - tyr - Y
valine - val - V
asparagine/aspartic acid - asx - B *
glutamine/glutamic acid - glx - Z *

*The last two amino acids (B and Z) are special due to sometimes it is difficult or impossible to differentiate between two amino acids. This uncertainty to identify between two amino acids (asparagine and aspartic acid, glutamine and glutamic acid) is shown with a special symbol (B or Z).

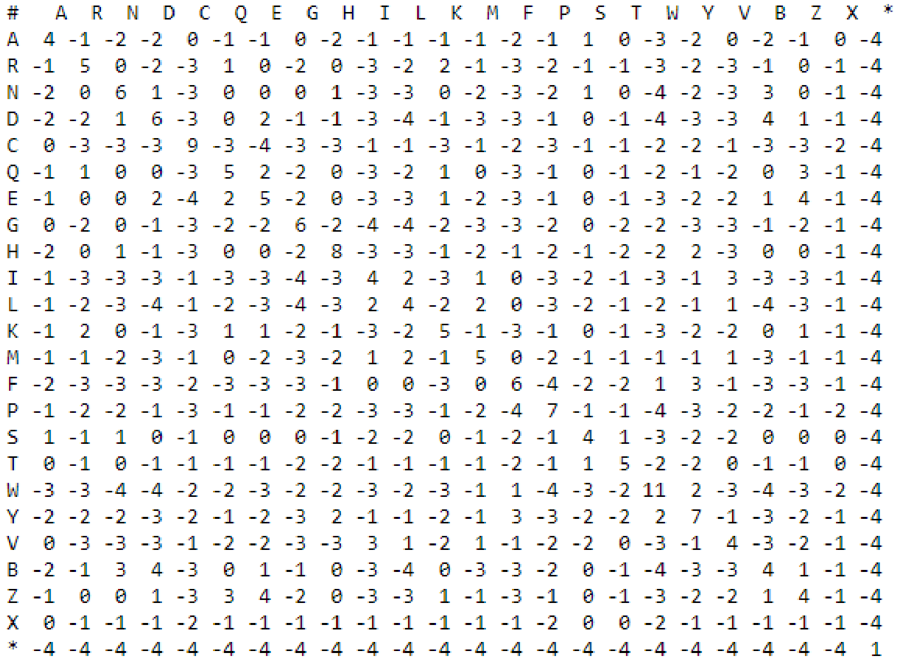
3.2.2 Encoding Method

When working with amino acids sequence data, the numeric representation of amino acids, which also is called encoding, will affect the model performance. Currently, there are two common encoding solutions for amino acids. The first is a simple method called a one-hot encoding model in which each amino acid is represented by a unit binary vector of length n, containing a single one and (n-1) zeros, e.g. [1,0,0, ., 0] for one amino acid and [0,1,0, ., 0] for another amino acid. The value of that channel is set to one if the corresponding amino acid appears, and the rest channels remain zero. This solution fairly treats all amino acids.

Another one is to use the BLOSUM matrix for encoding, representing each amino acid by its corresponding row in the BLOSUM matrix. The BLOSUM matrix is a substitution matrix to score alignments between evolutionarily divergent protein sequences [12]. Instead of fairly treating all amino acid, the BLOSUM matrix keeps the evolutionary information about which pairs of amino acids are easily interchangeable during evolution(i.e., which amino acid positions are highly variable and which are conserved)[16]. This will be very important in peptide-protein binding prediction.

In our MHCherryPan model, every input is a pair match of an HLA and a peptide. Taking the peptide and the HLA as the pair inputs, we utilize the Blosum62 matrix to encode according to locations of amino acids in sequences. Figure 3.1 shows the BLOSUM62 Matrix. Each peptide and HLA are represented to the network as a series of amino acids; each amino acid is represented as a 21-dimensional, smoothed, and blosum62- encoded vector. The peptide sequence is encoded into a tensor with the dimension as $1 \times 15 \times 21$, where the last dimension number represents the number of channels and every channel stands for one of 20 amino acids and one extra for padding and masking. 15 represents the peptide length. Although we choose 15 as the maximum length for peptide, our model can accept any-length peptides due to that we use the masking layer in RNN, which can handle variable-length inputs by skipping any input with mask value by copying the previous hidden state of the cell.

We tested our model on alignment-ready HLA sequences with different lengths and chose the length 180 as the fixed dimension. Then we encoded each aligned HLA sequence into a 2D tensor with dimension $1 \times 180 \times 21$. We acquired alignment-ready HLA - 2895 sequences of length 180 amino acids from the IMGT/HLA database[30]. Therefore, we encode each aligned HLA sequence into a 2D tensor with dimension $1 \times 180 \times 21$ in our model.



```
#  A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V  B  Z  X  *
A  4 -1 -2 -2  0 -1 -1  0 -2 -1 -1 -1 -1 -2 -1  1  0 -3 -2  0 -2 -1  0 -4
R -1  5  0 -2 -3  1  0 -2  0 -3 -2  2 -1 -3 -2 -1 -1 -3 -2 -3 -1  0 -1 -4
N -2  0  6  1 -3  0  0  0  1 -3 -3  0 -2 -3 -2  1  0 -4 -2 -3  3  0 -1 -4
D -2 -2  1  6 -3  0  2 -1 -1 -3 -4 -1 -3 -3 -1  0 -1 -4 -3 -3  4  1 -1 -4
C  0 -3 -3 -3  9 -3 -4 -3 -3 -1 -1 -3 -1 -2 -3 -1 -1 -2 -2 -1 -3 -3 -2 -4
Q -1  1  0  0 -3  5  2 -2  0 -3 -2  1  0 -3 -1  0 -1 -2 -1 -2  0  3 -1 -4
E -1  0  0  2 -4  2  5 -2  0 -3 -3  1 -2 -3 -1  0 -1 -3 -2 -2  1  4 -1 -4
G  0 -2  0 -1 -3 -2 -2  6 -2 -4 -4 -2 -3 -3 -2  0 -2 -2 -3 -3 -1 -2 -1 -4
H -2  0  1 -1 -3  0  0 -2  8 -3 -3 -1 -2 -1 -2 -1 -2 -2  2 -3  0  0 -1 -4
I -1 -3 -3 -3 -1 -3 -3 -4 -3  4  2 -3  1  0 -3 -2 -1 -3 -1  3 -3 -3 -1 -4
L -1 -2 -3 -4 -1 -2 -3 -4 -3  2  4 -2  2  0 -3 -2 -1 -2 -1  1 -4 -3 -1 -4
K -1  2  0 -1 -3  1  1 -2 -1 -3 -2  5 -1 -3 -1  0 -1 -3 -2 -2  0  1 -1 -4
M -1 -1 -2 -3 -1  0 -2 -3 -2  1  2 -1  5  0 -2 -1 -1 -1 -1  1 -3 -1 -1 -4
F -2 -3 -3 -3 -2 -3 -3 -3 -1  0  0 -3  0  6 -4 -2 -2  1  3 -1 -3 -3 -1 -4
P -1 -2 -2 -1 -3 -1 -1 -2 -2 -3 -3 -1 -2 -4  7 -1 -1 -4 -3 -2 -2 -1 -2 -4
S  1 -1  1  0 -1  0  0  0 -1 -2 -2  0 -1 -2 -1  4  1 -3 -2 -2  0  0  0 -4
T  0 -1  0 -1 -1 -1 -1 -2 -2 -1 -1 -1 -1 -2 -1  1  5 -2 -2  0 -1 -1  0 -4
W -3 -3 -4 -4 -2 -2 -3 -2 -2 -3 -2 -3 -1  1 -4 -3 -2 11  2 -3 -4 -3 -2 -4
Y -2 -2 -2 -3 -2 -1 -2 -3  2 -1 -1 -2 -1  3 -3 -2 -2  2  7 -1 -3 -2 -1 -4
V  0 -3 -3 -3 -1 -2 -2 -3 -3  3  1 -2  1 -1 -2 -2  0 -3 -1  4 -3 -2 -1 -4
B -2 -1  3  4 -3  0  1 -1  0 -3 -4  0 -3 -3 -2  0 -1 -4 -3 -3  4  1 -1 -4
Z -1  0  0  1 -3  3  4 -2  0 -3 -3  1 -1 -3 -1  0 -1 -3 -2 -2  1  4 -1 -4
X  0 -1 -1 -1 -2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -2  0  0 -2 -1 -1 -1 -1 -1 -4
* -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4  1
```

Figure 3.1: **BLOSUM62 Matrix**: The BLOSUM62 matrix is used to encode peptide and HLA sequences.

3.3 Model

In this section, we explain our model in detail, including how to extract peptide features, how to extract HLA feature, and how to predict HLA-peptide binding based on extracted peptide features and HLA features.

3.3.1 Peptide Feature Extraction Module

Recurrent neural networks (RNN) are neural network models for sequential data, and they have time-delayed connections between the neurons of a hidden layer. RNN process one element of the sequence at a time. In biological sequence context, RNN process one residue after the other[16]. The information, therefore, flows both from input to output and along the sequence. In this way, memory is generated and the neural network gains the ability to store and integrate information from past inputs. Long short-term memory (LSTM) neural networks are a particular type of RNN in which the scalar-valued hidden neuron is replaced with the LSTM memory

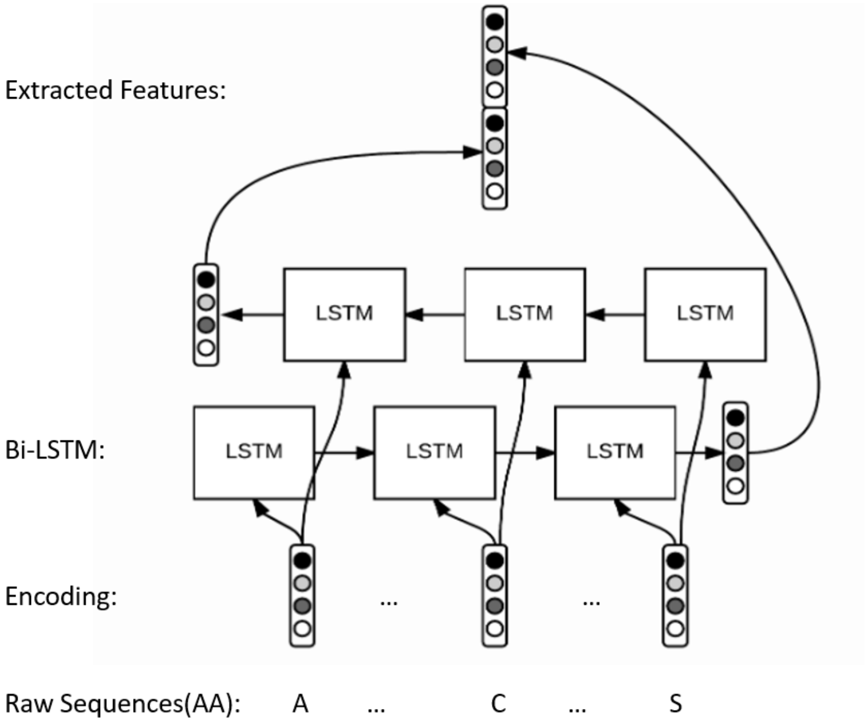


Figure 3.2: **Peptide Feature Extractor - RNN**

block[16]. The LSTM memory block is inspired by a computer memory cell and is easier for the network to store a given input over many time steps[16].

LSTM can take input sequences with varying length by using the masking layers, which makes LSTM highly flexible for dealing with unfixed length sequential data. In our model, LSTM was chosen as the extracting methods for peptide feature extraction in order to generalize the relationship between the amino acid representation in peptide sequence. Figure 3.2 shows the RNN peptide feature extractor. We used the Bi-direction LSTM layer with 128 number of units. Our peptide feature extraction section processes one amino acid from an entire sequence of peptide at a time, then predict whether it has the certain biological property after having seen the whole sequence. We utilized the bidirectional approach (biLSTM) where the network processed the input sequence forwards and backwards. In this way, each prediction was determined by not just what came before the current position but also what comes after.

Recurrent neural networks use the output of a certain element in the sequence as an extra

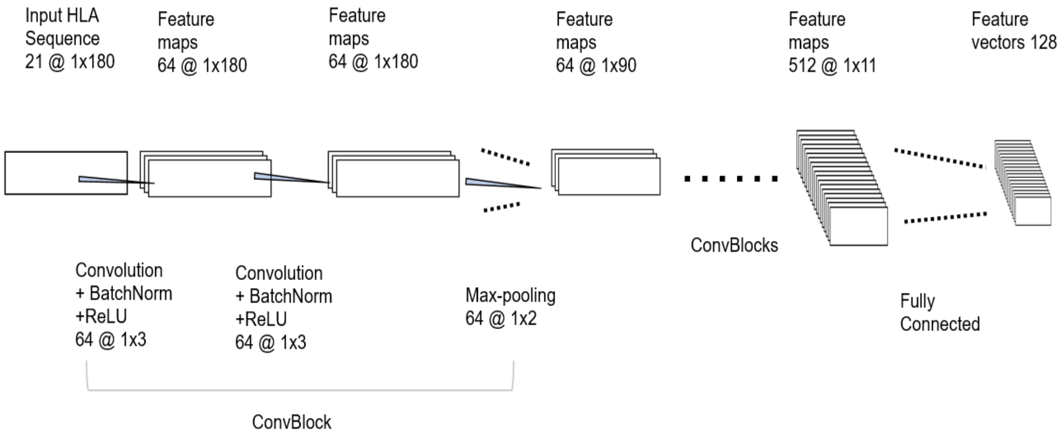


Figure 3.3: **HLA Feature Extractor - CNN**

input when handling the next element of that sequence. This allows the network to learn dependencies between elements of the same sequence. This is exactly the type of situation we are in. We presume that sequential features, such as the order or (relative) position of certain amino acids in the proteins sequence will influence the reaction between these proteins. This is the type of dependencies we want to discover. For this reason, choosing a recurrent neural network seems to be the more promising strategy for our problem.

3.3.2 HLA Feature Extraction Module

CNNs consist of several convolutional layers with many convolutional filters and maxpooling layers. Maxpooling layers enable the network to become invariant to small local deformations in the input[16]. In CNN, information flows only from the input to the output through layer by layer. They are not fully connected, but use a filter (a set of weights) over the input and feed the information into a different neuron in the next layer. The filter will thereby identify features in input irrespectively of where they appear[16]. We utilized a convolutional filter to detect a motif in an amino acid sequence.

Inspired by DeepSeqPan, we defined a similar Convolutional Block to extract high-level features from raw sequence data. As Figure 3.3 indicates, our convolutional block (ConvBlock) consists of two blocks with convolutional, batch normalization (a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch), max pooling and LeakyReLu layers (an activation function to transfer the negative number into a smaller number). Unlike DeepSeqPan that has many parameters needed to train, We simplified the structure and optimized it to extract the protein sequences data with 180 aa lengths. In our model, we applied a VGG-liked network configuration to extract the high-order features. VGG is a straight forward CNN architecture and the main idea is to stack the convolutional layers with increasing filter sizes. Information stored inside HLA proteins can be learned by the neural network automatically with its end-to-end training framework.

3.3.3 Affinity Prediction Module

As the training data is all labeled with the IC50 value (which is an experimental way to measure the affinity value between two peptides, see section 4.4.4 for more details), we devised our output layer to produce the predicted IC50 value for the input MHC/HLA - Peptide pair. To predict whether the input MHC/HLA - Peptide pair binding or not, we used a thread-hold value to get the binary label. We set the threshold value into 500nm, which is also commonly used by most of currently popular models such as MHCFlurry, DeepSeqPan, NetMHCPan and so on. The reason why most of the current models are using IC50 value as the label instead of using the binary label (0 for not binding and 1 for binding) is due to the IC50 value could potentially bring more useful information into our model like the strength of the binding.

In order to pass the high-order features into out layers, we designed a feed-forward neural network to perform and analyze the features passed from HLA Feature Extractor and Peptide Feature Extractor. We first used a concatenate layer to merge features from peptide and HLA. After this, we performed several dense layers (the regular deeply connected neural network layer), dropout layers (layers which is used to prevent a model from overfitting), and activation layers(LeakyReLu and linear layers) to further fully connect all the features. Eventually, our model outputs one number to represent the binding affinity (IC50).

