2018

Machine Learning Based Disease Gene Identification and MHC Immune Protein-peptide Binding Prediction

Zhonghao Liu
University of South Carolina - Columbia

Machine Learning Based Disease Gene Identification and MHC Immune Protein-peptide Binding Prediction

by

Zhonghao Liu

Bachelor of Software Engineering
Sichuan University China 2012

Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy in

Computer Science

College of Engineering and Computing

University of South Carolina

2018

Accepted by:

Jianjun Hu, Major Professor

John R. Rose, Committee Member

Jijun Tang, Committee Member

Yan Tong, Committee Member

Hexin Chen, Committee Member

Cheryl L. Addy, Vice Provost and Dean of the Graduate School

Acknowledgments

I would like to thank Dr. Jianjun Hu for his guidance and encouragement on my research journey. He enlightened me on my research topics in so many ways. And I am always inspired by how Dr. Hu pursuing the facts implied by the results and data. His passion of applying machine learning and computer techniques to building a better world is one of the most important lessons I have learned in my graduate studies.

Also I want to give thanks to my dissertation committee members, Dr. Yan Tong, Dr. John R. Rose, Dr. Jijun Tang and Dr. Hexin Chen, for their time and advice on my work. It is such an honor to have them serving in my committee.

I cannot be more thankful and grateful that my parents are always there to support their son and give all the encouragement and love they have without which I cannot make any of this happen.

Last but not least, thanks my wife Yang Deng for sacrificing a lot during these years and pouring her love on me. Without her accompanying, my life would never be so complete and meaningful.

Abstract

Machine learning and deep learning methods have been increasingly applied to solve challenging and important bioinformatics problems such as protein structure prediction, disease gene identification, and drug discovery. However, the performances of existing machine learning based predictive models are still not satisfactory. The question of how to exploit the specific properties of bioinformatics data and couple them with the unique capabilities of the learning algorithms remains elusive. In this dissertation, we propose advanced machine learning and deep learning algorithms to address two important problems: mislocation-related cancer gene identification and major histocompatibility complex-peptide binding affinity prediction.

Our first contribution proposes a kernel-based logistic regression algorithm for identifying potential mislocation-related genes among known cancer genes. Our algorithm takes protein-protein interaction networks, gene expression data, and subcellular location gene ontology data as input, which is particularly lightweight comparing with existing methods. The experiment results demonstrate that our proposed pipeline has a good capability to identify mislocation-related cancer genes.

Our second contribution addresses the modeling and prediction of human leukocyte antigen (HLA) peptide binding of human immune system. We present an allele-specific convolutional neural network model with one-hot encoding. With extensive evaluation over the standard IEDB datasets, it is shown that the performance of our model is better than all existing prediction models. To achieve further improvement, we propose a novel pan-specific model on peptide-HLA class I binding affinities prediction, which allows us to exploit all the training samples of different HLA alleles.

Our sequence based pan model is currently the only algorithm not using pseudo sequence encoding — a dominant structure-based encoding method in this area. The benchmark studies show that our method could achieve state-of-the-art performance. Our proposed model could be integrated into existing ensemble methods to improve their overall prediction capabilities on highly diverse MHC alleles.

Finally, we present a LSTM-CNN deep learning model with attention mechanism for peptide-HLA class II binding affinities and binding cores prediction. Our model achieved very good performance and outperformed existing methods on half of tested alleles. With the help of attention mechanism, our model could directly output the peptide binding core based on attention weight without any additional post- or pre-processing.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

In 1902, Archibald Garrod published the paper *The Incidence of Alkaptonuria: A Study of Chemical Individuality* which confirmed the inheritance of the disorder *alkaptonuria* (better known as black urine disease) in men. But it would wait 90 years later, a mutation in HGD gene were demonstrated and elucidated as the genetic basis of this disorder [9, 37]. The number of disease genes discovered has grown steadily in past years [37]. The large-scale genome sequencing and high-throughput sequencing provide a vast amount of data and bioinformatics has changed the way searching for disease genes. In the work of [44], researchers rediscovered nearly all known cancer genes of 21 tumor types in a large-scale genomic analysis. Beyond this, researchers are targeting more specific underlying mechanisms of how disease gene affect the proteins' subcecullar locations [51, 77].

With the advent of genomic sciences, rapid DNA sequencing, combinatorial chemistry, cell-based assays, and automated high-throughput screening (HTS), the way we design and discover drugs to diseases has also been dramatically shaped and enriched [16]. With known drug target (usually protein receptor or enzyme) and associated structure, virtual screening is a widely used computational technique to filter small molecule candidates from a large number of libraries of compounds [30, 83]. Actually, virtual screening has become an integral part of the drug discovery process [27]. Usually, virtual screening in drug discovery consists of docking, which decides where to bind, and scoring, which reports the binding affinity of a protein-peptide interaction.

1.2 Scope of the Proposed Research

In this dissertation proposal, we focus on two specific topics:

1) As described before, there are quite a few effective models and methods which can identify cancer genes. With so many cancer genes discovered, people are digging

into them trying to find out what's the effect of these cancer genes in cellular activities. In this work, we are interested in one type of cancer genes whose mutations or abnormal expressions lead to mislocation of its translated proteins. And we propose a computational pipeline to help identify mislocation related cancer genes.

2) Another part of our work is major histocompatibility complex-peptide (MHC-peptide) binding prediction. More specifically, we are working on class I MHC-peptide binding affinity prediction. The topic interests us because recent cancer immunotherapy studies show that peptide-based vaccine is a promising for cancer treatment and it requires a more reliable and improved computational tool to predict which peptide will bind to MHC proteins [74, 56, 101, 79, 1]. In this dissertation, we present two models to address this problem. The first one is an allele-specific model which outperforms all existing methods. Next, we propose a pan-specific model which could be used to predict on all known MHC class I alleles.

1.3 Structure of the Dissertation

In chapter 2, we present a pipeline of identifying mislocation-related cancer genes and show the potential gene candidates we found in our experiments. In Chapter 3, we introduce the CNN model we proposed for peptide-MHC I binding prediction and show our performance comparison with other methods. In Chapter 5, we present our sequence based pan-specific model and list the results on benchmark dataset. Finally, we conclude our work and discuss future work in Chapter 6.

Chapter 2

Mislocalization-related Disease Gene Discovery

2.1 Introduction

Discovering disease or cancer genes and understanding their underlying pathological mechanisms are the major challenges of biomedical research. In the past decades, many efforts have been devoted to disease gene discovery using either high-throughput techniques [45] or computational disease gene prediction methods [55, 17]. However, these approaches usually report dozens or even hundreds of candidate genes while the experimental validation of many candidates is often an expensive and time-consuming task. To address this issue, many computational candidate gene prioritization algorithms have been developed by exploiting the biomedical knowledge available about the disease of interest and related genes [32]. For example, network information and heterogeneous phenomic and genomic datadata sources have been used to rank candidate genes [72, 105, 111, 22, 33]. A recent work [32] integrates a plethora of phenomic and genomic data to pinpoint disease genes including disease phenotype similarity derived from the Unified Medical Language System (UMLS) and seven types of gene functional similarities calculated from gene expression, gene ontology, pathway membership, protein sequence, protein domain, protein-protein interaction and regulation pattern, respectively. Their methods thus covered a total of 7,719 diseases and 20,327 genes. However, such studies brought limited insights into the pathological mechanisms due to their generic nature.

Protein Subcellular Location Prediction

The Eukaryotic cell, as the basic structural and functional unit of eukaryotic living organisms, contains numerous proteins located in different subcellular organelles or compartments, such as nuclei, mitochondria, cytoplasm, and Golgi apparatus (Figure 2.1). Binder et al. developed a comprehensive database to incorporate all data sources of subcellular localization [11]. Protein translocation is key for a cell to function normally because proteins must be transported to their targeted compartment to

exert their functions. The precise trafficking and translocation of cytosolic proteins to their final destinations are crucial for the maintenance of appropriate cell functions and activities. Proteins are typically directed to compartments by short peptide sequences that act as targeting signals. Translocation to the proper compartment allows a protein to form the necessary interactions with its partners and take part in biological networks such as signaling and metabolic pathways. If a protein is not localized to its correct intracellular compartment, either the reaction performed or information carried by the protein does not reach the proper site, causing either inactivation of central reactions or misregulation of signaling cascades, or the mislocalized active protein has harmful effects by acting in the wrong place. This work is focused on identifying cancer genes that may cause their translated proteins mislocation in cells.

The protein sequence contains the information regulating protein trafficking [67]. Based on proteins' target destinations, there are two major mechanisms of translocation: post-translational translocation and co-translational translocation [29, 86, 78, 102]. Proteins translocating to peroxisomes, mitochondria or the nucleus are called post-translational translocation [102, 85, 92, 94, 104]. For proteins translocated into the endoplasmic reticulum during synthesis, a process known as co-translational translocation is used [102]. Subcellular localization determines the access of proteins to interacting partners and the post-translational modification machinery and enables the integration of proteins into functional biological network [29].

However, proteins not always go to the correct locations and mislocalized proteins can lead to disease like Alzheimer's disease, kidney stones and cancers [29]. This is because failure to be transported to the correct subcellular compartment can have adverse effects, as protein location is fundamental to the functioning of cells and regulatory control in the disease [65]. In [98], three major mechanisms were identified that can lead to protein mislocation: mislocation through alterations of the protein

trafficking machinery [24], mislocation through altered protein targeting signals [43] and mislocation through changes in protein interaction or modification [29].

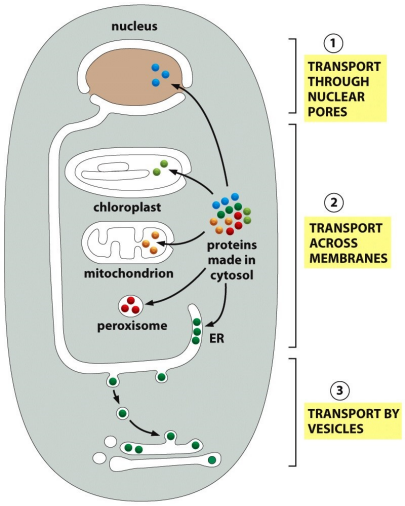


Figure 2.1 Subcellular Protein Sorting

2.1.1 Related Work

In the past decade, high-throughput methodological approaches such as cancer genome sequencing, RNA sequencing, cancer exome sequencing and functional genomics [55] have led to the identification of multiple cancer-causing genes, genetic alterations and deregulated pathways. Identifying and understanding the underlying mechanisms are the foundation for cancer diagnostics, therapeutics, clinical-trial design and selection of rational combination therapies [61, 66, 35, 107, 60, 95]. These hight-throughput experimental approaches plus the computational disease gene prioritization tools can greatly help to pin-down the range of cancer genes for experimental studies.

In a recent genome-wide study, Lawrence et al. [46] analyzed 4742 human cancers and their matched normal-tissue exome sequences accross 21 tumor types. This large-scale genomic analysis identified nearly all known cancer genes in 21 tumor types. Additionally, out of 81 "novel" candidate cancer genes, 33 genes are not previously

reported as significantly mutated in cancer. Based on close examination, they found at least 21 novel genes with strong and consistent connections to cancer. With more cancer sequencing, it is expected that a comprehensive catalog of cancer genes can be discovered to enable physicians to select best therapy fore each patient. However, experimental studies of these newly identified candidate genes are needed to verify and uncover the underlying mechanisms of these cancer-causing genes.

Lawrence et al.'s study [46] showed the power of tumor-normal pairs of the genome sequence data for discovery of candidate cancer genes, which inspired us to explore how to apply the tumor-normal pairs of gene expression data for identifying the cancer genes that may lead to mislocalization of proteins. Actually, Pinto et al. showed that dynamic redistribution of multitudinous proteins to different subcellular locations in response to cellular functional state is a crucial characteristic of cellular function that seems to be at least as important as overall changes in protein abundance [77]. Laurila and Vihinen [43] applied bioinformatics methods to investigate the effects of known disease-related mutations on protein targeting and localization by analyzing over 22,000 missense mutations in more than 1500 proteins with two complementary prediction approaches. However, many of the localization prediction algorithms that they used are not sensitive enough to capture the subtle sequence mutation to give different localization predictions. Lee et al. [51] proposed an integrative computational framework for mapping conditional location and mislocation of proteins on a proteome-wide scale. They mapped the locations of over 10,000 proteins in normal human brain and in glioma, out of which over 150 have a strong likelihood of mislocation under glioma. Fifteen of these mislocations have been confirmed. The most common type of mislocation occurs between the endoplasmic reticulum and the nucleus.

High-throughput sequencing has almost identified all known and potentially new cancer genes based on tomour-normal tissue sequences. It implies that the major

task now is on computational or experimental techniques to elucidate the underlying pathologic mechanisms of these cancer-causing genes. In this work, we are interested in discovering cancer genes which impact humans due to their mislocation within the cell. Lee's pipeline was proved successful in discovery of mislocalized proteins in cancer. However the data needed in Lee's pipeline were complex including sequences, chemistry, motifs and gene function ontology [50, 51]. Considering the data source and lacking of ready-to-use server or software, it is very difficult to use their pipeline to do large scale screening. When predicting conditional localization, we found that in Lee's method the only evidence source that has changed between normal and disease status is gene expression. It is thus natural to develop methods for conditional localization prediction based only on the changing factor, gene expression. Compared to Lee's approach, our approach only depends on gene expression files and gene ontology, which makes it applicable for broader range of cancers. As shown in Figure 2.2, our pipeline used gene ontology and gene expression files to predict proteins' subcellular locations in disease/tumor and normal states. Based on predicted locations of two states, we can identify proteins whose subcellular locations change dramatically. The reason we only use gene expression data as input is that we are trying to predict the condition-specific mislocations. If any gene in our candidates is a known cancer gene, it may cause cancer due to its mislocation within the cell. Our work aims to open a way to explore the complex relationship between protein subcellular locations and disease phenotypes.

2.2 Methodology

The main idea of this study is to predict subcellular localizations of proteins in normal and disease states cells and then filter proteins that are mislocated in disease state cells. Then we can filter mislocation proteins translated from known cancer genes and mark these cancer genes as mislocation-related genes. The overall procedure of

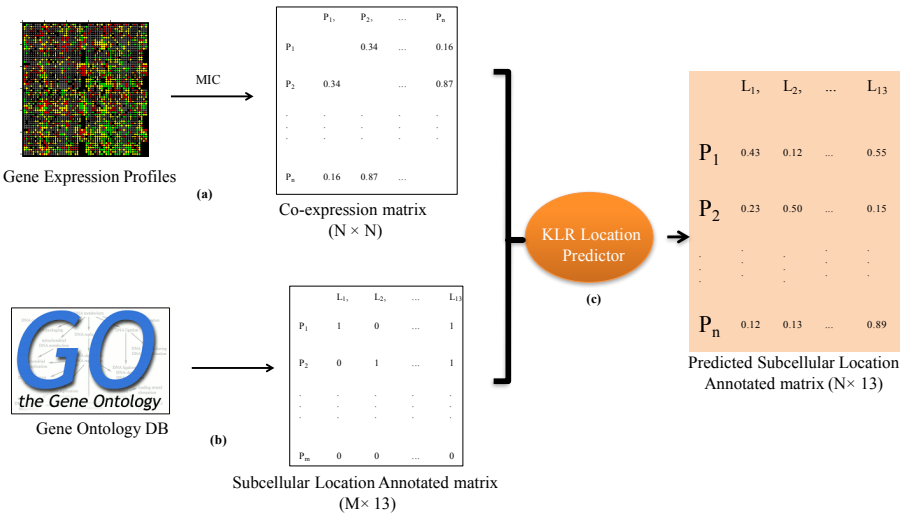


Figure 2.2 Pipeline of KLR model for subcellular localization prediction. (a) Based on expression values, maximal information coefficient was calculated of two proteins. Then we got a co-expression matrix ($N \times N$) which was considered as a condition-based PPI network. (b) In gene ontology database, we extracted subcellular ontology information of any possible protein appeared in gene expression profile to generate the subcellular location annotated matrix ($M \times 13$). (c) KLR predictor took two matrix as input to predict the probabilities on 13 subcellular locations of each protein appeared in gene expression file. The result was a $N \times 13$ subcellular location annotated matrix.

our approach:

1. Generate co-expression matrices using MIC [80] package based on gene expression files for normal and disease states.

2. Use KLR logistic regression model [48] to predict the conditional subcellular locations of all proteins in both states.