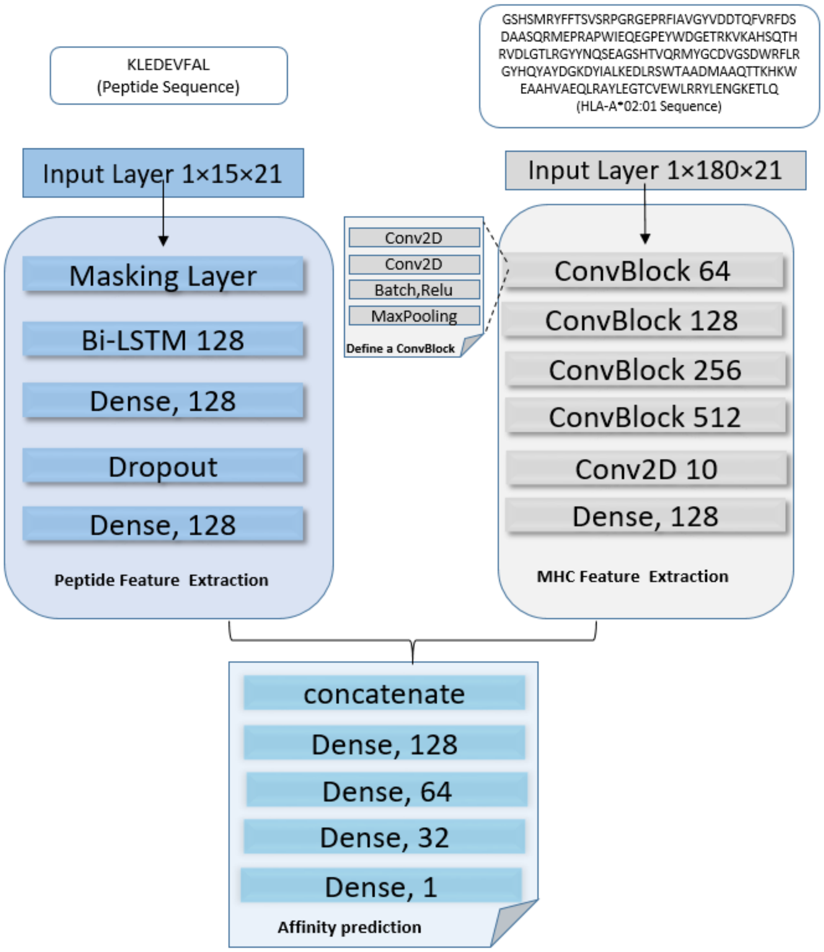


Figure 3.4: **MHCherryPan Network Architecture**: (I) Peptide Feature Extraction. (II) MHC Feature Extraction. (III) Affinity Prediction. (IV) Components of a convolutional Block

3.4 Pipeline

The MHCherryPan model for solving MHC-peptide binding prediction problems follows a straightforward pipeline, as shown in figure 3.4. As discussed, the model consisted three parts: peptide feature extraction, HLA Feature extraction, and affinity prediction. When the paired input data goes through the model, the HLA extractor module will extract abstract features from HLA sequences, and the peptide extractor module will extract features from peptide sequences. In the end, the affinity prediction module would concatenate the features from both modules and produce the prediction.

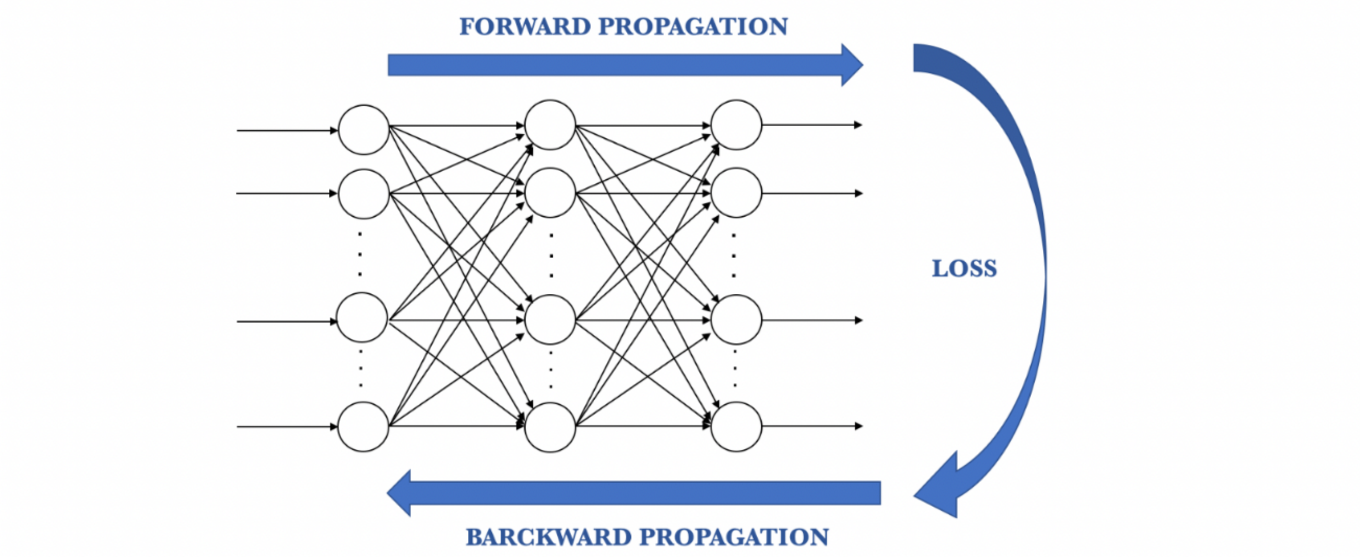


Figure 3.5: **Model Training**: Forward and Backward Propagation, replicated from original article[15].

3.5 Model Training

Training the neural network is to learn the values of parameters(weights and biases) for models. It is extremely significant for improving the model performance, and training is done by an iterative process of forward propagation (forward information flow), and backward propagation of the information by the layers of neurons. Figure 3.5shows the process of model training.

Forward propagation occurs when inputting the training data into the model, and these data go through the whole network for their labels or predictions to be computed. By doing this, the input data would be transformed through all the layers of network, and all its neurons have applied their transformation, the predicted label or result will be produced by the model or the final layer.

A loss function is used to calculate the error or the loss so we can compare and measure how close or distinct our prediction result was compared with the correct result (true label). The goal for training the model is to have the cost or loss function as small as possible, which means there is little or no divergence between predict label and expected value (true label). In order to achieve the goal, the weights of the neurons or the network will gradually be adjusted along the direction where the loss function is smaller and smaller. One of the common loss functions used for regression prediction is mean square error, which is also the loss function used to train this model. The function definition is shown as below (3.1).

$$L = \frac{1}{n}\sum_{i=1}^{n}(Y_{IC_{50}(i)} - \hat{Y}_{IC_{50}(i)})^2 \tag{3.1}$$

$Y_{IC_{50}(i)}$represents the true value and $\hat{Y}_{IC_{50}(i)}$ stands for the predict value.

Back propagation is a widely used algorithm to calculate the derivative (gradient) of the cost function with respect to the weight matrices. After the loss has been computed, this information is spread backward. The loss propagates to all layers that relate to the output result beginning from the output layer. Nevertheless, the hidden layers merely acquire a proportion of the total loss, according to how relative contribution that each layer has devoted to the former output. This process is repeated until the loss is minimum.

Chapter 4

Experiment setup

In this chapter, we discussed our experiment setup, including the datasets that we use, model training, model validation, performance evaluation methods and some experiment implementation details.

4.1 Dataset

This section would discuss the datasets we used. The data is the result of a combination of two datasets. We obtained the MHC sequential data from EBI, and we collected experimental binding data from IEDB. These two sets were then combined to create a single set to use for training and validation. We would discuss those databases we used and how we collect data in detail.

4.1.1 EBI

The EMBL Outstation-European Bioinformatics Institute (EBI) is a center for research and services in bioinformatics, which manages and makes available a wide range of the databases of biological data including nucleic acid, protein sequences, and macromolecular structures. For this thesis, we are particularly interested in the IPD-IMGT/HLA Database, which was used to collect details for the MHC molecules. The Immuno Polymorphism Database (IPD)- International ImMunoGeneTics (IMGT) Databases provides a database specialized for sequences of the human major histocompatibility complex (MHC). This database utilizes the official sequences names, which are from the WHO Nomenclature Committee For Factors of the HLA System[30]. The IPD-IMGT/HLA Database works as a part of the international ImMunoGeneTics project (IMGT).[30]. We obtained 2895 pre-aligned HLA sequences of length 180 amino acids from EBI.

4.1.2 IEDB

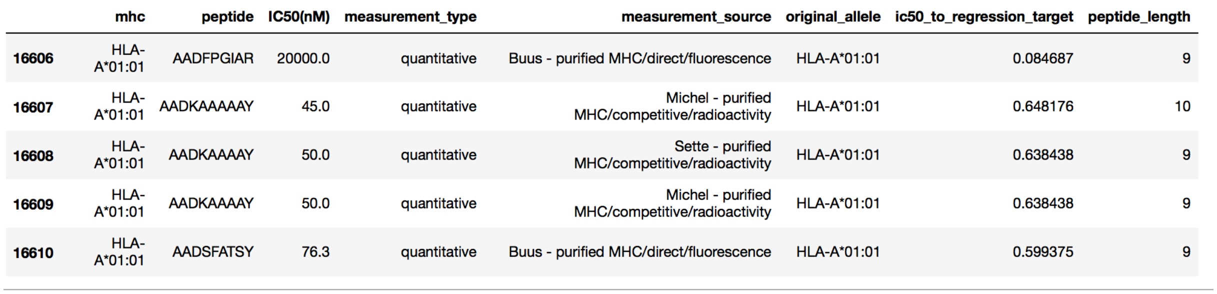
The Immune Epitope Database (IEDB)[9] provides a broad set of experimentally characterized peptides and peptide-MHC binding affinities data publicly for research use. Database entries are curated from published literature, and numerous affinities are conducted based on immunofluorescent assays. These affinities are represented as an IC_{50} value, the half-maximal inhibitory concentration in nano-molar (NM) units of peptide to MHC molecules. A total of approximately 1,100,000 examples from MHC ligand assays was available as of Aug 2019, spanning multiple mammalian and avian species.

The training dataset is downloaded from the IEDB database (`http://tools.iedb.org/main/datasets`). All training data is labeled by IC_{50} binding affinity values. The HLA sequences were obtained from the EBI database, and all the sequences are already in alignment in 180 aa length. We trained our model on peptides binding to HLA-A, HLA-B and HLA-C alleles with available HLA sequences. The training dataset contains 202091 peptide-HLA binding peptides covering 42 HLA-A alleles (132255 samples), 49 HLA-B alleles (66704 samples), and 10 HLA-C alleles (3132 samples). All peptides in the training set consist of 8-15 amino acid residues.

4.2 Data pre-processing and pre-analysis

In this section, we would further look into the peptide-hla data since the data is always the fundamental key for building up the model. Also, the more we comprehend our data, the better we could do to build up our model. We would introduce the pre-processing methods, data pre-analysis, and data visualization for our peptide-hla data.

4.2.1 Peptide-HLA data pre-processing

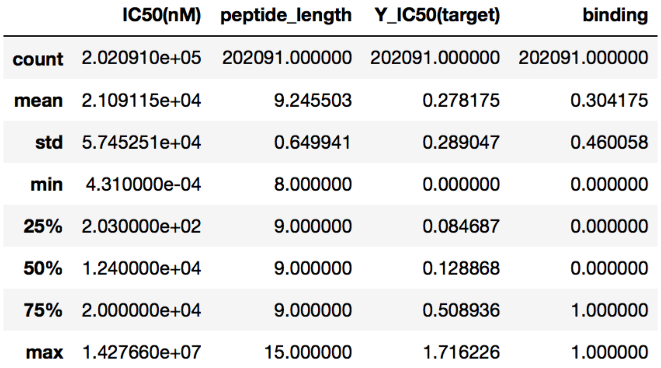


	mhc	peptide	IC50(nM)	measurement_type	measurement_source	original_allele	ic50_to_regression_target	peptide_length
16606	HLA-A*01:01	AADFPGIAR	20000.0	quantitative	Buus - purified MHC/direct/fluorescence	HLA-A*01:01	0.084687	9
16607	HLA-A*01:01	AADKAAAAAY	45.0	quantitative	Michel - purified MHC/competitive/radioactivity	HLA-A*01:01	0.648176	10
16608	HLA-A*01:01	AADKAAAAY	50.0	quantitative	Sette - purified MHC/competitive/radioactivity	HLA-A*01:01	0.638438	9
16609	HLA-A*01:01	AADKAAAAY	50.0	quantitative	Michel - purified MHC/competitive/radioactivity	HLA-A*01:01	0.638438	9
16610	HLA-A*01:01	AADSFATSY	76.3	quantitative	Buus - purified MHC/direct/fluorescence	HLA-A*01:01	0.599375	9

Figure 4.1: **MHC Training Data**

Data preprocessing is a data processing method to transform experimental data into a clean format, which could help to prevent some data issues such as incomplete data, data with lacking specific attributes, and data with errors[25]. The experimental peptide-HLA data we collected from IEDB contained many features(in total 98 fields), and many of them are irrelevant for model training. Therefore, the first step was to remove those irrelevant features from the dataset. When filtering the dataset, we kept only three fields (out of 98): mhc - the MHC name (e.g., HLA-A*01:01) using nomenclature for factors of the MHC System(`http://hla.alleles.org/nomenclature/naming.html`), peptide - the peptide amino acid sequences information for hla-peptide data, IC50(nM) - the binding affinity measurement by IC50 (The half maximal inhibitory concentration). Figure 4.1 illustrates an example of MHC training data. We also remove any entry with invalid data like sequences with uncertainty in the amino acids (see section 3.2.1).

Besides, We remove the sequences from the dataset that is rare enough. As mentioned, the peptide-HLA data varies by length of the peptide. The length of peptides for the majority of the data is around length 8 to length 15 aa (Please see figure 4.4 for more specific statistics). Therefore, we filtered out those data where the length is smaller than eight or greater than 15 due to that the Peptide-HLA data with those lengths don't have enough data. Figure 4.2 indicates the statistics summary (including total counts, means, Standard deviation, minimum, 25%, 50%, 75%, maximum) for our training dataset. Before we encode our data as the input to train our model, we also implemented data analysis and visualization to learn more about the data.



	IC50(nM)	peptide_length	Y_IC50(target)	binding
count	2.020910e+05	202091.000000	202091.000000	202091.000000
mean	2.109115e+04	9.245503	0.278175	0.304175
std	5.745251e+04	0.649941	0.289047	0.460058
min	4.310000e-04	8.000000	0.000000	0.000000
25%	2.030000e+02	9.000000	0.084687	0.000000
50%	1.240000e+04	9.000000	0.128868	0.000000
75%	2.000000e+04	9.000000	0.508936	1.000000
max	1.427660e+07	15.000000	1.716226	1.000000

Figure 4.2: **The statistics summary for training dataset**
The Y_IC50 is the constructed feature from IC50 using the log function. The details can be found in subsection 4.4.2.

4.2.2 HLA Sequences

We downloaded all the training data from IPD - IMGT datasets[30]. But since the goal is to build a pan-specific model, the amino acids sequences information for HLA are also necessary. HLA sequences have different lengths. Alignments for those sequences are necessary due to most of machine learning models require the fixed-length input. Therefore, the pre-aligned HLA sequences were collected from the EBI database and the length is 180 aa[33].

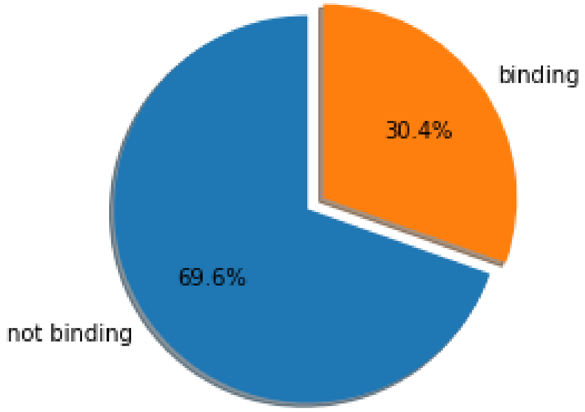


Figure 4.3: **Pie chart of binding or not for MHC Training Data.**[38]: The blue part is for ”not binding” data and the orange part is for ”binding” data.

4.2.3 Data pre-analysis and visualization

In order to have a better understanding of our data, we did a comprehensive analysis and visualization for our training data from IEDB.

We firstly group our data by ”binding” or ”not binding.” As Figure 4.3 indicates, there are 30.4% training data that are labeled ”binding” and 69.6% labeled ”not binding.” In reality, the ratio of positive vs negative was in the range of 1.3% vs 98.7% ([4]). This difference could be illustrated by the selection process used to select candidates on which the experiments are performed that contribute to those online datasets [4]. It is reasonable that lots of experiments are done with those samples that are more likely to yield a positive result [4]. Therefore, we get imbalanced data. In order to solve this, we used the Area under the ROC curve (AUC) instead of accuracy to measure our model performance. This is due to that accuracy only measures the number of correctly predicted samples over the total number of samples, and it could be misleading for unbalanced data since it will be more sensitive to the class of more data. In this case, the AUC can be used as a better model evaluation method since it could also consider the precision and recall. The AUC would be discussed in more detail in the section 4.4.2.

Regarding the mhc/hla peptide binding mechanisms, there are multiple features that play a

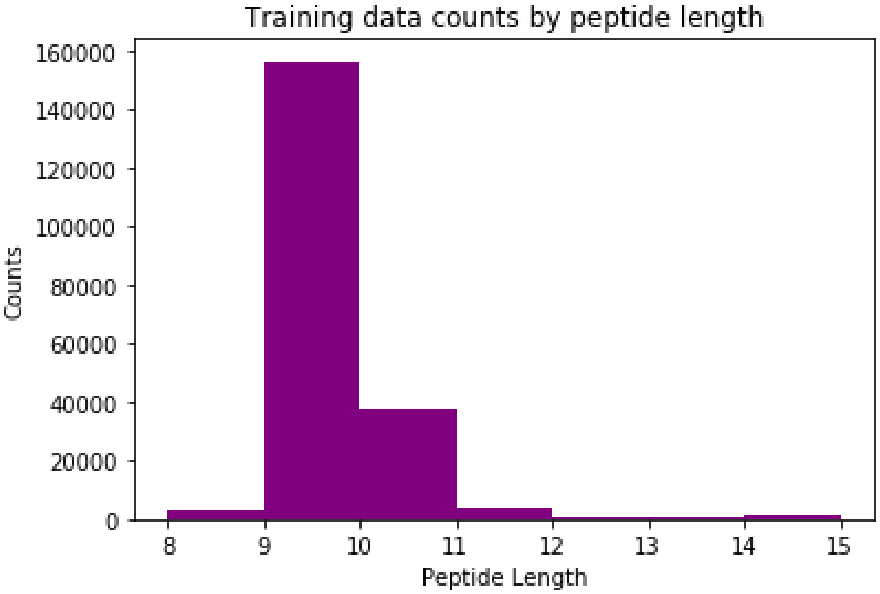


Figure 4.4: **Training data distribution by peptide length.**

role in it. Among them, two of the significant features which would have a big influence on model performance are hla alleles and peptide sequential data. Therefore, we grouped our data distribution by hla alleles and peptide sequences to have a better understanding of our data.

As we mentioned previously, the length of peptide is an important feature regarding the HLA peptide binding mechanism. Therefore, we grouped our data by the peptide length, and figure 4.4 is an overview of the hla-peptide data distribution by peptide length. As shown in figure 4.4, the numbers of training samples varies greatly. As we can see from the figure 4.4, most data concentrated in length 9, which is far more than any other length. The second most hla-peptide data is in length 10. As we can see from the figure, the majority of data stays in length 9 and length 10, which we would perform a more detailed analysis in the next subsection. The rest length data (length 8, 11, 12, 13, 14, 15) has less data compared with length 9 and length 10. Therefore building up a reasonable model which considering the data distribution in length would be significant. This is also one of the reasons why we choose the RNN. Due to the majority of data is in length 9 and length 10, we also analyze the frequency of amino acids in peptide.

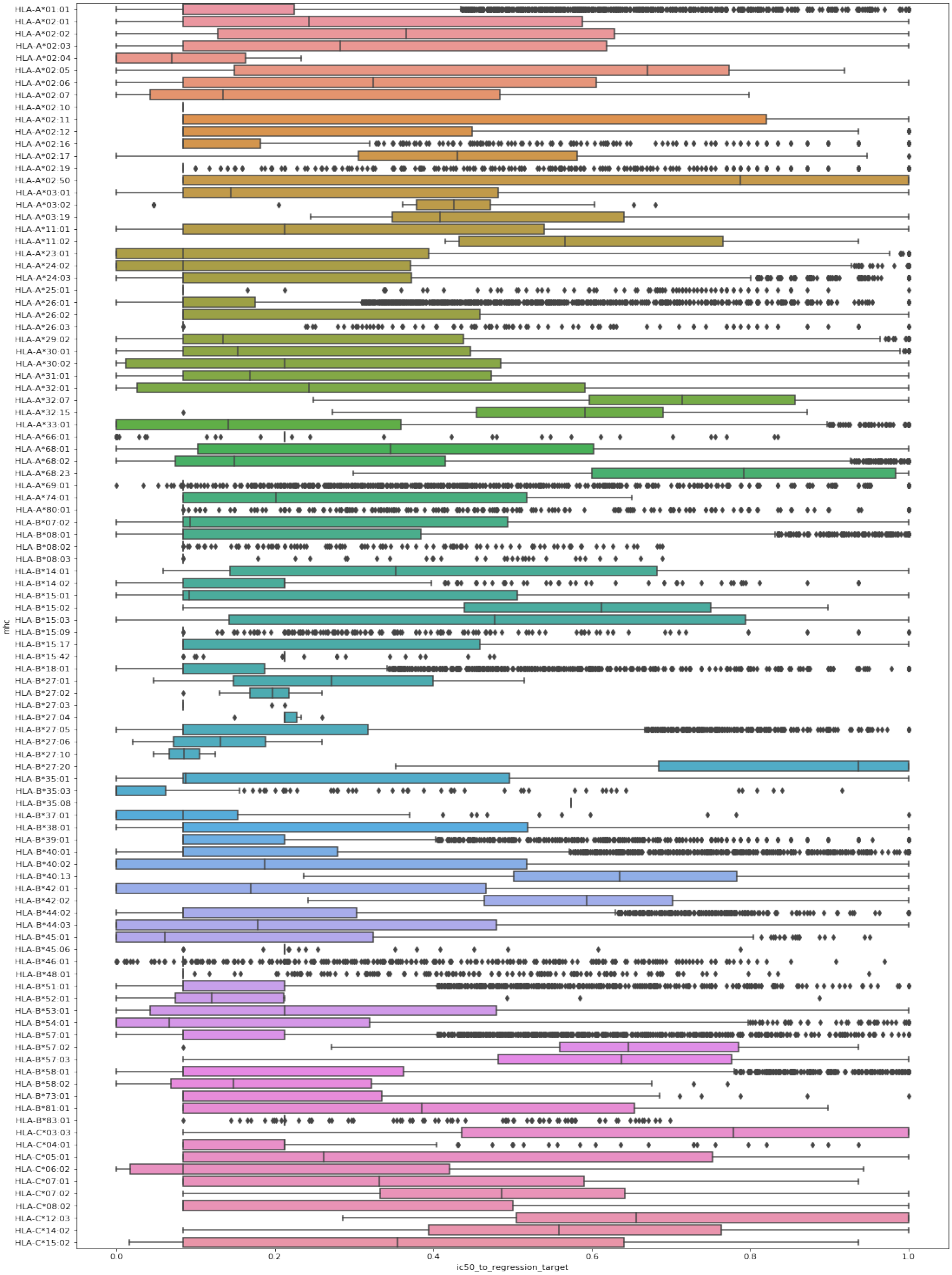


Figure 4.5: **Binding Affinity Distribution by Alleles**

Figure 4.5 indicated the affinity value(transformed IC50) distributions by different hla alleles. The x-axis is the target value and it represents the binding affinity between peptide and hla which will be inputted into our model(the transformed function is in 4.4.2 Metrics and Mathematical Function subsection) for training. We visualize the result through box plot. Using the box plot, we can compare the range and distribution of the affinity value for different hla alleles. Each box in alleles shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution. An outlier is an observation that is numerically distant from the rest of the data[42]. When reviewing a box plot, an outlier is defined as a data point that is located outside the fences(”whiskers”) of the box plot. As the figure shows, alleles have different binding affinity distribution. The neighboring alleles (the similar alleles) in the table tend to have the more similar distributions of binding affinities distribution.

4.2.4 The Amino Acid Enrichment for peptide-hla binding data

We also filtered out the non-binding data and analyzed HLA-binding binding data through statistics-based visualization using bar chart and iceLogo, examining amino acid abundance at each position of the 9-aa and 10-aa peptides.

Figure 4.6 and Figure 4.7 show the frequencies of each position of 9-aa peptides for HLA-peptide ligands. As we can see from the figure, for all the HLA allele types in this data set, the majority anchor positions were at the position 2, position 3, and the C-terminal position. In addition, we also found that the particular amino acid frequency at the C-terminal position correlated strongly among all the positions. Those results are in accordance with the similar statistic analysis we could find in Faridi’s paper in 2018 [7].

4.2.5 Encoding

After filtering, analyzing, and visualizing the data, the next step is to transfer the raw data, which is the amino acid sequence into the encoded data that the machine learning model could

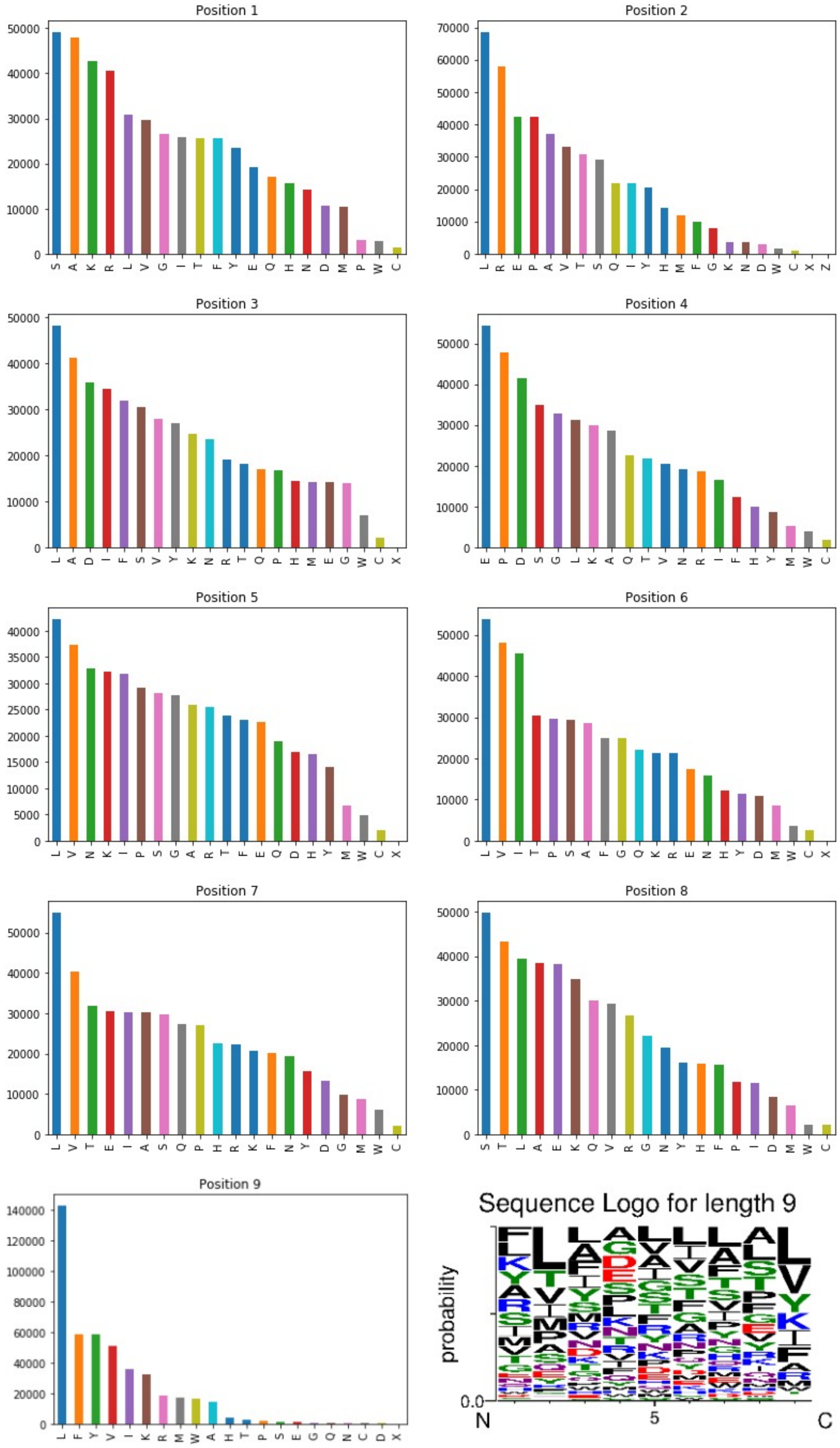


Figure 4.6: **Motif analysis for 9-mer peptides**[38]: showing the percentage of enriched amino acid at each position for hla-peptide binding data.

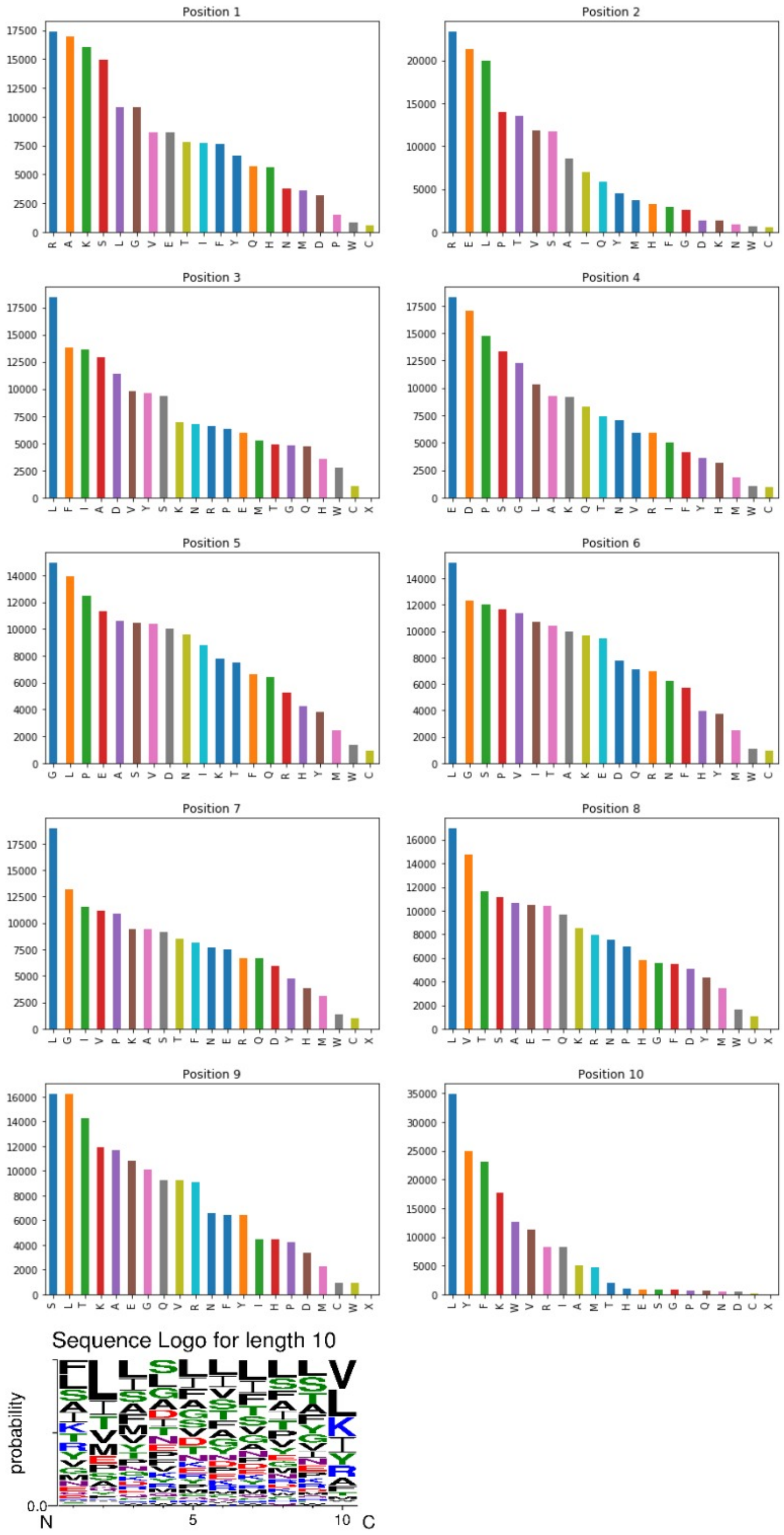


Figure 4.7: **Motif analysis for 10-mer peptides** showing the percentage of enriched amino acid at each position for hla-peptide binding data.

take as the input. As mentioned in section 2, we tested different encoding matrices and finally decided to use the BLOSUM62 Matrix to encode the pair of peptide-HLA sequences into the those NumPy arrays (A numpy array is the standard representation for numerical data [37]). The comparison result of different encoding metrics would be illustrated in the result section. Regarding the y label (IC50 values), due to our goal is to make a regression model to predict the binding affinity but the IC50 values in our datasets varies too much (from 0 to 80000NM), we use the equation as mentioned in subsection 4.4.2 to transfer our data into y value in the range of 0-1, which could help to input our data into the model.

4.3 Model Learning

In this section, we give some more details regarding the model learning. We introduce training the model, implementing the training experiment, and preventing overfitting.

4.3.1 Model training and preventing overfitting

We used training datasets and validation datasets to train our model. Firstly, we split all our training data randomly into training sets and validation sets by following a 4:1 ratio. Training datasets are used to fit or train the parameters of our model. The validation dataset would be used to give an unbiased evaluation for our trained model. By using validation sets, we could choose the best performance model from all the training models (with different trained parameters). The table 5.1 indicated the training AUC and validation AUC for our model.

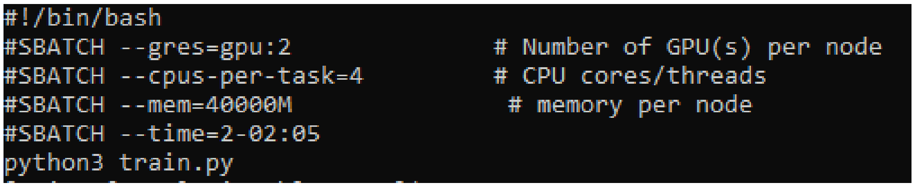
The MHCherryPan model was implemented with the Keras. Stochastic gradient descent (SGD) is utilized as the optimizer. We trained our model with learning rate decay and used an early stop approach to prevent overfitting. The initial learning rate (a tuning parameter in an optimization algorithm that determines the step size at each iteration while moving toward a minimum of a loss function [46]) is 0.005, and the momentum factor is 0.6. It is scheduled to decrease the learning rate if AUC has not increased within 20 epochs. We set the minimum learning rate as 0.00002. The training process stops if the loss on the validation set has not decreased within 50 epochs.

In addition to utilizing the early-stop strategy, we also applied the following methods to prevent overfitting. We designed several Drop Out layers in our model to reduce overfitting by introducing discrete noise during training. We also used multiple Batch normalization to re-parameterize the hidden unit activation in order to increase convergence speed and make the output stochastic, creating a regularizing effect[16].

4.3.2 Implementation

The entire programming for the model is done in Python 3.7. Pandas library package (version 0.24.1) is used for data preprocessing, and Keras library package (version 2.2.4) is utilized for building the model.