3. Compare and rank the probability discrepancy of localizations of all proteins.

4. Filter and mark mislocation-related cancer genes.

2.2.1 Material

Protein-protein Interaction Network

We generate the protein-protein interaction network based on two proteins interaction database: DIP (Released on 2014/04/27)[84] and HPRD (Version number: Relase 9) [39]. DIP provides specific datasets for different species and we downloaded Homo sapiens dataset which includes 3651 proteins and 5534 interactions. There are 11269 proteins and 36398 interactions in HPRD database (Table 2.1). Combining two databases, we constructed a 12516×12516 PPI network matrix in which each element $m_{i,j}$ represents the interaction status between protein i and j:

$$m_{i,j} = \begin{cases} 1 & \text{if protein } i \text{ and } j \text{ have interactions;} \\ 0 & \text{otherwise;} \end{cases} \tag{2.1}$$

Table 2.1 Source of evidence data: protein-protein interactions.

Data source	Proteins	Interactions
DIP	3685	5570
HPRD	11269	39240
Summary	12516	40940

Subcellular Location Annotation Matrix

We obtained subcellular localization annotations of human proteins from GOC website [5]. The datasets contains 44,900 annotated proteins. We then collected the proteins that have at least one of 13 subcellular locations as shown in Table 2.2. We screen the GOC database and 6270 proteins were found appearing in our PPI network previously constructed. Then we have the subcelluar location annotation matrix in

which each row is a protein p's annotation array $a_p[l_1, ..., l_{13}]$ where:

$$a_p[l_i] = \begin{cases} 1 & \text{protein } p \text{ is annotated at subcellular location } l_i; \\ 0 & \text{otherwise;} \end{cases} \tag{2.2}$$

Table 2.2 Subcellular locations.

Gene ontology ID	Location
GO:0005938	Cell cortex
GO:0005829	Cytosol
GO:0015629	Actin cytoskeleton
GO:0005794	Golgi apparatus
GO:0005783	Endoplasmic reticulum
GO:0005773	Vacuole
GO:0005730	Nucleolus
GO:0005777	Peroxisome
GO:0005739	Mitochondrion
GO:0005764	Lysosome
GO:0005813	Centrosome
GO:0005634	Nucleus
GO:0005886	Plasma membrane

Gene Expression Profiles and Co-expression Network

Five types of cancer gene expression profiles and normal human expression profiles were downloaded from the NCBI Gene Expression Omnibus(GEO) Database (Table 2.3). For each of these files, we applied KNN Impute to estimate the missing values. The KNN Impute was downloaded from [97]. All expression profiles were quantile normalized. If multiple probes were mapped to one gene, we use their average values to combine the records. For each dataset we obtained, we split samples into two categories (if applicable): disease samples and normal samples. Then, we generate corresponding co-expression matrices for both normal and disease samples,

respectively. A co-expression score $c_{i,j}$ of two genes i and j is calculated:

$$c_{i,j} = \begin{cases} 0 & MIC(E_i, E_j) < 0.5 \\ 1 & \text{otherwise} \end{cases} \tag{2.3}$$

where E_i and E_j are observed expression values arrays of genes i and j respectively. Her we use Maximal Information Coefficient (MIC) [80] to represent the correlation of two genes' expression levels.

Table 2.3 Gene Expression Profiles of 6 Cancers

Cancer	GEO dataset	Samples
Acute myeloid leukemia	GSE9476, GSE9476	25 (disease), 38 (normal)
Bladder	GSE3167, GSE48276, GSE3239, GSE1595	110 (disease), 21 (normal)
Breast	GSE27567, GSE20437	61 (disease), 49 (normal)
Colorectal	GSE21510	31 (disease), 25 (normal)
Diffuse large B-cell lymphoma	GSE14938, GSE43677	32 (disease), 42 (disease)

2.2.2 KLR Logistic Regression Model

In our previous work, we developed a diffusion kernel-based logistic regression (KLR) model to predict proteins' subcecullar localizations based on the locations of all other proteins within function linkage network. Our results showed that KRL has an outstanding performance on prediction of proteins' subcecullar localizations [63]. This method has the unique advantage of considering the location labels of all related proteins [49]. The diffusion kernel provides means to incorporate all neighbors (rather than direct neighbors) of proteins in the network. It also allows each protein annotated with multiple subcellular locations.

The protein subcellular localization prediction vis KLR model can be formulated as follows. Given a protein-protein interaction network or a gene co-expression network (briefly named network in later description) consists of N proteins $X_1, ..., X_N$ and n of them have location labels. In KLR model, the probability of one protein X_i

without labels locating at a specific subcellular location is given by:

$$P(X_i) = \frac{1}{1 + e^{-\gamma + \delta M_0(X_i) + \eta M_1(X_i)}} \tag{2.4}$$

In equation 2.4, γ, δ and η are parameters to be learned and $M_0(X_i)$ and $M_1(X_i)$ are define as:

$$M_0(X_i) = \sum_{j \neq i \text{ and } X_j = 0} K(X_i, X_j) \tag{2.5}$$

$$M_1(X_i) = \sum_{j \neq i \text{ and } X_j = 1} K(X_i, X_j) \tag{2.6}$$

In $M_0(i)$ and $M_1(i)$, $K(i, j)$ is the kernel function calculating the similarity between two proteins in network and $K(i, j)$ is defined as:

$$K(X_i, X_j) = e^{\tau L(X_i, X_j)} \tag{2.7}$$

where

$$L(X_i, X_j) = \begin{cases} 1 & \text{if protein } X_i \text{ interacts with protein } X_j \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

and τ is the diffusion constant, a hyper parameter. We trained our KLR model with protein-protein network and subcelullar annotation matrix with Maximum Likelihood Estimation. To prediction, the gene co-expression network and subceullar annotation matrix will be used as input.

2.3 Experiments

2.3.1 Case Studies: Discovery of Possible Mislocalized Cancer Related Proteins Across Several Types of Cancers

We test our pipeline on gene expression files from 6 cancers' data: Acute myeloid leukemia, Bladder, Breast, Colorectal and Diffuse large B-cell lymphoma. The gene expression files used are listed in Table 2.3. For each cancer, we filter potential mislocalization-related genes from known cancer genes following below steps:

1. Use proposed pipeline to get predicted probabilities of each protein in 13 subcellular locations for both normal and tumor states. Let $\mathbf{P}_a$ indicates protein a's probabilities on 13 subcecullar locations under normal state and $\mathbf{P}_a[i]$ represents a's probability locating on ith location of 13 subcellular locations. Let $\mathbf{P}_a^{'}$ represents its localization probabilities under disease state.

2. Calculate $\triangle \mathbf{P}_a$ by subtracting $\mathbf{P}_a^{'}$ from $\mathbf{P}_a$. If $\triangle \mathbf{P}_a[i]$ is a relatively large negative value, it indicates that protein a may be missing from location i in disease state. And if $\triangle \mathbf{P}_a[i]$ is a relatively large negative value, it may imply that the protein a may be mislocated to the subcellular location i in disease state.

We reported the proteins with either at least 10% probability change or ranked within top 4 proteins with at least 5% probability change. We observed that in most cases, the cancer genes not selected in our method usually had much lower probability change with a p-value of less than 0.002% for t test scores of all $\triangle P$ on a subcellular location i. So the simple threshold approach adopted here to selecting mislocated proteins is sufficient to distinguish the potential mislocalized cancer genes from other genes. Table 2.4 lists potential candidate genes we filtered.