The model training is done through Sharcnet server. Regarding the resources, as Figure 4.8 shows, 2 GPU and 4 CPU cores are used. The memory usage is allocated for 40GB. The details regarding the used package and sharcnet server can be found in section 4.5.



```
#!/bin/bash
#SBATCH --gres=gpu:2               # Number of GPU(s) per node
#SBATCH --cpus-per-task=4          # CPU cores/threads
#SBATCH --mem=40000M                # memory per node
#SBATCH --time=2-02:05
python3 train.py
```

Figure 4.8: **Job script for Model training**

4.4 Performance Evaluation

In this section, we introduce the test benchmark dataset and how to evaluate the performance of our model by using the test dataset.

4.4.1 Test Benchmark Dataset

In order to evaluate our model performance, we use a public test benchmark dataset, which is provided by IEDB to test several popular hla-peptide binding algorithms. The test benchmark dataset and training dataset are distinct with each other. The test benchmark dataset is downloaded from IEDB's weekly benchmark dataset, and the date is ranged from 2014-03-21 to 2019-05-26 (`http://tools.iedb.org/auto_bench/mhci/weekly`). In case that duplicate peptides may appear in both the training and testing dataset downloaded from IEDB, we eliminated all duplicate peptides from the benchmark testing dataset.

4.4.2 Metrics and Mathematical Function

In this thesis, both Area under the curve (AUC) and Spearman's rank correlation coefficient (SRCC) are used as the evaluation metrics to compare the performance of our model with the public benchmark results of other current widely-used models from IEDB [32]. Both AUC and SRCC are calculated using the Scikit learn package for Python3.

Area under the receiver operating characteristic curve (AUC)

AUC is the area under the resulting curve of the positive rate and false positive rate at each possible score threshold.It is in a sense a broader metric, testing the quality of the internal value that the classifier generates and then compares to a threshold. As Figure 4.9 indicates, the false positive rate and the true positive rate(precision) are calculated for each possible score threshold, and the final metric is the area under the resulting curve.[3]

$$\text{True positive rate (or sensitivity or recall)} = \frac{\text{True positive}}{\text{true positive + false negative}} \tag{4.1}$$

$$\text{True negative rate (or Specificity)} \ = \frac{\text{True Negatives}}{\text{True Negatives + False Positives}} \tag{4.2}$$

$$\text{False positive rate} = \frac{\text{False Positive}}{\text{False Positive+True Negative}} \tag{4.3}$$

$$\text{Precision or positive predictive value} = \frac{\text{True Positive}}{\text{True Positive+False Positive}} \tag{4.4}$$

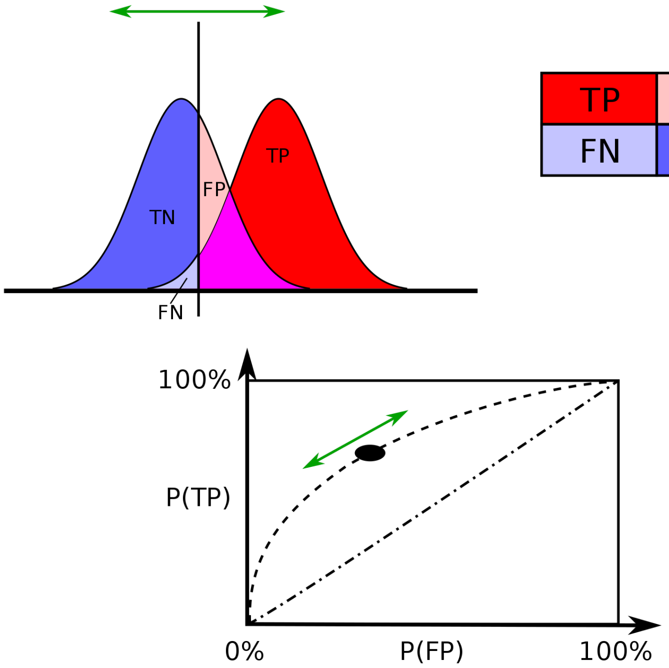


Figure 4.9: **AUC Curve.**: Taken from Receiver operating characteristic - Wiki [40]

A true positive (TP) defines as an outcome in which the model correctly predicts the positive class. Also, a true negative is an outcome in which the model correctly predicts the negative class.[45] A false positive (FP) defines as an outcome in which the model wrongly predicts the positive class. A false negative is an outcome in which the model wrongly predicts the negative class.[45] The true positive rate, true negative rate, and false positive rate are calculated as formula 4.1, 4.2, and 4.3. Sensitivity (which is also called the true positive rate or the recall) measures the proportion of actual positives that are correctly identified as such [48]. For instance, the percentage of infected people who are correctly identified as having the condition [48]. Specificity (which is called the true negative rate) measures the proportion of actual negatives that are correctly identified as such. For example, the percentage of uninfected people who are correctly identified as not having the condition [48].Positive rate is the sum of the true positive rate and the false positive rate. precision is the fraction of relevant instances among the retrieved instances, while recall (or sensitivity) is the fraction of the total amount of relevant instances that were actually retrieved. [47].

The reason why we prefer AUC is due to two points: first, AUC and Spearman's rank correlation coefficient are the standard measurements which are used on the IEDB benchmark website. It would be appropriate to use the same measurements as the benchmark website used. Second, AUC is more sensitive to imbalanced class. As explained, AUC considers true positive rate and false positive rate in the measurement, which in that sense may be a better summary of the performance of a classifier since it incorporates different aspects of the performance into a single number. Regarding accuracy, it may not be suitable for the imbalanced class. Accuracy would be favor for the majority class, which is not what we want. For example, if we have an imbalanced dataset where 95 are positive and 10 are negative, in this case, the classifiers could always predict ”positive” to achieve the high accuracy. But this is not what we want. We want a classifier to distinguish two classes instead of always guessing the ”majority” result. On the other hand, AUC is better since it consider true positive rate and false positive rate. In addition, some other measurements may also be acceptable like Precision-Recall curves. But since we want to compare our results with the results reported on benchmark website, it may be more appropriate to use the same measurement (AUC) as website used.

Spearman's rank correlation coefficient (SRCC)

The Spearman correlation coefficient is a method to summarise the direction and strength of a relationship between two variables.[50] The Spearman correlation coefficient is to measure the monotonic relationship between distributed variables. It represent the rank correlation between target and predicted values for quantitative measurements. The pair we used to measure here is the target value and predicted value. It defines as below:

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \tag{4.5}$$

d is the pairwise distances of the ranks of the variables and n is the number of samples.

There is an another similar method is called Pearson coefficients. It is also used to measure the relationship between two variables. But the difference is that the Pearson coefficients measure the linear relationship between two continuous variables while the Spearman correlation evaluates the monotonic relationship between two continuous or ordinal variables[50]. The monotonic relationship means the variables tend to change together, but may not at a constant rate. Since what we measure is the target value and predict value, we believe the monotonic relationship may makes more sense instead of only linear relationship. Also, as mentioned previously, another big consideration is that SRCC is the measurement used by the benchmark website. Those are the reasons why we choose the SRCC instead of Pearson coefficients. However, this is just our preference. Some model like MHCflurry used Kendall rank correlation coefficient (a similar matrix like SRCC to measure the monotonic relationship), AUC and Pearson coefficients for model performance evaluation. From their published result, the Pearson coefficients have the similar results as other two measurements.

Binding Affinity(IC_{50})

The IC_{50} value in our HLA-peptides training dataset covered a large range from 0 to 80000nM. IC_{50} value is used as the an experimental measurement for binding affinity. The higher the IC_{50} value stands for, the lower the binding affinity. In order to fit all data smoothly into the model and prevent the gradient explosion problem from training the deep neural network, we transform the IC_{50} into the regression y value using the equation below. We define $MaxIC_{50}$ as 50000nM.

$$Y_{IC_{50}} = \begin{cases} 0 & \text{if } IC_{50} > MaxIC_{50} \\ 1 - \log_{MaxIC_{50}} IC_{50} & \text{if } IC_{50} \leq MaxIC_{50} \end{cases} \tag{4.6}$$

In order to obtain the predicted label (whether binding or not), we use a common threshold 500nM for IC_{50}, for which many other popular models used and previous studies[31] demonstrated 500 nM as an MHC affinity threshold for HLA class I. We use binary label 0 - not binding and 1 - binding.

$$label = \begin{cases} 0 & \text{if } IC_{50} > 500nM \\ 1 & \text{if } IC_{50} \leq 500nM \end{cases} \tag{4.7}$$

The overall loss L is the sum of the regression loss defined as the Mean Square Error.

$$L = \frac{1}{n} \sum_{i=1}^{n} (Y_{IC_{50}(i)} - \hat{Y}_{IC_{50}(i)})^2 \tag{4.8}$$

$Y_{IC_{50}(i)}$represents the true value and $\hat{Y}_{IC_{50}(i)}$ stands for the predict value.

4.5 Technologies and Tools

This section gives an overview of the tools used throughout the various parts of this thesis. All code was written in Python 3.7.

Keras (2.2.4) is an open-source, high-level neural networks API written in Python. It can be configured to use Tensorflow, CNTK or Theano as backend. We used the default backend which is Tensorflow. Keras greatly simplifies the code needed to design, train and evaluate Neural Networks, while retaining the performance of more complicated libraries. This makes it a great tool for research.

Tensorflow (1.14.1) was used as the underlying library for Keras. It is an open-source library for numerical computations such as neural networks. It was originally designed by Google to perform a deep neural network and machine learning research.

Pandas (0.24.1) was used to reading, parsing and saving pre-processed files. It is an open-source library providing multiple data structures and data analysis tools which are commonly used in machine learning.

NumPy (1.13.1) is a package for scientific computing in Python. It is part of the SciPy (0.19.1) ecosystem. Its N-dimensional array structure is used as the main data container for the various mathematical libraries used throughout the thesis.

SHARCNET was used as the server for training our model. It is initiated by universities in Ontario, Canada to accumulate funding together to purchase supercomputer systems, and it could be shared and used by university researchers to perform research, instead of individually purchasing smaller systems.[49]

Chapter 5

Result and discussion

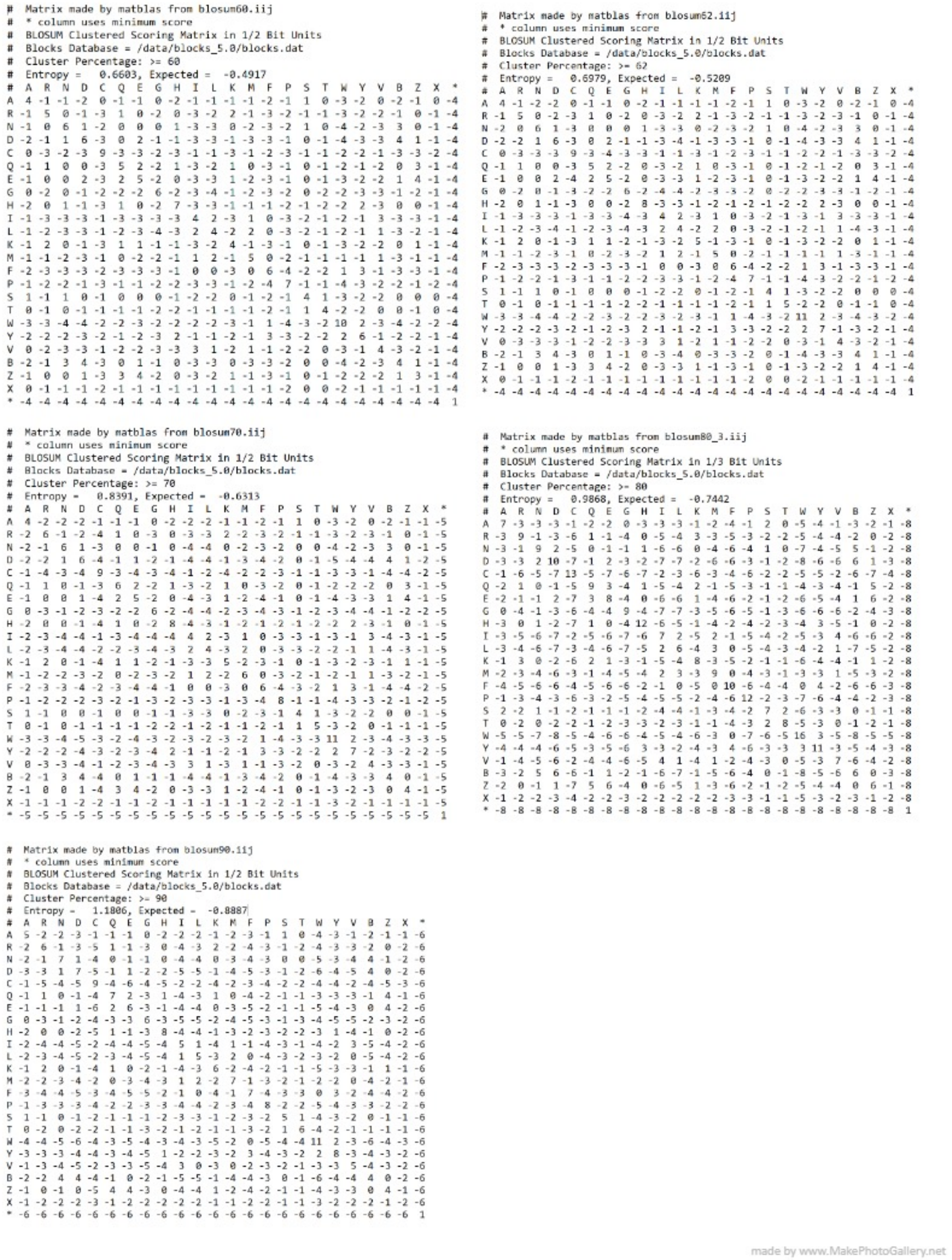
In this chapter, firstly, we discuss the experimental results, including the model performance on validation sets using different encoding matrix. Secondly, we compare the performance of our model with current popular models including as follows: pan-specific algorithms: NetMHCPan3.0, NetMHCPan4.0, and PickPocket, allele-specific algorithms: SMM, NetMHC3.4, NetMHC4.0, ARB, MHCflurry and AMMPMBEC, and consensus algorithms: IEDB Consensus and netMHCcons.

5.1 Experimental results on different encoding metrics

In this section, we discussed the model performance on the validation dataset by using different encoding metrics. Finally, selecting Blosum62 as the final encoding matrix.