Acute Myeloid Leukemia

In Acute myeloid leukemia, we used 25 samples in tumor state and 38 samples in normal state as our analysis data. They were all from hematopoietic cells. Our pipeline identified two genes DNMT3A and STAG2 with high localization change that are reported as cancer genes in [45]. DNMT3A, in our model, was predicted mislocating from Golgi apparatus in disease cell (2.96%) comparing with normal cells (6.29%). Another covered candidate gene is STAG2, which was predicted mislocating to two locations: Nucleus where had 41.29% probability in normal and had probability

Table 2.4 Potential mislocalized cancer gene candidates identified by proposed method.

Cancer type	**Mislocalized protein candidates and subcellular places mislocalized to**	**Missing protein candidates and locations missing from**
Acute myeloid leukemia	STAG2 (Nucleus [GO:0005634], Endoplasmic reticulum [GO:0005783])	DNMT3A (Golgi apparatus [GO:0005794])
Bladder	DDX5(Nucleus [GO:0005634]), ELF3 (Mitochondrion [GO:0005739])	DDX5 (Endoplasmic reticulum [GO:0005783]), RB1 (Cell cortex [GO:0005938])
Breast Cancer	ERBB3 (Golgi apparatus [GO:0005794]), RBB2 (Golgi apparatus [GO:0005794])	ERBB3 (Nucleolus [GO:0005730], Cilium [GO:0005929]), ERBB2 (Nucleolus [GO:0005730], Cilium [GO:0005929])
Colorectal	PIK3CA (Cytosol [GO:0005829])	PIK3CA (Golgi apparatus [GO:0005794]), GOT1 (Peroxisome [GO:0005777], Endoplasmic reticulum [GO:0005783])
Diffuse large B-cell lymphoma	CD70 (plasma membrane [GO:0005886]), CREBBP (Golgi apparatus [GO:0005794]), MAP2K1 (Cytosol [GO:0005829])	B2M (Golgi apparatus [GO:0005794]), EZH2 (Centrosome [GO:0005813]), CREBBP (Mitochondrion [GO:0005739]), CD70 (Nucleus [GO:0005634])

64.45% in disease and Endoplasmic reticulum where had 5.52% probability in normal and had probability 12.12%

Bladder Tumor

Our results used 9 normal bladder tissue samples and more than 110 disease bladder tissue samples. The cancer gene DDX5 was predicted to have a 36.19% probability of being localized to Nucleus in normal tissues. The predicted probability of being localized to Neucleus in disease tissues was 52.25% with an increase of more than 40%. At the same time, DDX5 had 8.60% probability in normal and 3.80% in disease localizing on Endoplasmic reticulum with an decrease of more than 55%. So based on this probabilities changes, we can make a speculation that in bladder tumor cell DDX5 mislocalized from Endoplasmic reticulum to neucleus. Tumor suppressor gene RB1 was predicted with a high probability change in its subcellular location from cell cortex to other locations (3.39% in normal, 0.98% in disease). Recently a study [8] showed that in breast cancer mutated RB1 will lead to their localization to nucleus. We also observed in our prediction the increase of probability localization in nucleus (48.23% in normal, 57.27% in disease). Our predicted localization change of RB1 maybe caused by its mutation. Also protein ELF3 was predicted mislocating to Mitochondrion where its location probability changing from 8.16% in normal state to 15.31% in cancer state.

Breast Cancer

After analyzing 61 disease samples and 49 normal samples, We obtained several candidate proteins that probably mislocate under cancer states. At subcellular location Cilium, protein ERBB2's predicted localization probability decreased from 63.68% (normal) to 55.87% (disease). Also its probability localization on Nucleolus had a decreasing from 6.41% (normal) to 1.99% (disease). And at the same time, ERBB2's probability localization on Golgi apparatus had an increasing from 8.65% (normal) to 11.37% (disease). These observations could indicate that some ERBB2 proteins in breast cancer cells mislocalized from Cilium and Nucleolus to Golgi apparatus. Pro-

tein ERBB3 had similar mislocation activity from Cilium (52.77% in normal, 44.78% in disease) and Nucleolus (9.60% in normal, 3.05% in disease) to Golgi apparatus (8.21% in normal, 10.75% in disease).

Colorectal Tumor

For colorectal tumor, we had 31 tumor samples and 25 normal samples. Protein PIK3CA was observed in our prediction mislocating from Golgi apparatus (11.17% in normal, 4.81% in disease) to Cytosol (27.74% in normal, 45.84% in disease). Protein GOT1 was predicted mislocating from Peroxisome where its location probability decreased from 5.09% to 0.84% and from Endoplasmic reticulum where its location probability decreased from 10.95% to 3.99%.

Diffuse Large B-cell Lymphoma

In Diffuse large B-cell lymphoma test, we used 42 normal samples and 32 disease samples. Protein CD70 was predicted missing from Nucleus (59.45% in normal, 35.96% in disease) and at the same time its probability in Plasma membrane increased from 6.41% to 39.84%. So we speculated that some CD70 proteins may mislocate to Plasma membrane from Nucleus in cancer cells. Protein CREBBP was also predicted having mislocation activity that from Mitochondrion to Golgi apparatus. Its predicted probability in Mitochondrion decreased from 25.64% in normal to 6.26% in disease and the probability in Golgi apparatus increased from 3.95% to 7.62%. Besides, protein MAP2K1 was observed mislocating to Cytosol (38.22% in normal, 66.17% in disease). Protein B2M and protein EZH2 were predicted mislocating from Golgi apparatus (12.90% in normal, 5.90% in disease) and Centrosome (23.92% in normal, 6.90% in disease), respectively.

2.3.2 Rediscovery of Mislocalized Proteins in Glioma

In the work of Lee [51], a proteome-wide method was proposed to predict mislocalized proteins under glioma tumor. They used sequence, chemical properties, motifs, and function information of proteins as the basic features. Condition-dependent dynamic network features were generated by assigning different weights to each neighbor of a protein, depending on their similarity in gene expression profiles. Based on the basic features and dynamic network features, they computed a CLM score for each protein, listing the quantitative probability that the protein is located in each location under each condition. Mislocations are identified by calculating differences in degrees of probabilities across conditions. Their prediction result showed that their model can successfully identify potential mislocation proteins under tumor conditions. However, their approach requires many kinds of data resources which might not be available for some proteins. Since the condition-dependent information is only contained in the gene expression data, we wanted to test our model on the same gene expression data to see if we can rediscover mislocalized proteins under glioma. We used GEO dataset used by Lee's team which contained normal, low and high states. In order to find possible mislocalized proteins, firstly, our pipeline was used to predict location probabilities of 13 subcellular locations of each protein for each status. For each protein, we selected the location with highest probability as its predicted location in that status. After we obtained a protein's location probabilities in all normal and disease states, we just picked up proteins having different locations under normal and disease states. Totally, we found 230 proteins predicted as mislocalized proteins among 11500 proteins. And for 157 proteins predicted as potentially mislocalized under glioma in Lee's paper, we rediscovered 31 proteins of them in our 230 predicted candidates having same mislocation activity. Among rediscovered 31 proteins, TBX19 was experimentally verified mislocation from endoplasmic reticulum to nucleus within glioma cell in Lee's paper. 31 proteins predicted by both Lee's and our work with same misloca-

tion activities: PSPN, TMEM132A, ARF5, TIA1, TIMM8A, OBP2A, CIC, OLFM2, CPSF3L, DUSP14, PPM1B, SEC13, SSBP3, PEX13, THPO, RCAN2, TTN, WNT6, TBX19, DAP, GEM, ATIC, DCPS, PROC, PAX1, SSB, TYMS, CCDC116, FTL, CBLL1 and PLK3.

2.4 Chapter Summary

In this work, we have developed an approach for discovering mislocation related cancer genes based on aberrant gene expression data and diffusion kernel based logistic regression for subcellular localization prediction. Our approach is complementary to high-throughput genomic sequencing approaches for cancer gene detection by providing the means for understanding the pathological mechanisms. The experiments showed that our approach has identified several interesting cancer genes reported by genomic study, by means of which the cancer may be related to their mislocalization within the cell.