5.1.1 BLOSUM Matrix

The figure 5.1 illustrates all the BLOSUM matrix which we tested for our model. It includes blosum 60, blosum 62, blosum 70, blosum 80, and blosum 90. BLOSUM stands for Block Substitution Matrices. It computes pairwise amino acid alignment counts and it counts amino acid replacement frequencies directly from columns in blocks. The number XX after BO-



```
#  Matrix made by matblas from blosum60.iij
#  * column uses minimum score
#  BLOSUM Clustered Scoring Matrix in 1/2 Bit Units
#  Blocks Database = /data/blocks_5.0/blocks.dat
#  Cluster Percentage: >= 60
#  Entropy =   0.6603, Expected =  -0.4917
#  A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V  B  Z  X  *
A  4 -1 -1 -2  0 -1 -1  0 -2 -1 -1 -1 -1 -2 -1  1  0 -3 -2  0 -2 -1  0 -4
R -1  5  0 -1 -3  1  0 -2  0 -3 -2  2 -1 -3 -2 -1 -1 -3 -2 -2 -1  0 -1 -4
N -1  0  6  1 -2  0  0  0  1 -3 -3  0 -2 -3 -2  1  0 -4 -2 -3  3  0 -1 -4
D -2 -1  1  6 -3  0  2 -1 -1 -3 -3 -1 -3 -3 -1  0 -1 -4 -3 -3  4  1 -1 -4
C  0 -3 -2 -3  9 -3 -3 -2 -3 -1 -1 -3 -1 -2 -3 -1 -1 -2 -2 -1 -3 -3 -2 -4
Q -1  1  0  0 -3  5  2 -2  1 -3 -2  1  0 -3 -1  0 -1 -2 -1 -2  0  3 -1 -4
E -1  0  0  2 -3  2  5 -2  0 -3 -3  1 -2 -3 -1  0 -1 -3 -2 -2  1  4 -1 -4
G  0 -2  0 -1 -2 -2 -2  6 -2 -3 -4 -1 -2 -3 -2  0 -2 -2 -3 -3 -1 -2 -1 -4
H -2  0  1 -1 -3  1  0 -2  7 -3 -3 -1 -1 -1 -2 -1 -2 -2  2 -3  0  0 -1 -4
I -1 -3 -3 -3 -1 -3 -3 -3 -3  4  2 -3  1  0 -3 -2 -1 -2 -1  3 -3 -3 -1 -4
L -1 -2 -3 -3 -1 -2 -3 -4 -3  2  4 -2  2  0 -3 -2 -1 -2 -1  1 -3 -2 -1 -4
K -1  2  0 -1 -3  1  1 -1 -1 -3 -2  4 -1 -3 -1  0 -1 -3 -2 -2  0  1 -1 -4
M -1 -1 -2 -3 -1  0 -2 -2 -1  1  2 -1  5  0 -2 -1 -1 -1 -1  1 -3 -1 -1 -4
F -2 -3 -3 -3 -2 -3 -3 -3 -1  0  0 -3  0  6 -4 -2 -2  1  3 -1 -3 -3 -1 -4
P -1 -2 -2 -1 -3 -1 -1 -2 -2 -3 -3 -1 -2 -4  7 -1 -1 -4 -3 -2 -2 -1 -2 -4
S  1 -1  1  0 -1  0  0  0 -1 -2 -2  0 -1 -2 -1  4  1 -3 -2 -2  0  0  0 -4
T  0 -1  0 -1 -1 -1 -1 -2 -2 -1 -1 -1 -1 -2 -1  1  4 -2 -2  0  0 -1  0 -4
W -3 -3 -4 -4 -2 -2 -3 -2 -2 -2 -2 -3 -1  1 -4 -3 -2 10  2 -3 -4 -2 -2 -4
Y -2 -2 -2 -3 -2 -1 -2 -3  2 -1 -1 -2 -1  3 -3 -2 -2  2  6 -1 -2 -2 -1 -4
V  0 -2 -3 -3 -1 -2 -2 -3 -3  3  1 -2  1 -1 -2 -2  0 -3 -1  4 -3 -2 -1 -4
B -2 -1  3  4 -3  0  1 -1  0 -3 -3  0 -3 -3 -2  0  0 -4 -2 -3  4  1 -1 -4
Z -1  0  0  1 -3  3  4 -2  0 -3 -2  1 -1 -3 -1  0 -1 -2 -2 -2  1  3 -1 -4
X  0 -1 -1 -1 -2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -2  0  0 -2 -1 -1 -1 -1 -1 -4
* -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4  1
```

```
#  Matrix made by matblas from blosum62.iij
#  * column uses minimum score
#  BLOSUM Clustered Scoring Matrix in 1/2 Bit Units
#  Blocks Database = /data/blocks_5.0/blocks.dat
#  Cluster Percentage: >= 62
#  Entropy =   0.6979, Expected =  -0.5209
#  A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V  B  Z  X  *
A  4 -1 -2 -2  0 -1 -1  0 -2 -1 -1 -1 -1 -2 -1  1  0 -3 -2  0 -2 -1  0 -4
R -1  5  0 -2 -3  1  0 -2  0 -3 -2  2 -1 -3 -2 -1 -1 -3 -2 -3 -1  0 -1 -4
N -2  0  6  1 -3  0  0  0  1 -3 -3  0 -2 -3 -2  1  0 -4 -2 -3  3  0 -1 -4
D -2 -2  1  6 -3  0  2 -1 -1 -3 -4 -1 -3 -3 -1  0 -1 -4 -3 -3  4  1 -1 -4
C  0 -3 -3 -3  9 -3 -4 -3 -3 -1 -1 -3 -1 -2 -3 -1 -1 -2 -2 -1 -3 -3 -2 -4
Q -1  1  0  0 -3  5  2 -2  0 -3 -2  1  0 -3 -1  0 -1 -2 -1 -2  0  3 -1 -4
E -1  0  0  2 -4  2  5 -2  0 -3 -3  1 -2 -3 -1  0 -1 -3 -2 -2  1  4 -1 -4
G  0 -2  0 -1 -3 -2 -2  6 -2 -4 -4 -2 -3 -3 -2  0 -2 -2 -3 -3 -1 -2 -1 -4
H -2  0  1 -1 -3  0  0 -2  8 -3 -3 -1 -2 -1 -2 -1 -2 -2  2 -3  0  0 -1 -4
I -1 -3 -3 -3 -1 -3 -3 -4 -3  4  2 -3  1  0 -3 -2 -1 -3 -1  3 -3 -3 -1 -4
L -1 -2 -3 -4 -1 -2 -3 -4 -3  2  4 -2  2  0 -3 -2 -1 -2 -1  1 -4 -3 -1 -4
K -1  2  0 -1 -3  1  1 -2 -1 -3 -2  5 -1 -3 -1  0 -1 -3 -2 -2  0  1 -1 -4
M -1 -1 -2 -3 -1  0 -2 -3 -2  1  2 -1  5  0 -2 -1 -1 -1 -1  1 -3 -1 -1 -4
F -2 -3 -3 -3 -2 -3 -3 -3 -1  0  0 -3  0  6 -4 -2 -2  1  3 -1 -3 -3 -1 -4
P -1 -2 -2 -1 -3 -1 -1 -2 -2 -3 -3 -1 -2 -4  7 -1 -1 -4 -3 -2 -2 -1 -2 -4
S  1 -1  1  0 -1  0  0  0 -1 -2 -2  0 -1 -2 -1  4  1 -3 -2 -2  0  0  0 -4
T  0 -1  0 -1 -1 -1 -1 -2 -2 -1 -1 -1 -1 -2 -1  1  5 -2 -2  0 -1 -1  0 -4
W -3 -3 -4 -4 -2 -2 -3 -2 -2 -3 -2 -3 -1  1 -4 -3 -2 11  2 -3 -4 -3 -2 -4
Y -2 -2 -2 -3 -2 -1 -2 -3  2 -1 -1 -2 -1  3 -3 -2 -2  2  7 -1 -3 -2 -1 -4
V  0 -3 -3 -3 -1 -2 -2 -3 -3  3  1 -2  1 -1 -2 -2  0 -3 -1  4 -3 -2 -1 -4
B -2 -1  3  4 -3  0  1 -1  0 -3 -4  0 -3 -3 -2  0 -1 -4 -3 -3  4  1 -1 -4
Z -1  0  0  1 -3  3  4 -2  0 -3 -3  1 -1 -3 -1  0 -1 -3 -2 -2  1  4 -1 -4
X  0 -1 -1 -1 -2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -2  0  0 -2 -1 -1 -1 -1 -1 -4
* -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4  1
```

```
#  Matrix made by matblas from blosum70.iij
#  * column uses minimum score
#  BLOSUM Clustered Scoring Matrix in 1/2 Bit Units
#  Blocks Database = /data/blocks_5.0/blocks.dat
#  Cluster Percentage: >= 70
#  Entropy =   0.8391, Expected =  -0.6313
#  A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V  B  Z  X  *
A  4 -2 -2 -2 -1 -1 -1  0 -2 -2 -2 -1 -1 -2 -1  1  0 -3 -2  0 -2 -1 -1 -5
R -2  6 -1 -2 -4  1  0 -3  0 -3 -3  2 -2 -3 -2 -1 -1 -3 -2 -3 -1  0 -1 -5
N -2 -1  6  1 -3  0  0 -1  0 -4 -4  0 -2 -3 -2  0  0 -4 -2 -3  3  0 -1 -5
D -2 -2  1  6 -4 -1  1 -2 -1 -4 -4 -1 -3 -4 -2  0 -1 -5 -4 -4  4  1 -2 -5
C -1 -4 -3 -4  9 -3 -4 -3 -4 -1 -2 -4 -2 -2 -3 -1 -1 -3 -3 -1 -4 -4 -2 -5
Q -1  1  0 -1 -3  6  2 -2  1 -3 -2  1  0 -3 -2  0 -1 -2 -2 -2  0  3 -1 -5
E -1  0  0  1 -4  2  5 -2  0 -4 -3  1 -2 -4 -1  0 -1 -4 -3 -3  1  4 -1 -5
G  0 -3 -1 -2 -3 -2 -2  6 -2 -4 -4 -2 -3 -4 -3 -1 -2 -3 -4 -4 -1 -2 -2 -5
H -2  0  0 -1 -4  1  0 -2  8 -4 -3 -1 -2 -1 -2 -1 -2 -2  2 -3 -1  0 -1 -5
I -2 -3 -4 -4 -1 -3 -4 -4 -4  4  2 -3  1  0 -3 -3 -1 -3 -1  3 -4 -3 -1 -5
L -2 -3 -4 -4 -2 -2 -3 -4 -3  2  4 -3  2  0 -3 -3 -2 -2 -1  1 -4 -3 -1 -5
K -1  2  0 -1 -4  1  1 -2 -1 -3 -3  5 -2 -3 -1  0 -1 -3 -2 -3 -1  1 -1 -5
M -1 -2 -2 -3 -2  0 -2 -3 -2  1  2 -2  6  0 -3 -2 -1 -2 -1  1 -3 -2 -1 -5
F -2 -3 -3 -4 -2 -3 -4 -4 -1  0  0 -3  0  6 -4 -3 -2  1  3 -1 -4 -4 -2 -5
P -1 -2 -2 -2 -3 -2 -1 -3 -2 -3 -3 -1 -3 -4  8 -1 -1 -4 -3 -3 -2 -1 -2 -5
S  1 -1  0  0 -1  0  0 -1 -1 -3 -3  0 -2 -3 -1  4  1 -3 -2 -2  0  0 -1 -5
T  0 -1  0 -1 -1 -1 -1 -2 -2 -1 -2 -1 -1 -2 -1  1  5 -3 -2  0 -1 -1 -1 -5
W -3 -3 -4 -5 -3 -2 -4 -3 -2 -3 -2 -3 -2  1 -4 -3 -3 11  2 -3 -4 -3 -3 -5
Y -2 -2 -2 -4 -3 -2 -3 -4  2 -1 -1 -2 -1  3 -3 -2 -2  2  7 -2 -3 -2 -2 -5
V  0 -3 -3 -4 -1 -2 -3 -4 -3  3  1 -3  1 -1 -3 -2  0 -3 -2  4 -3 -3 -1 -5
B -2 -1  3  4 -4  0  1 -1 -1 -4 -4 -1 -3 -4 -2  0 -1 -4 -3 -3  4  0 -1 -5
Z -1  0  0  1 -4  3  4 -2  0 -3 -3  1 -2 -4 -1  0 -1 -3 -2 -3  0  4 -1 -5
X -1 -1 -1 -2 -2 -1 -1 -2 -1 -1 -1 -1 -1 -2 -2 -1 -1 -3 -2 -1 -1 -1 -1 -5
* -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5  1
```

```
#  Matrix made by matblas from blosum80_3.iij
#  * column uses minimum score
#  BLOSUM Clustered Scoring Matrix in 1/3 Bit Units
#  Blocks Database = /data/blocks_5.0/blocks.dat
#  Cluster Percentage: >= 80
#  Entropy =   0.9868, Expected =  -0.7442
#  A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V  B  Z  X  *
A  7 -3 -3 -3 -1 -2 -2  0 -3 -3 -3 -1 -2 -4 -1  2  0 -5 -4 -1 -3 -2 -1 -8
R -3  9 -1 -3 -6  1 -1 -4  0 -5 -4  3 -3 -5 -3 -2 -2 -5 -4 -4 -2  0 -2 -8
N -3 -1  9  2 -5  0 -1 -1  1 -6 -6  0 -4 -6 -4  1  0 -7 -4 -5  5 -1 -2 -8
D -3 -3  2 10 -7 -1  2 -3 -2 -7 -7 -2 -6 -6 -3 -1 -2 -8 -6 -6  6  1 -3 -8
C -1 -6 -5 -7 13 -5 -7 -6 -7 -2 -3 -6 -3 -4 -6 -2 -2 -5 -5 -2 -6 -7 -4 -8
Q -2  1  0 -1 -5  9  3 -4  1 -5 -4  2 -1 -5 -3 -1 -1 -4 -3 -4 -1  5 -2 -8
E -2 -1 -1  2 -7  3  8 -4  0 -6 -6  1 -4 -6 -2 -1 -2 -6 -5 -4  1  6 -2 -8
G  0 -4 -1 -3 -6 -4 -4  9 -4 -7 -7 -3 -5 -6 -5 -1 -3 -6 -6 -6 -2 -4 -3 -8
H -3  0  1 -2 -7  1  0 -4 12 -6 -5 -1 -4 -2 -4 -2 -3 -4  3 -5 -1  0 -2 -8
I -3 -5 -6 -7 -2 -5 -6 -7 -6  7  2 -5  2 -1 -5 -4 -2 -5 -3  4 -6 -6 -2 -8
L -3 -4 -6 -7 -3 -4 -6 -7 -5  2  6 -4  3  0 -5 -4 -3 -4 -2  1 -7 -5 -2 -8
K -1  3  0 -2 -6  2  1 -3 -1 -5 -4  8 -3 -5 -2 -1 -1 -6 -4 -4 -1  1 -2 -8
M -2 -3 -4 -6 -3 -1 -4 -5 -4  2  3 -3  9  0 -4 -3 -1 -3 -3  1 -5 -3 -2 -8
F -4 -5 -6 -6 -4 -5 -6 -6 -2 -1  0 -5  0 10 -6 -4 -4  0  4 -2 -6 -6 -3 -8
P -1 -3 -4 -3 -6 -3 -2 -5 -4 -5 -5 -2 -4 -6 12 -2 -3 -7 -6 -4 -4 -2 -3 -8
S  2 -2  1 -1 -2 -1 -1 -1 -2 -4 -4 -1 -3 -4 -2  7  2 -6 -3 -3  0 -1 -1 -8
T  0 -2  0 -2 -2 -1 -2 -3 -3 -2 -3 -1 -1 -4 -3  2  8 -5 -3  0 -1 -2 -1 -8
W -5 -5 -7 -8 -5 -4 -6 -6 -4 -5 -4 -6 -3  0 -7 -6 -5 16  3 -5 -8 -5 -5 -8
Y -4 -4 -4 -6 -5 -3 -5 -6  3 -3 -2 -4 -3  4 -6 -3 -3  3 11 -3 -5 -4 -3 -8
V -1 -4 -5 -6 -2 -4 -4 -6 -5  4  1 -4  1 -2 -4 -3  0 -5 -3  7 -6 -4 -2 -8
B -3 -2  5  6 -6 -1  1 -2 -1 -6 -7 -1 -5 -6 -4  0 -1 -8 -5 -6  6  0 -3 -8
Z -2  0 -1  1 -7  5  6 -4  0 -6 -5  1 -3 -6 -2 -1 -2 -5 -4 -4  0  6 -1 -8
X -1 -2 -2 -3 -4 -2 -2 -3 -2 -2 -2 -2 -2 -3 -3 -1 -1 -5 -3 -2 -3 -1 -2 -8
* -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8  1
```

```
#  Matrix made by matblas from blosum90.iij
#  * column uses minimum score
#  BLOSUM Clustered Scoring Matrix in 1/2 Bit Units
#  Blocks Database = /data/blocks_5.0/blocks.dat
#  Cluster Percentage: >= 90
#  Entropy =   1.1806, Expected =  -0.8887
#  A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V  B  Z  X  *
A  5 -2 -2 -3 -1 -1 -1  0 -2 -2 -2 -1 -2 -3 -1  1  0 -4 -3 -1 -2 -1 -1 -6
R -2  6 -1 -3 -5  1 -1 -3  0 -4 -3  2 -2 -4 -3 -1 -2 -4 -3 -3 -2  0 -2 -6
N -2 -1  7  1 -4  0 -1 -1  0 -4 -4  0 -3 -4 -3  0  0 -5 -3 -4  4 -1 -2 -6
D -3 -3  1  7 -5 -1  1 -2 -2 -5 -5 -1 -4 -5 -3 -1 -2 -6 -4 -5  4  0 -2 -6
C -1 -5 -4 -5  9 -4 -6 -4 -5 -2 -2 -4 -2 -3 -4 -2 -2 -4 -4 -2 -4 -5 -3 -6
Q -1  1  0 -1 -4  7  2 -3  1 -4 -3  1  0 -4 -2 -1 -1 -3 -3 -3 -1  4 -1 -6
E -1 -1 -1  1 -6  2  6 -3 -1 -4 -4  0 -3 -5 -2 -1 -1 -5 -4 -3  0  4 -2 -6
G  0 -3 -1 -2 -4 -3 -3  6 -3 -5 -5 -2 -4 -5 -3 -1 -3 -4 -5 -5 -2 -3 -2 -6
H -2  0  0 -2 -5  1 -1 -3  8 -4 -4 -1 -3 -2 -3 -2 -2 -3  1 -4 -1  0 -2 -6
I -2 -4 -4 -5 -2 -4 -4 -5 -4  5  1 -4  1 -1 -4 -3 -1 -4 -2  3 -5 -4 -2 -6
L -2 -3 -4 -5 -2 -3 -4 -5 -4  1  5 -3  2  0 -4 -3 -2 -3 -2  0 -5 -4 -2 -6
K -1  2  0 -1 -4  1  0 -2 -1 -4 -3  6 -2 -4 -2 -1 -1 -5 -3 -3 -1  1 -1 -6
M -2 -2 -3 -4 -2  0 -3 -4 -3  1  2 -2  7 -1 -3 -2 -1 -2 -2  0 -4 -2 -1 -6
F -3 -4 -4 -5 -3 -4 -5 -5 -2 -1  0 -4 -1  7 -4 -3 -3  0  3 -2 -4 -4 -2 -6
P -1 -3 -3 -3 -4 -2 -2 -3 -3 -4 -4 -2 -3 -4  8 -2 -2 -5 -4 -3 -3 -2 -2 -6
S  1 -1  0 -1 -2 -1 -1 -1 -2 -3 -3 -1 -2 -3 -2  5  1 -4 -3 -2  0 -1 -1 -6
T  0 -2  0 -2 -2 -1 -1 -3 -2 -1 -2 -1 -1 -3 -2  1  6 -4 -2 -1 -1 -1 -1 -6
W -4 -4 -5 -6 -4 -3 -5 -4 -3 -4 -3 -5 -2  0 -5 -4 -4 11  2 -3 -6 -4 -3 -6
Y -3 -3 -3 -4 -4 -3 -4 -5  1 -2 -2 -3 -2  3 -4 -3 -2  2  8 -3 -4 -3 -2 -6
V -1 -3 -4 -5 -2 -3 -3 -5 -4  3  0 -3  0 -2 -3 -2 -1 -3 -3  5 -4 -3 -2 -6
B -2 -2  4  4 -4 -1  0 -2 -1 -5 -5 -1 -4 -4 -3  0 -1 -6 -4 -4  4  0 -2 -6
Z -1  0 -1  0 -5  4  4 -3  0 -4 -4  1 -2 -4 -2 -1 -1 -4 -3 -3  0  4 -1 -6
X -1 -2 -2 -2 -3 -1 -2 -2 -2 -2 -2 -1 -1 -2 -2 -1 -1 -3 -2 -2 -2 -1 -2 -6
* -6 -6 -6 -6 -6 -6 -6 -6 -6 -6 -6 -6 -6 -6 -6 -6 -6 -6 -6 -6 -6 -6 -6  1
```

made by www.MakePhotoGallery.net

Figure 5.1: **Blosum Matrix:** from top to bottom: blosum 60, blosum 62, blosum 70, blosum 80, and blosum 90. The matrix is downloaded from ftp://ftp.ncbi.nlm.nih.gov/blast/matrices/

LUSM indicates the sequences that are XX% similar are clustered during the construction of the BLOSUM matrix. BLOSUM matrices with high numbers are designed for comparing closely related sequences, while those with low numbers are designed for comparing distantly related sequences[39].