Chapter 3

Peptide-HLA class I Binding Prediction With Allele specific CNN model

3.1 Introduction

Major Histocompatibility Complex (MHC) proteins located on cell surface play a critical role helping immune system recognizing pathogens. MHC proteins bind to peptide fragments derived from pathogens and display them on the cell surface for recognition by the appropriate T cells [57]. There are two major types of MHC (named human leukocyte antigen (HLA) for human): class I and class II. HLAs class I present peptides from inside the cell while HLAs class II present antigens from outside of the cell to T-lymphocytes [103]. Peptides binding to HLA-I proteins will be recognized by Cytotoxic T lymphocytes, which is a fundamental activity of immune system. As shown in 3.2, endogenous antigens are first cleaved into peptide fragments by the proteasome, which are then generally translocated by the transporter associated with antigen processing (TAP) into the endoplasmic reticulum (ER). Then, HLA-I molecules bind certain peptides and present them to cytotoxic T lymphocytes (CTL)-stimulating cellular immunity [110]. The schematic structure of HLA class I is shown in Figure 3.1. As shown in the figure, HLA class I molecules are heterodimers contains two polypeptide chains: α and β2-microglobulin. Only the α chain is polymorphic and encoded by a HLA gene. The α_1 and α_2 domains hold to make up a groove for peptides binding while the α_3 domain interacts with the CD8 co-receptor of T-cells [103].

The genomic regions encoding HLA class I are highly diverse. As of Aug. 2017, IPD-IMGT/HLA database [82] contains more than 12,000 HLA class I alleles. However, binding specificities of most HLA-I alleles have not been experimentally characterized because it costs immense amount of financing and time. Thus, a large number of computation methods were proposed on peptide-HLA Class I binding prediction [38, 108, 12, 71, 76, 26, 40]. Generally, current computational methods for peptide-HLA class I binding affinity prediction can be grouped into two categories: allele-specific and pan-specific models. Allele-specific models are trained with only

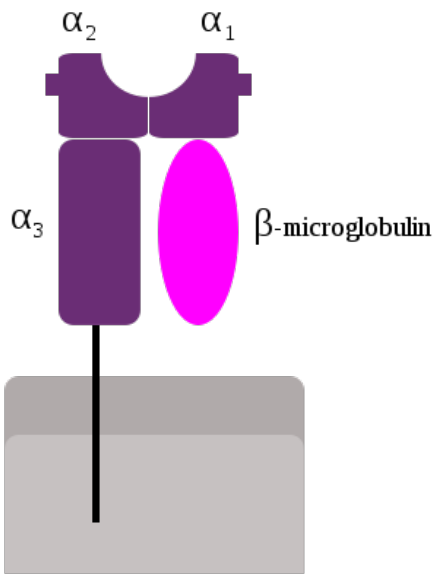


Figure 3.1 Schematic representation of HLA class I. Reproduced from wikipedia

the binding peptides tested on a specific allele and a separate allele-specific binding affinity prediction model is needed for each HLA allele. Allele-specific models have the advantage of good performance when sufficient number of training peptide samples are available. In pan-specific models, binding peptides of different alleles are all combined to train a single prediction model for all HLA alleles. Typically, a pan-specific model uses binding affinity data from multiple alleles for training and could predict peptide binding affinity for the alleles that may have or have not appeared in the training data. In Figure 4.1, we show the difference between allele-specific methods and pan-specific methods.

In this chapter, we propose an allele-specific convolutional neural network model (DeepMHC) targeting peptide-HLA class I binding affinity prediction and we show that our method obtains better performance than all existing methods. First, we review state-of-the-art methods in Section 3.2. Then we introduce our proposed allele-specific model in Section 3.3. At the end of this chapter, Section 3.4, we compare our performance with existing methods and do analyzing of our results and proposed

method.

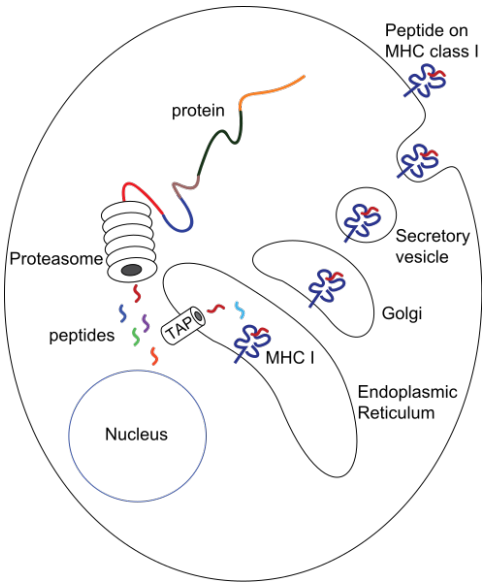


Figure 3.2 Simplified diagram of HLA class I pathway. Reproduced from wikipedia.org

3.2 Related work

Existing prediction models for peptide-MHC I binding prediction can be classified into two major categories: allele-specific model and pan-specific model. Allele-specific prediction models are trained with binding data to a single allele and the trained model can only be used to predict peptides binding to this allele. NetMHC [58] and SMM [76] are representative ones with competitive performance. In this chapter, we propose an allele-specific method based on convolutional neural network (CNN). We compare its performance with both allele-specific and pan-specific methods. Pan-specific methods take binding data of multiple alleles in a same *supertype* [87] and MHC contacting environment data to train a model and the trained model could be used to predict on not only input alleles but also alleles with few or even no known binders [110]. Currently, outstanding pan-specific methods for MHC-I include

NetMHCPan [26] and PickPocket [108]. In next chapter, we'll introduce a pan-specific method.

3.2.1 Deep Neural Network Methods

Currently, we just find few publications involving with deep neural network:

HONN [42] applied a Gaussian restricted Boltzmann machine(RBM) based deep neural network (DNN) and a semi-RBMs based feed-forward high-order neural network(HONN) to peptide-MHC I binding prediction. A BLOSUM encoding is used to encode peptides into 180-dimensional continuous vectors.

HLA-CNN [99] is a newly reported model applied convolutional neural network on this problem. It used distributed representation encoding a amino acids peptide into a 15-dimensional vector. Then encoded vectors are fed into a network consisting of two convolutional layers and one fully connected layer.

JHU-CNN proposed a CNN model using one-hot encoding[10]. Their model uses different size of 1D convolutional filters on input sequences and concatenates convolutional layers' output into a single tensor and then feeds this concatenated tensor into a multiple-layer fully connected neural network.

Comparison with these methods are described in section 3.4.3.

3.3 Methodology

3.3.1 Peptide sequence encoding

We use one-hot encoding approach to transform a amino acid sequence into a tensor representation. As shown in Figure 3.3, each amino acid of the peptide is encoded by a sparse column vector of dimension 20 with corresponding component set to 1 and remaining 19 components set as 0. With the one-hot representation, there are still two different ways to map the encoded matrix to a tensor in implementation level. One is to map the input matrix as a 20-channel 1-row 2D tensor (or 20-channel 1D

tensor), as is done in our proposed approach. Another way is to map the input matrix as a single channel 20-row 2D tensor. In section 3.4.5, we compare and analyze two different encoding ways' performance.

	PAD	PAD	A	L	T	L	S	P	Y	Y	K	PAD	PAD
A	0	0	1	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0	0	0
.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.
T	0	0	0	0	1	0	0	0	0	0	0	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0
Y	0	0	0	0	0	0	0	0	1	1	0	0	0

Figure 3.3 One-hot encoding example of protein sequence ALTLSPYYK.

3.3.2 Convolutional Neural Network

We proposed a CNN model composed of two stacked convolutional layers, one max-pooling layer, and one fully connected layer as shown in Figure 3.4. Of two convolution layers, both has 512 convolution filters of size 1×2 and 1×3 respectively. Adam [41] is used as the optimization algorithm for training the networks.

The convolutional layers: For each convolutional layer, it has 512 filters. These filters will scan the peptide sequence from left to the right on 20 channels. We use $\mathbf{S}$ to denote the encoded single row, 20-channel 2D tensor from a peptide sequence. It shape will be $(1, 13, 20)$ where 13 is the padded encoding length for a peptide sequence with 9 or 10 length. $\mathbf{F}$ represent one of the 512 convolution filter tensors with shape $(1, 2, 20)$ which indicates that every filter has a 1×2 receptive field for each channel. Then the output tensor $\mathbf{D}$ with a shape $(1, 12)$ from a convolution operation between $\mathbf{S} \bigodot \mathbf{F}$ is calculated from:

$$D_{i,j} = \sum_{c=1}^{20} \sum_{m=i,p=1}^{1} \sum_{n=j,q=1}^{2} S_{m,n,c} K_{p,q,c} \tag{3.1}$$

Max-pooling layers: The pooling layers are used for summarizing the activations of adjacent neurons. Different from the convolution layers in which the convolution filters move along the sequence by stride size of 1 with overlapping, the nonoverlapping pooling is applied with a stride size $(1, N)$ to reduce the dimension of the input sequence and thus the number of model parameters. Two commonly used poolings are Average-Pooling and Max-Pooling. Here the max-pooling layers are used in our CNN models for summarizing the activations of adjacent N neurons by their maximum value. In our model, we chose 2 as the N value. For a input tensor A with shape $(1, L)$, the output tensor P with shape $(1, L/N)$ is calculated from:

$$P_{i,j} = max_{m \in [i,i+N], n \in [j,j+N]}(A_{m,n}) \tag{3.2}$$

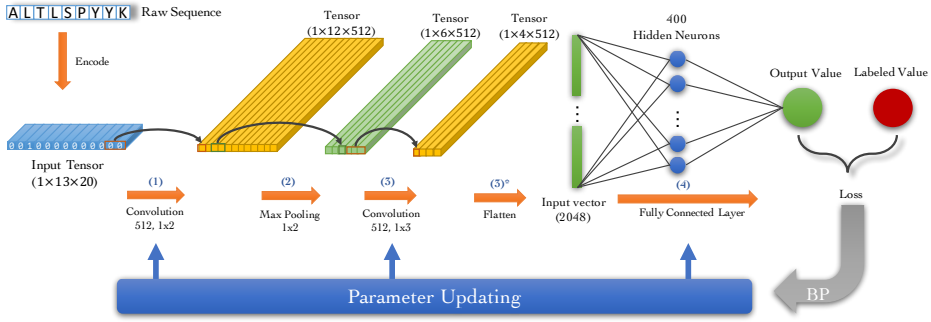


Figure 3.4 Network Architecture of Proposed CNN Model.

3.3.3 Datasets and Evaluation Metrics

To ensure fair comparison with existing approaches, we used the IEDB training and test datasets released on the IEDB website. These datasets can be found at http://tools.iedb.org/main/datasets/. We trained on MHC-I alleles with at least about 2000 training samples from BD2013. The details of training datasets are listed in Table 3.1. The evaluation dataset were downloaded from IEDB's weekly benchmark dataset [96]. We download all dataset from 2014-03-21 to 2016-12-09 combining

test samples according to alleles, sequence lengths and measure types. We separate the evaluation datasets into two groups: one with affinity scores: IC_{50} and t1/2 (Table 3.3) and another one with binary affinity labels (Table 3.2).

Metrics we use to evaluate the performance are: area under the receiver operating curve (AUC), the Spearman rank correlation coefficient (SRCC) and Pearson correlation coefficient (Pearson).

3.3.4 Allele-specific Training and Evaluation Protocol

The network structure is shown in Figure 3.4 and we use Mean Squared Error (MSE) as the loss function. In the output layer, sigmoid activation is used to give the output value. The labeled values in BD2013 are measured in IC_{50} which are distributed in a huge real value space ($0.0 \backsim 80000.0$). Considering this, we convert IC_{50} to pIC_{50} via equation 3.3 and we use pIC_{50} values as training labels in our experiments.

$$pIC_{50} = -log_{10}(IC_{50}) \tag{3.3}$$

Training. First, we configure a CNN model described in section 3.3.2 and train this model running up to 5000 epochs. Before each training epoch, training samples are randomly split into training and validation sets following 9:1 ratio. The training stops if the minimum validation loss haven't reduced for 100 epochs (known as early stop) or reaches 5000 epochs.

Cross-validation Evaluation. For each allele's training data, we split them into 5 folds. For sequences in each fold held out as the test set, we applied the trained CNN model to predict their affinity values. This process is repeated 5 times for all 5 folds. And then the Pearson correlation coefficients of all peptide samples of a MHC-I allele are recorded.

Benchmark Evaluation. For each peptide sequence in the benchmark dataset, we applied the 5 trained cross-validation models of peptide's targeting allele and predict 5 affinity values for one peptide sequence. And then the average of these

predicted values from all 5 models is set as the final predicted pIC_{50} for the peptide sequence. Finally, the predicted pIC_{50} values are converted back into IC_{50} values based on Equation 3.3.

3.4 Experiments

3.4.1 Performance of DeepMHC on MHC-I Binding Affinity Prediction

Cross-validation Results

The 5-fold cross-validation performances of the CNN models on all MHC-I alleles are listed in Table 3.1. The Pearson scores range from 0.48 to 0.92. Our CNN models achieved Pearson scores of more than 0.8 for 27 out of the 37 alleles. We also found that most of the low-performance scores are from the alleles with fewer number of peptide sequences (< 1000), which indicates that sufficient number of peptide sequences are required for our CNN models to achieve good performance. It is also found that all these low-performance allele dataset consist of peptide sequences of length 10.

Benchmark Testing Results

Results of benchmark dataset are showed in Table 3.4 for peptides of 9-length and Table 3.5 for peptides of 10-length. We sorted results by the combination of alleles, sequence lengths (9 or 10) and measure types (IC_{50} or t1/2). Methods in these two tables are: NetMHCpan [26], SMM [76], ANN [71], ARB [12], NetMHCcons [38], SMMPMBEC [40], IEDBconsensus [64] and PickPocket [108].

Table 3.4 shows the comparison results on 9-length sequences. Few methods listed in IEDB weekly benchmark dataset are omitted in this table because it doest not have either any best results or complete prediction results. From Table 3.4, we can safely say that our method performs overall better than others for obtaining 6 highest AUC

scores, 8 highest SRCC scores and 7 highest Pearson scores of 15 testing entries. And in only 6 test entries, our model didn't obtain any best results among three metrics. Also the average scores of our methods are still the best which indicating that our proposed convolution network model has a stable prediction capability and could fit in different alleles. The state-of-the-art NetMHCpan utilized pan-specific strategy and trained each alleles's samples on a set of artificial neural networks with 22 to 86 hidden neurons with 3 types of input encoding ways and then pick the best 15 networks [26]. Comparing with NetMHCpan, our allele-specific method fixed a neural network structure and encoded sequences in the same way but our model performed better more than half of the all test cases comparing AUC scores. Of 4 test entries where our model didn't get best results on AUC scores, HLA-A*02:02 (IC_{50}), HLA-A*24:02 (IC_{50}) and HLA-B*07:02 (IC_{50} and t1/2), our performance were still either very close to best results or among the top 3. For alleles HLA-A*02:03 (IC_{50}), HLA-B*58:01 (IC_{50}), HLA-B*68:01 (IC_{50} and t1/2), HLA-B*57:01(IC_{50}), we analyze the prediction results and give following possible reasons our model could not obtain better performance.

- **Sample size.** In the Table 3.1, HLA-B*68:01 just has 2036 training samples of 9 length sequences. It is well known that in order to obtain a good prediction capability for deep neural network, the training sample size should be large enough letting the model to do the optimization for its tons of parameters. So the trained model may not be fitted well on this kind of cases.