5.1.2 Results on different blossom matrix

The table 5.1 indicates the model performance using different encoding matrix including blosum60, blosum62, blosum65, blosum70, and blosum90. Regarding my model, all blosum matrix performances similarly and blosum62 gives a slightly better model performance. Those blosum matrix were downloaded from NCBI website(`ftp://ftp.ncbi.nih.gov/blast/matrices/`)

Encoding Matrix	**AUC**	**SRCC**
Blosum60	0.833	0.734
Blosum62	0.843	0.742
blosum70	0.839	0.738
blosum80	0.842	0.748
blosum90	0.840	0.743

Table 5.1: Validation Experiment results for the MHCherryPan model trained on IEDB dataset.

From the table 5.1, it can be seen that for the blosum 60, blosum 70, blosum 80, and blosum 90, the highest AUC (blosum62) is 84.3% and the lowest (blosum60) is 83.3%. Also, the difference is very small (0.1%). The reasons why We finally choose the blosum 62 as the final encoding matrix is due to the following two points: firstly, it gives us the slighter better AUC performance comparing with other matrices. Second, BLOSUM-62 matrix is a more widely used matrix, and it is among the best for detecting the weakest protein similarities. It is also the default matrix in lots of protein alignments such as protein BLAST. Although we choose the Blosum62 as the default encoding matrix, programming is designed to be flexible, and it can easily switch to other encoding matrices as well.

5.2 Experimental results on Benchmark testing dataset

In order to compare our model performance with other popular HLA-binding prediction models, we evaluated our model on the public IEDB weekly benchmark datasets where a set of top algorithms have been evaluated with the published results. Those top algorithms include as follows: pan-specific algorithms: NetMHCPan3.0, NetMHCPan4.0, and PickPocket, allele-specific algorithms: SMM, NetMHC3.4, NetMHC4.0, ARB, MHCflurry and AMMPMBEC, and consensus algorithms (results are based on several different algorithms): IEDB Consensus and netMHCcons. We firstly trained our MHCherryPan model on the training datasets. Then we evaluated our trained MHCherryPan on all the available IEDB weekly benchmark datasets. AUC and SRCC are used as the metrics to compare the performance.

Table 5.2 summarized the performance of different models on a total of 77 IEDB benchmark testing datasets. For each IEDB dataset, we highlighted the highest AUC scores in red and the highest SRCC scores in green. Then we summed up the number of datasets where each algorithm achieved the highest scores and we put them at the bottom row of the table. It can be observed that our MHCherryPan acquired the highest AUC scores in 29 records out of a total of 77 testing datasets. In 48 datasets where our MHCherryPan didn't acquire the highest AUC scores, there are 14 records on which the AUC scores of MHCherryPan are very close to the highest AUC scores within a small margin around 0.2. Comparing with all other algorithms regarding AUC in the benchmark table, mhcflurry and our model ranked the first and second, obtaining the similar numbers of 30 and 29. In terms of SRCC, MHCherryPan obtained the highest scores on 17 records, which ranked the second. Among pan-specific models (NetMHC-Pan3.0, NetMHCPan4.0, and PickPocket), our model performed much better in both AUC and SRCC, and our numbers of highest AUC and SRCC are almost double the other pan-specific algorithms.

Table 5.2: Performance Evaluation on IEDB Benchmark Database

Measure					**Pan-specific**								**Allele-specific**												**Ensemble**			
					MHCherryPan		**NetMHCpan3**		**NetMHCpan4**		**PickPocket**		**SMM**		**NetMHC3.4**		**NetMHC4**		**ARB**		**SMMPMBEC**		**mhcflurry**		**IEDBCons**		**NetMHCcons**	
HLA	**IEDB Ref**	**Type**	**Len**	**Count**	**auc**	**srcc**	**auc**	**srcc**	**auc**	**srcc**	**auc**	**srcc**	**auc**	**srcc**	**auc**	**srcc**	**auc**	**srcc**	**auc**	**srcc**	**auc**	**srcc**	**auc**	**srcc**	**auc**	**srcc**	**auc**	**srcc**
HLA-B*57-01	1029061	ic50	9	26	0.87	0.64	0.84	0.6	0.82	0.57	0.88	0.67	0.85	0.62	0.8	0.62	0.74	0.56	0.45	0.16	0.79	0.56	0.8	0.66	0.81	0.58	0.8	0.57
HLA-B*57-01	1028554	ic50	9	53	0.96	0.7	0.87	0.64	0.88	0.6	0.85	0.44	0.77	0.33	0.94	0.52	0.96	0.53	0.63	0.12	0.77	0.29	0.97	0.62	0.77	0.33	0.92	0.56
HLA-C*05-01	1028230	ic50	9	172	0.98	0.85	0.98	0.89	0.97	0.88	0.95	0.84	1	0.91	1	0.94	0.99	0.92	-	-	0.99	0.9	0.91	0.8	1	0.91	1	0.92
HLA-C*07-02	1028232	ic50	9	140	0.92	0.78	0.96	0.81	0.97	0.83	0.91	0.76	0.96	0.92	0.97	0.89	0.95	0.85	-	-	0.97	0.91	0.86	0.67	0.96	0.92	0.97	0.88
HLA-A*24-02	1026840	binary	9	357	0.91	0.49	0.86	0.43	0.87	0.44	0.84	0.41	0.84	0.4	0.87	0.44	0.86	0.43	0.83	0.4	0.84	0.4	0.86	0.43	0.85	0.42	0.87	0.44
HLA-A*24-02	1026840	ic50	9	20	0.74	0.34	0.67	0.3	0.66	0.27	0.76	0.45	0.74	0.4	0.59	0.21	0.62	0.23	0.44	0.05	0.75	0.38	0.75	0.39	0.73	0.37	0.59	0.2
HLA-B*27-04	1029125	binary	9	21	0.98	0.78	0.99	0.8	0.99	0.8	0.83	0.53	-	-	-	-	-	-	-	-	-	-	0.95	0.73	-	-	0.94	0.72
HLA-A*30-02	1026840	binary	9	360	0.87	0.59	0.78	0.45	0.78	0.45	0.75	0.4	0.73	0.36	0.75	0.4	0.75	0.4	0.65	0.25	0.72	0.35	0.83	0.52	0.75	0.4	0.77	0.43
HLA-A*30-02	1026840	ic50	9	56	0.67	0.34	0.48	-0.02	0.49	-0.01	0.4	-0.1	0.54	0.12	0.59	0.13	0.54	0.03	0.64	0.27	0.52	0.07	0.8	0.58	0.55	0.08	0.51	0.05
HLA-A*30-02	1026840	t1/2	9	56	0.46	0.04	0.48	0.03	0.5	0.05	0.47	-0.01	0.5	0.07	0.55	0.19	0.48	0.07	0.52	0.15	0.53	0.12	0.57	0.19	0.48	0.07	0.5	0.09
HLA-A*02-02	1028790	ic50	9	55	0.84	0.69	0.75	0.59	0.76	0.61	0.72	0.53	0.71	0.56	0.74	0.64	0.72	0.59	0.73	0.49	0.74	0.57	0.91	0.86	0.71	0.56	0.75	0.62
HLA-A*02-02	1024516	ic50	9	51	0.86	0.7	0.78	0.6	0.79	0.62	0.7	0.5	0.69	0.55	0.74	0.65	0.72	0.6	0.76	0.51	0.73	0.57	0.93	0.86	0.68	0.56	0.75	0.63
HLA-B*27-05	1029125	binary	9	21	0.99	0.8	0.95	0.73	0.94	0.72	0.91	0.67	0.91	0.67	0.94	0.72	0.96	0.75	0.88	0.62	0.96	0.75	0.96	0.75	0.91	0.68	0.96	0.75
HLA-B*27-05	1031253	ic50	9	12	0.63	0.57	0.63	0.57	0.57	0.48	0.6	0.49	0.63	0.44	0.69	0.6	0.69	0.58	0.6	0.46	0.6	0.46	0.71	0.71	0.64	0.49	0.66	0.62
HLA-A*02-01	1033576	binary	9	27	0.9	0.5	0.87	0.46	0.88	0.47	0.89	0.48	0.89	0.48	0.92	0.52	0.91	0.51	0.78	0.35	0.89	0.48	0.9	0.5	0.88	0.47	0.9	0.5
HLA-A*02-01	1028554	ic50	9	44	0.74	0.46	0.88	0.71	0.88	0.72	0.88	0.51	0.89	0.58	0.82	0.62	0.81	0.6	0.76	0.51	0.86	0.55	0.91	0.63	0.84	0.5	0.89	0.69
HLA-A*02-01	1028553	ic50	9	22	1	0.61	0.85	0.59	0.85	0.57	0.83	0.58	0.85	0.59	0.91	0.7	0.92	0.63	0.9	0.69	0.83	0.62	0.97	0.76	0.85	0.58	0.92	0.64
HLA-A*02-01	1029824	binary	9	77	0.6	0.16	0.57	0.1	0.56	0.09	0.55	0.08	0.57	0.1	0.57	0.11	0.58	0.12	0.52	0.04	0.57	0.1	0.61	0.17	0.58	0.12	0.56	0.09
HLA-A*02-01	1033071	ic50	9	112	0.9	0.81	0.88	0.82	0.88	0.81	0.85	0.78	0.87	0.8	0.88	0.81	0.88	0.82	0.85	0.79	0.87	0.81	0.88	0.82	0.87	0.81	0.88	0.81
HLA-A*02-01	1026371	t1/2	9	85	0.8	0.54	0.81	0.57	0.81	0.57	0.79	0.54	0.81	0.56	0.82	0.58	0.81	0.57	0.81	0.56	0.81	0.56	0.82	0.6	0.8	0.52	0.82	0.57
HLA-A*02-01	1027079	binary	9	16	0.71	0.34	0.8	0.48	0.84	0.54	0.8	0.48	0.75	0.39	0.76	0.42	0.76	0.42	0.78	0.45	0.75	0.39	0.82	0.51	0.73	0.37	0.8	0.48
HLA-A*02-01	1027588	binary	9	19	0.81	0.53	0.85	0.6	0.84	0.58	0.74	0.42	0.84	0.58	0.84	0.57	0.86	0.62	0.77	0.47	0.85	0.6	0.85	0.6	0.84	0.58	0.83	0.56
HLA-A*02-01	1028790	ic50	9	55	0.68	0.67	0.58	0.61	0.58	0.63	0.66	0.59	0.56	0.63	0.57	0.61	0.56	0.61	0.57	0.58	0.55	0.61	-	-	0.51	0.56	0.57	0.61
HLA-A*02-01	1031894	ic50	9	431	0.81	0.57	0.9	0.74	0.9	0.74	0.78	0.57	0.82	0.65	0.88	0.72	0.88	0.72	0.76	0.54	0.82	0.65	0.86	0.75	0.82	0.64	0.88	0.73
HLA-A*02-01	1024516	ic50	9	51	0.74	0.7	0.63	0.65	0.63	0.67	0.66	0.59	0.64	0.69	0.64	0.67	0.63	0.67	0.65	0.66	0.63	0.68	0.71	0.75	0.57	0.62	0.63	0.67
HLA-A*02-01	1028928	binary	9	13	0.91	0.51	0.95	0.57	0.95	0.57	0.93	0.54	0.95	0.57	0.95	0.57	0.95	0.57	0.95	0.57	0.95	0.57	0.95	0.57	0.95	0.57	0.95	0.57
HLA-A*02-01	1027471	binary	9	45	0.94	0.51	0.84	0.4	0.84	0.4	0.86	0.43	0.85	0.42	0.85	0.41	0.86	0.43	0.85	0.41	0.85	0.42	0.9	0.47	0.85	0.41	0.86	0.42
HLA-A*02-01	1026840	binary	9	357	0.9	0.57	0.91	0.57	0.9	0.57	0.89	0.55	0.9	0.57	0.89	0.55	0.91	0.58	0.9	0.56	0.9	0.57	0.91	0.58	0.91	0.58	0.9	0.57
HLA-A*02-01	1026840	ic50	9	24	0.65	0.22	0.66	0.38	0.66	0.35	0.68	0.38	0.64	0.33	0.59	0.26	0.64	0.33	0.69	0.4	0.61	0.29	0.71	0.47	0.56	0.19	0.65	0.32
HLA-A*02-01	1026840	t1/2	9	24	0.72	0.45	0.76	0.48	0.73	0.43	0.67	0.39	0.75	0.38	0.69	0.32	0.76	0.41	0.71	0.45	0.73	0.35	0.75	0.48	0.69	0.28	0.72	0.4
HLA-A*02-01	1028285	t1/2	9	881	0.85	0.65	0.86	0.71	0.87	0.72	0.84	0.68	0.85	0.7	0.85	0.69	0.86	0.7	0.82	0.65	0.85	0.69	0.87	0.71	0.84	0.68	0.86	0.71
HLA-C*03-03	1028228	ic50	9	163	0.95	0.88	0.92	0.83	0.92	0.82	0.9	0.81	0.95	0.85	0.93	0.85	0.95	0.87	-	-	0.94	0.84	0.87	0.78	0.95	0.85	0.93	0.84
HLA-B*07-02	1028554	ic50	9	52	0.92	0.74	0.79	0.63	0.8	0.66	0.74	0.7	0.85	0.66	0.88	0.7	0.83	0.67	0.76	0.65	0.86	0.7	0.92	0.86	0.82	0.59	0.86	0.73
HLA-B*07-02	1028553	ic50	9	22	0.88	0.62	0.89	0.67	0.88	0.73	0.89	0.62	0.89	0.62	0.92	0.76	0.89	0.65	0.88	0.55	0.88	0.62	0.91	0.68	0.88	0.67	0.91	0.72
HLA-B*07-02	1031253	ic50	9	13	1	0.93	1	0.97	1	0.98	1	0.87	1	0.96	1	0.96	1	0.98	1	0.89	1	0.97	1	0.96	1	0.97	1	0.97
HLA-B*07-02	1026371	t1/2	9	43	0.93	0.81	0.97	0.86	0.98	0.87	0.88	0.65	0.96	0.79	0.96	0.84	0.94	0.82	0.78	0.53	0.96	0.81	0.96	0.85	0.96	0.79	0.95	0.84
HLA-B*07-02	1028928	binary	9	12	1	0.65	1	0.65	1	0.65	1	0.65	1	0.65	1	0.65	1	0.65	1	0.65	1	0.65	1	0.65	1	0.66	1	0.65
HLA-B*07-02	1026840	binary	9	296	0.95	0.43	0.89	0.37	0.89	0.38	0.88	0.36	0.9	0.39	0.9	0.38	0.9	0.38	0.88	0.37	0.9	0.39	-	-	0.89	0.38	0.89	0.38
HLA-A*68-02	1028790	ic50	9	55	0.86	0.63	0.81	0.54	0.81	0.54	0.8	0.45	0.81	0.54	0.83	0.56	0.83	0.55	0.76	0.45	0.8	0.52	0.83	0.58	0.8	0.53	0.83	0.55
HLA-A*68-02	1024516	ic50	9	51	0.9	0.68	0.85	0.58	0.86	0.58	0.86	0.54	0.83	0.56	0.86	0.59	0.86	0.59	0.8	0.47	0.83	0.55	0.86	0.6	0.83	0.56	0.85	0.58
HLA-C*04-01	1028229	ic50	9	153	0.97	0.52	0.94	0.48	0.95	0.49	0.85	0.39	0.97	0.56	1	0.58	1	0.56	-	-	1	0.59	0.85	0.4	0.97	0.55	1	0.59
HLA-B*35-01	1028554	ic50	9	56	0.82	0.54	0.67	0.4	0.66	0.36	0.5	0.28	0.59	0.21	0.57	0.27	0.69	0.27	0.64	0.26	0.53	0.2	0.81	0.5	0.7	0.29	0.64	0.36
HLA-C*03-04	315209	t1/2	9	14	0.8	0.29	1	0.88	1	0.86	0.87	0.72	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.91	0.78
HLA-A*03-01	1031253	ic50	9	14	1	0.78	0.93	0.77	0.96	0.8	0.96	0.71	0.87	0.69	0.89	0.73	0.96	0.79	0.71	0.47	0.91	0.75	0.96	0.83	0.9	0.73	0.91	0.76
HLA-C*14-02	1028234	ic50	9	218	0.9	0.8	0.87	0.77	0.86	0.75	0.83	0.67	0.87	0.77	0.93	0.88	0.91	0.81	-	-	0.88	0.78	0.88	0.74	0.88	0.77	0.91	0.85
HLA-B*38-01	1029957	ic50	9	27	0.95	0.78	0.95	0.81	0.97	0.79	0.97	0.78	0.99	0.84	1	0.84	0.98	0.82	0.65	0.1	0.99	0.84	0.95	0.8	0.98	0.84	1	0.84
HLA-B*27-03	315174	binary	9	11	0.79	0.48	0.82	0.54	0.89	0.66	0.89	0.66	-	-	-	-	-	-	-	-	0.5		0.96	0.78	-	-	0.89	0.66
HLA-B*39-06	1029957	ic50	9	33	0.94	0.89	0.79	0.55	0.71	0.4	0.75	0.54	-	-	-	-	-	-	-	-	-	-	0.92	0.79	-	-	0.81	0.59
HLA-B*27-06	1029125	binary	9	21	0.84	0.57	0.77	0.45	0.73	0.39	0.8	0.5	-	-	-	-	-	-	-	-	-	-	0.87	0.62	-	-	0.75	0.42
HLA-A*30-01	1026840	binary	9	349	0.96	0.24	0.8	0.16	0.83	0.17	0.75	0.13	0.79	0.15	0.77	0.14	0.83	0.17	0.71	0.11	0.78	0.15	0.91	0.21	0.73	0.12	0.79	0.15
HLA-A*02-06	1028790	ic50	9	55	0.79	0.66	0.78	0.63	0.78	0.62	0.75	0.53	0.73	0.59	0.76	0.62	0.76	0.62	0.73	0.57	0.73	0.59	0.88	0.82	0.73	0.59	0.77	0.64

Table 5.2: Performance Evaluation on IEDB Benchmark Database

Measure					Pan-specific								Allele-specific												Ensemble			
					MHCherryPan		NetMHCpan3		NetMHCpan4		PickPocket		SMM		NetMHC3.4		NetMHC4		ARB		SMMPMBEC		mhcflurry		IEDBCons		NetMHCcons	
HLA-A*02-06	1024516	ic50	9	51	0.85	0.75	0.83	0.72	0.83	0.7	0.76	0.55	0.79	0.68	0.81	0.71	0.82	0.71	0.78	0.66	0.79	0.68	0.92	0.87	0.78	0.68	0.83	0.73
HLA-C*15-02	1028235	ic50	9	181	0.92	0.75	0.94	0.77	0.93	0.77	0.87	0.74	0.94	0.85	0.97	0.87	0.93	0.84	-	-	0.94	0.86	0.9	0.7	0.94	0.86	0.96	0.85
HLA-A*66-01	315312	binary	9	8	0.58	0.13	0.33	-0.25	0.42	-0.13	0.5	0	0.5	0	0.5	0	0.33	-0.25	-	-	1	0.76	0.17	-0.5	0.5	0	0.5	0
HLA-A*31-01	315312	binary	9	10	1	0.85	0.96	0.7	1	0.85	0.92	0.71	0.92	0.71	0.92	0.71	0.92	0.71	0.88	0.64	0.92	0.71	-	-	0.92	0.71	0.92	0.71
HLA-B*15-02	1027131	binary	9	14	1	0.71	1	0.71	1	0.71	1	0.71	0.91	0.58	1	0.71	1	0.71	1	0.72	1	0.71	1	0.71	0.97	0.67	1	0.71
HLA-B*44-03	1028554	ic50	9	46	0.72	0.58	0.64	0.6	0.58	0.56	0.64	0.43	0.75	0.47	0.65	0.56	0.74	0.65	0.56	0.25	0.76	0.55	0.69	0.77	0.84	0.56	0.64	0.55
HLA-C*12-03	1028286	ic50	9	172	0.87	0.75	0.71	0.54	0.72	0.55	0.65	0.43	0.57	0.52	0.69	0.65	0.72	0.65	-	-	0.61	0.61	0.81	0.63	0.57	0.52	0.71	0.63
HLA-A*68-01	1026840	binary	9	436	0.93	0.45	0.86	0.37	0.85	0.36	0.88	0.4	0.86	0.37	0.87	0.38	0.89	0.4	0.78	0.34	0.88	0.39	0.88	0.39	0.88	0.39	0.87	0.39
HLA-A*68-01	1026840	ic50	9	35	0.87	0.68	0.84	0.6	0.85	0.61	0.86	0.64	0.79	0.62	0.84	0.65	0.85	0.68	0.77	0.53	0.83	0.63	0.87	0.71	0.82	0.66	0.85	0.65
HLA-A*68-01	1026840	t1/2	9	35	0.29	-0.38	0.38	-0.2	0.38	-0.22	0.24	-0.46	0.25	-0.42	0.27	-0.41	0.28	-0.4	0.31	-0.39	0.25	-0.43	0.27	-0.38	0.26	-0.42	0.29	-0.35
HLA-A*02-03	1028790	ic50	9	55	0.76	0.58	0.71	0.54	0.71	0.53	0.77	0.5	0.68	0.52	0.67	0.51	0.66	0.49	0.68	0.5	0.69	0.53	0.69	0.58	0.66	0.42	0.69	0.54
HLA-A*02-03	1024516	ic50	9	51	0.8	0.6	0.75	0.56	0.74	0.57	0.77	0.47	0.68	0.53	0.68	0.54	0.67	0.53	0.73	0.54	0.69	0.54	0.72	0.62	0.66	0.42	0.71	0.56
HLA-C*08-02	1028233	ic50	9	87	0.9	0.7	0.92	0.77	0.92	0.76	0.83	0.65	0.92	0.79	0.96	0.88	0.92	0.81	-	-	0.92	0.79	0.79	0.6	0.92	0.79	0.96	0.86
HLA-B*07-02	1026371	t1/2	10	25	0.71	0.58	0.77	0.62	0.77	0.6	0.73	0.59	0.73	0.58	0.74	0.58	0.82	0.71	0.72	0.57	0.73	0.61	0.78	0.67	0.74	0.6	0.76	0.62
HLA-A*02-06	1028790	ic50	10	35	0.87	0.74	0.82	0.57	0.81	0.56	0.66	0.39	0.79	0.56	0.75	0.55	0.82	0.51	0.76	0.49	0.8	0.59	0.88	0.82	0.78	0.54	0.76	0.57
HLA-A*02-06	1024516	ic50	10	32	0.9	0.74	0.88	0.58	0.86	0.57	0.76	0.46	0.84	0.58	0.82	0.57	0.88	0.5	0.85	0.53	0.85	0.59	0.91	0.82	0.85	0.57	0.84	0.6
HLA-A*02-01	1033071	ic50	10	75	0.91	0.82	0.9	0.86	0.91	0.86	0.88	0.82	0.89	0.83	0.91	0.84	0.91	0.87	0.86	0.8	0.89	0.83	0.91	0.83	0.89	0.86	0.9	0.84
HLA-A*02-01	1024516	ic50	10	32	0.71	0.66	0.62	0.53	0.62	0.52	0.66	0.44	0.67	0.54	0.69	0.55	0.64	0.53	0.64	0.53	0.62	0.52	0.85	0.75	0.67	0.54	0.69	0.55
HLA-A*02-01	1026371	t1/2	10	22	0.56	0.04	0.54	0.12	0.56	0.14	0.56	0.07	0.56	0.14	0.58	0.19	0.52	0.06	0.53	0.11	0.55	0.11	0.58	0.16	0.55	0.13	0.56	0.15
HLA-A*02-01	1028790	ic50	10	35	0.72	0.59	0.6	0.43	0.6	0.43	0.65	0.3	0.65	0.47	0.67	0.45	0.64	0.44	0.64	0.44	0.59	0.45	0.84	0.67	0.66	0.46	0.68	0.44
HLA-A*02-01	1028285	t1/2	10	62	0.62	0.26	0.7	0.44	0.7	0.44	0.68	0.43	0.69	0.4	0.68	0.38	0.69	0.41	0.68	0.45	0.68	0.39	0.7	0.42	0.68	0.4	0.72	0.45
HLA-A*68-02	1028790	ic50	10	35	0.75	0.41	0.62	0.36	0.66	0.42	0.66	0.29	0.72	0.38	0.7	0.43	0.65	0.46	0.7	0.31	0.69	0.34	0.67	0.41	0.7	0.37	0.66	0.33
HLA-A*68-02	1024516	ic50	10	32	0.88	0.43	0.76	0.41	0.81	0.47	0.78	0.31	0.82	0.41	0.81	0.47	0.78	0.51	0.78	0.37	0.81	0.36	0.82	0.48	0.83	0.42	0.76	0.35
HLA-A*02-03	1028790	ic50	10	35	0.72	0.31	0.76	0.27	0.76	0.26	0.7	0.07	0.73	0.34	0.78	0.31	0.76	0.29	0.68	0.29	0.7	0.31	0.92	0.69	0.7	0.28	0.79	0.29
HLA-A*02-03	1024516	ic50	10	32	0.72	0.36	0.77	0.36	0.77	0.36	0.71	0.16	0.72	0.31	0.79	0.38	0.78	0.4	0.73	0.36	0.7	0.3	0.92	0.73	0.71	0.3	0.81	0.38
HLA-A*02-01	1028285	t1/2	11	38	0.75	0.47	0.58	0.18	0.62	0.26	0.68	0.32	0.75	0.36	0.67	0.2	0.5	0.02	0.65	0.1	0.69	0.2	0.7	0.34	0.75	0.36	0.66	0.21
Highlighed	Counts				29	17	8	5	14	8	6	2	4	3	16	10	11	9	3	2	7	4	30	30	7	5	10	4

5.2.1 Average AUC and SRCC

Besides counting the highest numbers of each dataset in terms of AUC and SRCC, we also calculate the average AUC and SRCC for each algorithm based on the data in Table 5.2. As we can see in the figure 5.2, our model and MHCFlurry both achieved the best average (0.83) in AUC comparing with other algorithms using IEDB Benchmark Datasets. Our SRCC ranked the second and performed a little bit less than MHCFlurry did. If only comparing within the pan-specific algorithms, our model outperformed all the rest pan-specific models. Those results are consistent with previous analysis by counting the highest numbers. In total, our proposed MHCherryPan could thus be an excellent supplementary algorithm for current existing pan-specific models.

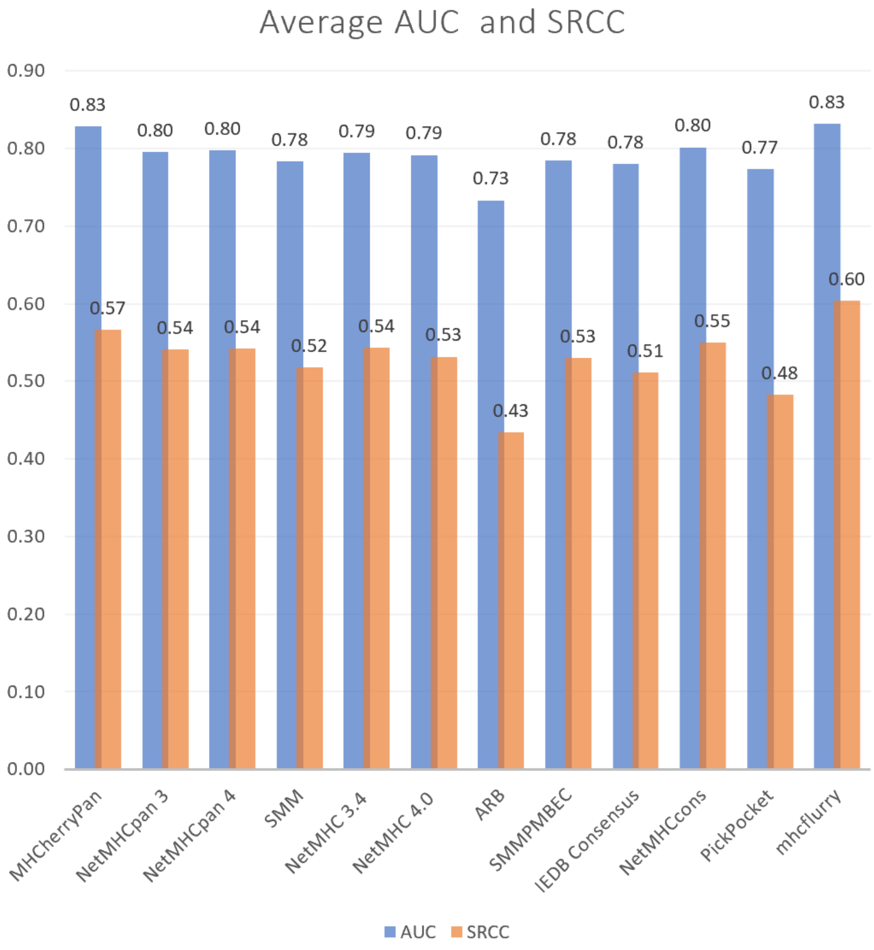


Figure 5.2: **Average AUC and SRCC Bar Chart**

Chapter 6

Conclusion

In this chapter, We conclude the work presented in this thesis. In the end, we discuss the future work.

6.1 Conclusion

In this thesis, we have developed MHCherryPan, a novel model combining convolutional and recurrent neural networks for pan-specific HLA-peptide binding affinity prediction. Our model is composed of an RNN-based peptide feature extractor, a CNN-based HLA feature extractor and a binding predictor. The extensive benchmark experiments on HLA-I peptide binding prediction demonstrated that MHCherryPan achieved state-of-the-art performance over a majority of the benchmark datasets from IEDB. MHCherryPan is characterized by its capability of binding prediction with only the raw amino acid sequences of HLA and peptides, which makes it applicable to HLA-peptide binding prediction for HLA alleles without structural information. Our main contribution is to propose a novel design combining CNN-based and RNN-based networks inside one pan-specific model. Instead of using the consensus model to combine the different results from various model architectures, we bring another possibility to combine the features from different networks in one model.

6.2 Future Work

Our thesis work may be enhanced in the following. First, in this thesis, merely raw amino acid sequence data is used for representing the input peptide and HLA protein. However, this can be improved by including MS for the peptide. Secondly, we can further test the performance of our MHCherryPan on alleles where few data is available. Moreover, combining our model with current software to create a pipeline to identify neoantigens may be another meaningful work in the future.

Bibliography

[1] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831, 2015.

[2] Massimo Andreatta and Morten Nielsen. Gapped sequence alignment using artificial neural networks: application to the mhc class i system. *Bioinformatics*, 32(4):511–517, 2016.

[3] Rohit Bhattacharya, Ashok Sivakumar, Collin Tokheim, Violeta Beleva Guthrie, Valsamo Anagnostou, Victor E Velculescu, and Rachel Karchin. Evaluation of machine learning methods to predict peptide binding to mhc class i proteins. *bioRxiv*, page 154757, 2017.

[4] Bruno De Deken. Predicting mhc binding preferences using recurrent neural networks, 2017. [Online; accessed 17-February-2020].

[5] Scott R Burrows, Jamie Rossjohn, and James McCluskey. Have we cut ourselves too short in mapping ctl epitopes? *Trends in immunology*, 27(1):11–16, 2006.

[6] Sung Yoon Choo. The hla system: genetics, immunology, clinical testing, and clinical implications. *Yonsei medical journal*, 48(1):11–23, 2007.

[7] Pouya Faridi, Chen Li, Sri H Ramarathinam, Julian P Vivian, Patricia T Illing, Nicole A Mifsud, Rochelle Ayala, Jiangning Song, Linden J Gearing, Paul J Hertzog, et al. A

subset of hla-i peptides are not genomically templated: Evidence for cis-and trans-spliced peptide ligands. *Science immunology*, 3(28):eaar3947, 2018.

[8] Gary S Firestein, Ralph Budd, Sherine E Gabriel, Iain B McInnes, and James R O'Dell. *Kelley and Firestein's Textbook of Rheumatology E-Book*. Elsevier Health Sciences, 2016.

[9] Ward Fleri, Sinu Paul, Sandeep Kumar Dhanda, Swapnil Mahajan, Xiaojun Xu, Bjoern Peters, and Alessandro Sette. The immune epitope database and analysis resource in epitope discovery and synthetic vaccine design. *Frontiers in immunology*, 8:278, 2017.

[10] Pingping Guan, Irini A Doytchinova, Christianna Zygouri, and Darren R Flower. Mhcpred: a server for quantitative prediction of peptide–mhc binding. *Nucleic acids research*, 31(13):3621–3624, 2003.

[11] Youngmahn Han and Dongsup Kim. Deep convolutional neural networks for pan-specific peptide-mhc class i binding prediction. *BMC bioinformatics*, 18(1):585, 2017.

[12] Steven Henikoff and Jorja G Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

[13] C Janeway, P Travers, M Walport, and M Shlomchik Immunobiology. Garland science publishing. *New York, NY*, 2005.

[14] Charles A Janeway, Paul Travers, Mark Walport, Mark Shlomchik, et al. *Immunobiology: the immune system in health and disease*, volume 7. Current Biology London, 1996.

[15] Jordi TORRES. Learning process of a neural network, 2018. [Online; accessed 17-February-2020].

[16] Vanessa Isabell Jurtz, Alexander Rosenberg Johansen, Morten Nielsen, Jose Juan Almagro Armenteros, Henrik Nielsen, Casper Kaae Sønderby, Ole Winther, and Søren Kaae Sønderby. An introduction to deep learning on biological sequence data: examples and solutions. *Bioinformatics*, 33(22):3685–3690, 2017.

[17] Edita Karosiene, Claus Lundegaard, Ole Lund, and Morten Nielsen. Netmhccons: a consensus method for the major histocompatibility complex class i predictions. *Immunogenetics*, 64(3):177–186, 2012.