- **Misleading testing data.** In the training dataset BD2013, IEDB combines KD (thermo-dynamic constant), IC_{50} and EC50 together representing as IC_{50} since IC_{50} and EC50 measurements can approximate KD. And we found in weekly benchmark data, there are some sequences appeared in training data but contradict to the training labels dues to different measuring techniques. For

example, the sequence ELAAHQKKI targeting to allele HLA-A*02:03. In the training data, it is labeled with value 2650 (radioactivity dissociation constant KD). In the weekly testing data, it is labeled with value 1.0 (radioactivity half maximal inhibitory concentratio IC_{50}). When we were testing, our method and all other benchmark methods significantly failed on predicting this sequence's IC_{50} value. Since we followed the same method in [96] marking sequence as negative if its $IC_{50} < 500.0$ nm, this kind of samples would drag the AUC scores down for all models. In test entry HLA-A*02:03 (IC_{50}), our model gave a highest related prediction with a 0.62 Pearson score and we found among 105 testing sequences, there are 14 sequences appeared in training data with contradicting labels. This maybe a big reason we failed on this kind of cases.

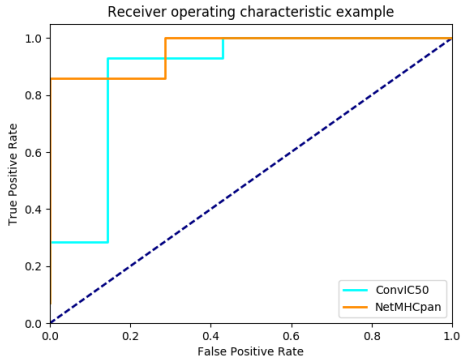
Table 3.5 shows performance comparison of benchmark dataset on 10-length sequences. From this table we can find that our model performs much better than all other comparing methods with 4/5 best AUC scores, 5/5 best SRCC scores and 2/5 best Pearson scores. And in test entries HLA-A*02-01(t1/2), HLA-A*02-03(IC_{50}) and HLA- A*68-02(IC_{50}) we have a pretty large leading margin. Also, the alleles in Table 3.5 where we almost obtained all the best results also are the same alleles in Table 3.4 where we obtained some good results. This indicated that our model could capture the basic feature representation across in different length sequences.

3.4.2 Performance of DeepMHC on MHC-I Binding Prediction

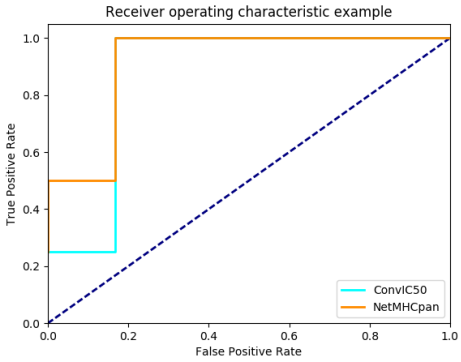
To evaluate how DeepMHC performs on binary MHC-peptide binding prediction, we trained a set of convolution neural network models over the MHC allele peptide datasets of length of 9 with binary binding labels. The architectures of these models are exactly as described in previous section except that the binary entropy loss function instead of MSE is used and the outputs are binary labels instead of real-value binding affinity. We then evaluate their prediction performance on the external test

datasets in Table 3.2. The prediction performances for all the datasets are shown in Table 3.6.

First, we find that our DeepMHC models based on raw amino acid sequences are competitive compared to other algorithms that depend on sophisticated feature engineering. Out of 10 allele datasets, DeepMHC achieves top performance on 4 datasets, which is better than NetMHCpan, SMM and ANN methods. Out of the remaining 6 datasets, DeepMHC obtains competitive results on 4 datasets (HLA-A*02:01, HLA-A*68:01,HLA-B*07:02, HLA-A*31:01) with AUCs differences of less than or equal to 0.04 compared to the top scores. But for cases like HLA-B*27:05 and HLA-A*24:02, our performance is much worse, which can be partially attributed to the reasons as mentioned in Section 3.4.1. To further explore the reasons, we plotted the ROC curves of DeepMHC compared to NetMHCpan on these two datasets in Figure 3.5. It is found that our models do not perform as well as NetMHCpan does when the false positive rate is less than 0.2.



(a) HLA-B*27:05



(b) HLA-A*24:02

Figure 3.5 ROC curves of prediction results on benchmark data with binary labels of allele HLA-B*27:05 and HLA-A*24:02.

3.4.3 Comparison with Other DNN and CNN based approaches

HONN

Since there's no available code for HONN, we trained another set of allele-specific models on dataset BD2009 and tested these models on dataset BLIND. BD2009 and BLIND were described in the HONN paper to train and evaluate the HONN model. Table 3.7 shows the performances comparison between DeepMHC and HONN. The AUC scores of HONN were directly obtained from its supporting material. From the table we can say that on dataset BLIND, our model and HONN have a very similar performance. Of all 29 test entries, on 15 test entries both model have same AUC scores. In the rest entries, our model has 6 better results and HONN has 8.

JHU-CNN

JHU-CNN gives trained models and codes, we tested theirs models on the same weekly benchmark dataset on all available alleles. The results are showed in Table 3.8. From this table, of all 20 testing entries, we obtained 12 better AUC scores, 13 better SRCC scores and 10 better PRCC scores. This shows that our method is a better modeled structure for peptide-MHC I binding problem.

HLA-CNN

HLA-CNN tested their model on recent weekly benchmark data from IEDB. We extract our prediction results for those testing sequences and compared the AUC and SRCC. Results are shown in Table 3.9. The table displays a trend that HLA-CNN performed better on AUC metric with 7 better AUC scores of 9 testing entries and our model is better on SRCC metric with 7 better SRCC scores. A possible reason is that HLA-CNN uses binary cross entropy as the loss function for their training. Some studies indicated that cross entropy could improve performance of classification [47].

3.4.4 Transfer Learning on MHC Alleles with Small Number of Samples

A major issue in MHC-I binding prediction is that some alleles have small-size sample sets. This makes it challenging to train accurate allele-specific prediction models. And in deep learning, it is well known that a relative large dataset is critical for better performance. An effective approach to address this issue is to use transfer learning, which improves a learner from one domain by transferring information from a related domain [13]. In this experiment, We would like to explore if transfer learning can improve the prediction performance of DeepMHC on alleles with small number of training samples. To verify this approach, we picked two alleles: HLA-A*02:01 and HLA-A*02:02. We chose these two alleles because: i) According to [88], most HLA-A alleles can be clustered in to 6 supertypes. HLA-A*02:01 and HLA-A*02:02 are clustered into same supertype A02. So we can assume that HLA-A*02:01 and HLA-A*02:02 share similar preference for binding peptide sequences to a certain degree. ii) The HLA-A*02:01 dataset has the most training samples (9051) and HLA-A*02:02 has 2465 training samples. This is an idea situation to verify transfer learning's effectiveness.

We trained 5 models for HLA-A*02:02 following the same procedure described before in a 5-fold cross validation way except that before we start to train a model, we initialized the model's weights of its first convolutional layer as the same weights of first convolutional layer from one of the 5 pretrained models of allele HLA-A*02:01. Figure 3.6 shows the performance comparison of affinity benchmark dataset of allele HLA-A*02:02 on testing on models with and without transfer learning. We found that the model trained with transfer learning has achieved a much better prediction performance. The AUC has increased to 0.79 from 0.73; the SRCC reaches 0.72 from 0.66; and the Pearson improves from -0.03 to 0.01. If we compare these improved metrics of allele HLA-A*02:02 with other methods' reported in Table 3.4, this transfer learning based model achieved the best results on test dataset HLA-A*02:02(IC_{50}).

Such significant performance improvement by transfer learning may be partially due to the fact that we were actually doing a *good* initialization of weights for the first convolutional layers by reusing the weights from the pretrained CNN models of HLA-A*02:01. It is increasingly recongized that proper initiallization of the weights of the convolution layers is critical for the CNN model to get better prediction performance [62, 28].