[18] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[19] Zhonghao Liu, Yuxin Cui, Zheng Xiong, Alierza Nasiri, Ansi Zhang, and Jianjun Hu. Deepseqpan, a novel deep convolutional neural network model for pan-specific class i hla-peptide binding affinity prediction. *Scientific reports*, 9(1):794, 2019.

[20] Yong-Chen Lu and Paul F Robbins. Cancer immunotherapy targeting neoantigens. In *Seminars in immunology*, volume 28, pages 22–27. Elsevier, 2016.

[21] Claus Lundegaard, Kasper Lamberth, Mikkel Harndahl, Søren Buus, Ole Lund, and Morten Nielsen. Netmhc-3.0: accurate web accessible predictions of human, mouse and monkey mhc class i affinities for peptides of length 8–11. *Nucleic acids research*, 36(suppl_2):W509–W512, 2008.

[22] Heng Luo, Hao Ye, Hui Wen Ng, Sugunadevi Sakkiah, Donna L Mendrick, and Huixiao Hong. snebula, a network-based algorithm to predict binding between human leukocyte antigens and peptides. *Scientific reports*, 6:32115, 2016.

[23] Shutao Mei, Fuyi Li, André Leier, Tatiana T Marquez-Lago, Kailin Giam, Nathan P Croft, Tatsuya Akutsu, A Ian Smith, Jian Li, Jamie Rossjohn, et al. A comprehensive review and performance evaluation of bioinformatics tools for hla class i peptide-binding prediction. *Brief Bioinform*, 10, 2019.

[24] Seonwoo Min, Byunghan Lee, and Sungroh Yoon. Deep learning in bioinformatics. *Briefings in bioinformatics*, 18(5):851–869, 2017.

[25] Mohit Sharma. What steps should one take while doing data preprocessing?, 2018. [Online; accessed 17-February-2020].

[26] Magdalini Moutaftsi, Bjoern Peters, Valerie Pasquetto, David C Tscharke, John Sidney, Huynh-Hoa Bui, Howard Grey, and Alessandro Sette. A consensus epitope prediction approach identifies the breadth of murine t cd8+-cell responses to vaccinia virus. *Nature biotechnology*, 24(7):817, 2006.

[27] 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. *Genome medicine*, 8(1):33, 2016.

[28] Timothy J O'Donnell, Alex Rubinsteyn, Maria Bonsack, Angelika B Riemer, Uri Laserson, and Jeff Hammerbacher. Mhcflurry: open-source class i mhc binding affinity prediction. *Cell systems*, 7(1):129–132, 2018.

[29] Bjoern Peters and Alessandro Sette. Generating quantitative models describing the sequence specificity of biological processes with the stabilized matrix method. *BMC bioinformatics*, 6(1):132, 2005.

[30] James Robinson, Jason A Halliwell, James D Hayhurst, Paul Flicek, Peter Parham, and Steven GE Marsh. The ipd and imgt/hla database: allele variant databases. *Nucleic acids research*, 43(D1):D423–D431, 2014.

[31] Alessandro Sette, Antonella Vitiello, Barbara Reherman, Patricia Fowler, Ramin Nayersina, W Martin Kast, CJ Melief, Carla Oseroff, Lunli Yuan, Jorg Ruppert, et al. The relationship between class i binding affinity and immunogenicity of potential cytotoxic t cell epitopes. *The Journal of Immunology*, 153(12):5586–5592, 1994.

[32] Satarudra Prakash Singh and Bhartendu Nath Mishra. Major histocompatibility complex linked databases and prediction tools for designing vaccines. *Human immunology*, 77(3):295–306, 2016.

[33] Jugmohit S Toor, Arjun A Rao, Andrew C McShan, Mark Yarmarkovich, Santrupti Nerli, Karissa Yamaguchi, Ada A Madejska, Son Nguyen, Sarvind Tripathi, John M Maris, et al. a recurrent mutation in anaplastic lymphoma kinase with distinct neoepitope conformations. *Frontiers in immunology*, 9:99, 2018.

[34] Thomas Trolle, Imir G Metushi, Jason A Greenbaum, Yohan Kim, John Sidney, Ole Lund, Alessandro Sette, Bjoern Peters, and Morten Nielsen. Automated benchmarking of peptide-mhc class i binding predictions. *Bioinformatics*, 31(13):2174–2181, 2015.

[35] Yeeleng S Vang and Xiaohui Xie. Hla class i binding prediction via convolutional neural networks. *Bioinformatics*, 33(17):2658–2665, 2017.

[36] Ingrid Wagner and Hans Musso. New naturally occurring amino acids. *Angewandte Chemie International Edition in English*, 22(11):816–828, 1983.

[37] Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[38] Wikipedia contributors. Major histocompatibility complex — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Major_histocompatibility_complex.

[39] Wikipedia contributors. Blosum — Wikipedia, the free encyclopedia, 2019. [Online; accessed 17-February-2020].

[40] Wikipedia contributors. Receiver operating characteristic — Wikipedia, the free encyclopedia, 2019. [Online; accessed 10-November-2019].

[41] Wikipedia contributors. Recurrent neural network — Wikipedia, the free encyclopedia, 2019. [Online; accessed 27-October-2019].

[42] Wikipedia contributors. Box plot — Wikipedia, the free encyclopedia, 2020. [Online; accessed 23-February-2020].

[43] Wikipedia contributors. Convolutional neural network — Wikipedia, the free encyclopedia, 2020. [Online; accessed 27-April-2020].

[44] Wikipedia contributors. Deep learning — Wikipedia, the free encyclopedia, 2020. [Online; accessed 26-April-2020].

[45] Wikipedia contributors. False positives and false negatives — Wikipedia, the free encyclopedia, 2020. [Online; accessed 23-February-2020].

[46] Wikipedia contributors. Learning rate — Wikipedia, the free encyclopedia, 2020. [Online; accessed 27-April-2020].

[47] Wikipedia contributors. Precision and recall — Wikipedia, the free encyclopedia, 2020. [Online; accessed 27-April-2020].

[48] Wikipedia contributors. Sensitivity and specificity — Wikipedia, the free encyclopedia, 2020. [Online; accessed 27-April-2020].

[49] Wikipedia contributors. Sharcnet — Wikipedia, the free encyclopedia, 2020. [Online; accessed 23-February-2020].

[50] Wikipedia contributors. Spearman's rank correlation coefficient — Wikipedia, the free encyclopedia, 2020. [Online; accessed 28-April-2020].

[51] Zhoujian Xiao, Yuwei Zhang, Runsheng Yu, Yin Chen, Xiaosen Jiang, Ziwei Wang, and Shuaicheng Li. In silico design of mhc class i high binding affinity peptides through motifs activation map. *BMC bioinformatics*, 19(19):516, 2018.

[52] Andreas Zell, Niels Mache, Ralf Huebner, Günter Mamier, Michael Vogt, Michael Schmalzl, and Kai-Uwe Herrmann. Snns (stuttgart neural network simulator). In *Neural network simulation environments*, pages 165–186. Springer, 1994.

[53] Hao Zhang, Ole Lund, and Morten Nielsen. The pickpocket method for predicting binding specificities for receptors based on receptor pocket similarities: application to mhc-peptide binding. *Bioinformatics*, 25(10):1293–1299, 2009.

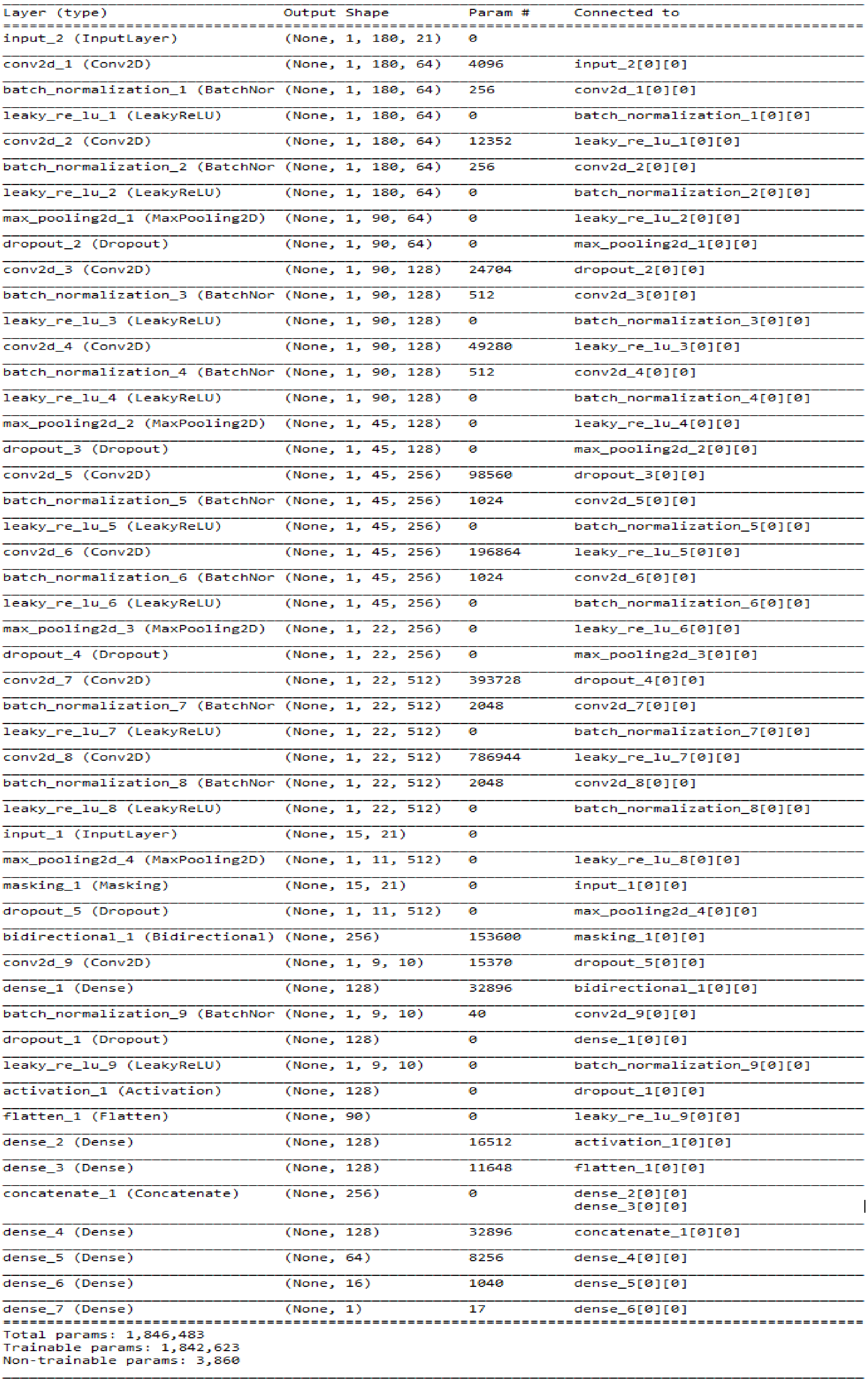
[54] Lianming Zhang, Keiko Udaka, Hiroshi Mamitsuka, and Shanfeng Zhu. Toward more accurate pan-specific mhc-peptide binding prediction: a review of current methods and tools. *Briefings in bioinformatics*, 13(3):350–364, 2011.

[55] Jacek M Zurada. *Introduction to artificial neural systems*, volume 8. West St. Paul, 1992.

Appendix A

MHCherryPan Model Summary

In the following, we provides the detailed architecture summaries for the MHCherryPan model which we discussed in Chapter 3 and Chapter 4.



```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_2 (InputLayer)            (None, 1, 180, 21)   0
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 1, 180, 64)   4096        input_2[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 1, 180, 64)   256         conv2d_1[0][0]
__________________________________________________________________________________________________
leaky_re_lu_1 (LeakyReLU)       (None, 1, 180, 64)   0           batch_normalization_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 1, 180, 64)   12352       leaky_re_lu_1[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 1, 180, 64)   256         conv2d_2[0][0]
__________________________________________________________________________________________________
leaky_re_lu_2 (LeakyReLU)       (None, 1, 180, 64)   0           batch_normalization_2[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 1, 90, 64)    0           leaky_re_lu_2[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 1, 90, 64)    0           max_pooling2d_1[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 1, 90, 128)   24704       dropout_2[0][0]
__________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 1, 90, 128)   512         conv2d_3[0][0]
__________________________________________________________________________________________________
leaky_re_lu_3 (LeakyReLU)       (None, 1, 90, 128)   0           batch_normalization_3[0][0]
__________________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 1, 90, 128)   49280       leaky_re_lu_3[0][0]
__________________________________________________________________________________________________
batch_normalization_4 (BatchNor (None, 1, 90, 128)   512         conv2d_4[0][0]
__________________________________________________________________________________________________
leaky_re_lu_4 (LeakyReLU)       (None, 1, 90, 128)   0           batch_normalization_4[0][0]
__________________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)  (None, 1, 45, 128)   0           leaky_re_lu_4[0][0]
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 1, 45, 128)   0           max_pooling2d_2[0][0]
__________________________________________________________________________________________________
conv2d_5 (Conv2D)               (None, 1, 45, 256)   98560       dropout_3[0][0]
__________________________________________________________________________________________________
batch_normalization_5 (BatchNor (None, 1, 45, 256)   1024        conv2d_5[0][0]
__________________________________________________________________________________________________
leaky_re_lu_5 (LeakyReLU)       (None, 1, 45, 256)   0           batch_normalization_5[0][0]
__________________________________________________________________________________________________
conv2d_6 (Conv2D)               (None, 1, 45, 256)   196864      leaky_re_lu_5[0][0]
__________________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, 1, 45, 256)   1024        conv2d_6[0][0]
__________________________________________________________________________________________________
leaky_re_lu_6 (LeakyReLU)       (None, 1, 45, 256)   0           batch_normalization_6[0][0]
__________________________________________________________________________________________________
max_pooling2d_3 (MaxPooling2D)  (None, 1, 22, 256)   0           leaky_re_lu_6[0][0]
__________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 1, 22, 256)   0           max_pooling2d_3[0][0]
__________________________________________________________________________________________________
conv2d_7 (Conv2D)               (None, 1, 22, 512)   393728      dropout_4[0][0]
__________________________________________________________________________________________________
batch_normalization_7 (BatchNor (None, 1, 22, 512)   2048        conv2d_7[0][0]
__________________________________________________________________________________________________
leaky_re_lu_7 (LeakyReLU)       (None, 1, 22, 512)   0           batch_normalization_7[0][0]
__________________________________________________________________________________________________
conv2d_8 (Conv2D)               (None, 1, 22, 512)   786944      leaky_re_lu_7[0][0]
__________________________________________________________________________________________________
batch_normalization_8 (BatchNor (None, 1, 22, 512)   2048        conv2d_8[0][0]
__________________________________________________________________________________________________
leaky_re_lu_8 (LeakyReLU)       (None, 1, 22, 512)   0           batch_normalization_8[0][0]
__________________________________________________________________________________________________
input_1 (InputLayer)            (None, 15, 21)       0
__________________________________________________________________________________________________
max_pooling2d_4 (MaxPooling2D)  (None, 1, 11, 512)   0           leaky_re_lu_8[0][0]
__________________________________________________________________________________________________
masking_1 (Masking)             (None, 15, 21)       0           input_1[0][0]
__________________________________________________________________________________________________
dropout_5 (Dropout)             (None, 1, 11, 512)   0           max_pooling2d_4[0][0]
__________________________________________________________________________________________________
bidirectional_1 (Bidirectional) (None, 256)          153600      masking_1[0][0]
__________________________________________________________________________________________________
conv2d_9 (Conv2D)               (None, 1, 9, 10)     15370       dropout_5[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 128)          32896       bidirectional_1[0][0]
__________________________________________________________________________________________________
batch_normalization_9 (BatchNor (None, 1, 9, 10)     40          conv2d_9[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 128)          0           dense_1[0][0]
__________________________________________________________________________________________________
leaky_re_lu_9 (LeakyReLU)       (None, 1, 9, 10)     0           batch_normalization_9[0][0]
__________________________________________________________________________________________________
activation_1 (Activation)       (None, 128)          0           dropout_1[0][0]
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 90)           0           leaky_re_lu_9[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 128)          16512       activation_1[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 128)          11648       flatten_1[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 256)          0           dense_2[0][0]
                                                                 dense_3[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 128)          32896       concatenate_1[0][0]
__________________________________________________________________________________________________
dense_5 (Dense)                 (None, 64)           8256        dense_4[0][0]
__________________________________________________________________________________________________
dense_6 (Dense)                 (None, 16)           1040        dense_5[0][0]
__________________________________________________________________________________________________
dense_7 (Dense)                 (None, 1)            17          dense_6[0][0]
==================================================================================================
Total params: 1,846,483
Trainable params: 1,842,623
Non-trainable params: 3,860
__________________________________________________________________________________________________
```

Curriculum Vitae

Name:	Xuezhi Xie
Post-Secondary Education and Degrees:	University of Waterloo Waterloo, On, Canada 2012 - 2014 BscH
	University of Western Ontario London, ON 2018 - 2020 Msc
Honours and Awards:	Western Graduate Research Scholarships(WGRS) 2018-2020
Related Work Experience:	Teaching Assistant The University of Western Ontario 2018 - 2020

Publications:

MHCherryPan, a novel model to predict the binding affinity of pan-specific class I HLA-peptide, accepted by IEEE - BIBM conference 2019 (in press).