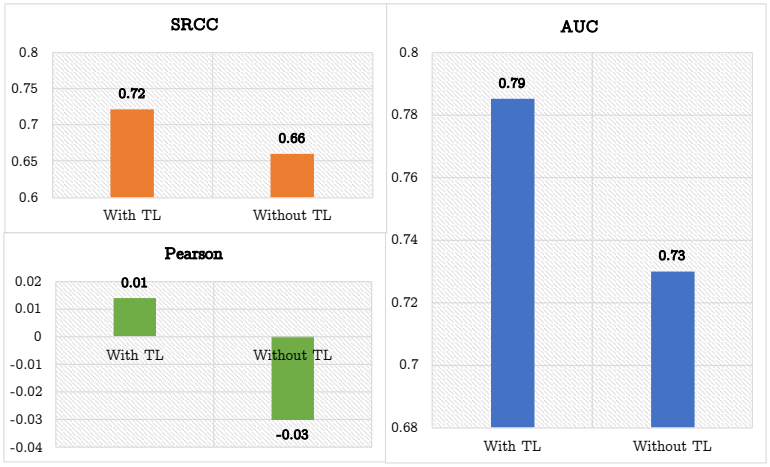


Figure 3.6 Performance comparison of benchmark dataset of allele HLA-A*02:02 testing on model without transfer learning (Without TL) and model with transfer learning (With TL).

3.4.5 Comparison of Protein Sequence Encoding Schemas for One-Hot Encoding

Unlike images naturally encoded as 2D multi-channel matrices, amino acid sequences require us to chose proper encoding method to transfer them into tensors for deep neural networks. In this section, we explore whether different encoding methods will effect the prediction performance for peptide-MHC I binding prediction.

In previous experiments, we adopted a single row 20-channel encoding schema. Here we are evaluating a single-channel 20-row encoding schema and compared its

performance to 20-channel encoding schema. For first encoding schema, a protein sequences is encoded into a tensor object with width of 13, height of 1 and depth (channel) of 20. In the single-channel encoding, the amino acids sequence is represented as a tensor with width of 13, height of 20 and depth of 1. For single-channel encoding schema, we constructed a compatible CNN model for the input tensors. As illustrated in Figure 3.7, we first encode a 9-length sequence into a 20 $\times$ 9 $\times$ 1 dimension input tensor. Each row encoded the corresponding amino acid position. Then the input tensor was feed to (1) convolution layer which has 512 filters with 2 $\times$ 2 receptive field size. Next in layer (2), output tensor of (1) is applied by a Max Pooling layer with a 2 $\times$ 2 window. After Max Pooling, another convolution layer (3) with 252 filters of 2 $\times$ 2 receptive field size is applied to the pooling layer output. Then do pooling operation. A flatten operation is applied on the layer (4)'s output and a 1D vector with 1008 dimension is obtained. This vector is feed to the fully connected layer (4) which has 200 hidden units.

We configured a model as described above and following same training and evaluation procedure in section 3.3.4 for allele HLA-A*02:01. We tested the trained model with 20-row encoding schema (2D matrix encoding) on benchmark dataset of allele HLA-A*02:01 and compare it with our previous model's performance. The results are showed in Figure 3.8. As shown in the Figure, the 2D matrix encoding model performs worse in all three metrics comparing with previous 20-channel encoding model. A possible explanation is that when we encode a sequence into a 2D matrix, we manually fixed the spatial relationships of 20 types amino acids. When a convolution filter is searching the frequent patterns in this 2D matrix, it will be influenced by this pre-fixed spatial arrangement. In images, the spatial arrangement of pixel values are natrual and meaningful. However, in our 2D matrix encoding, we had to chose 1 of C_{20}^{20} possible arrangements of 20 amino acids. If we initialize convolutional filters with same strategy but with different arrangements of 20 types amino acids in en-

coding matrix, the final filters learned may be dramatically different. By contrasting, in the 20-channel model, a convolutional filter will have 20 receptive fields for each channel and each field is search the feature patten in its channel independently. So if we initialize convolutional filters with same strategy, no matter how we arrange the relative order of 20 amino acids in terms of channels, the final filters learned should be always similar. That is to say, 20-channel encoding schema model is more stable during backpropagation.

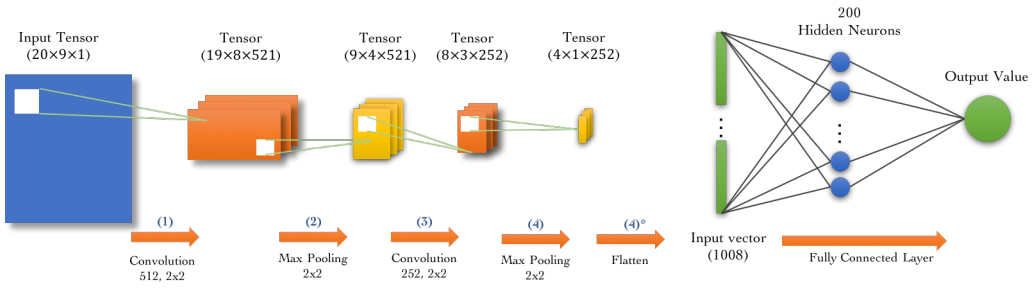


Figure 3.7 Comparison results on benchmark dataset for allele HLA-A*02:01 on models with different encoding method.

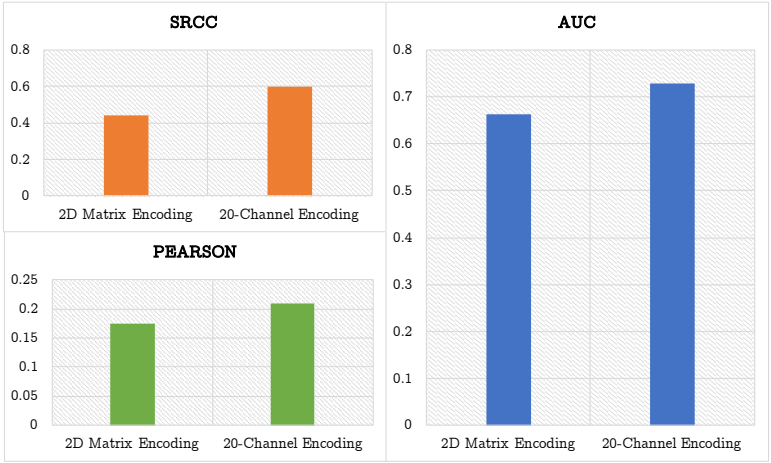


Figure 3.8 Network Structure of Single-Channel 20-Row Encoding ethod.

3.4.6 Exploring Network Structure's Effect on Peptide-MHC I Binding Prediction Performance

Before we fixing our final structure of CNN model in Figure 3.4, we did some exploring of different network structures with variant depth and number of convolution filters. Our experiments showed that for our problem a wider network performs much better than a narrow network and the depth does not influence performance very much.

First, we evaluated how the number of filters of two convolutional layers affects the prediction performance. We configured a set of CNN models as described in Figure 3.4 but varied the number of filters of two convolution layers by setting it as 25, 50, 100, 250 and 512. Then we trained and evaluated these models on allele HLA-A*02:01 following same procedure described in section 3.3.4. The results are showed in Figure 3.9. From the figure we can see that, the model with 512 filters performs better than other models with large leading margins on all metrics: SRCC, Pearson and AUC. In the convolution layer, the number of filters decides the diverse and amount of features we want to capture. Each filter is supposed to capture a basic feature representation of input tensors. The results implies that a wider model is much more helpful improving performance for our targeting problem.

Then, we explored if a deeper network could improve the performance. The model shown in Figure 3.4 consisting of 4 layers: conv-pool-conv-full. We constructed another two types of model:

- 6-layer model: conv-pool-conv-pool-conv-full
- 7-layer model: conv-pool-conv-pool-conv-pool-conv-full

We trained and evaluated 6-layer model and 7-layer model with same procedure on allele HLA-A*02:01. Results are shown in Figure 3.10 and no significant differences were observed. A possible reason is that the dimension of our input tensor is $1\times13\times20$. After too many layers, the intermediate tensors will be very short in width. So after

several layers, it may lost too much information and could not improve performance much.

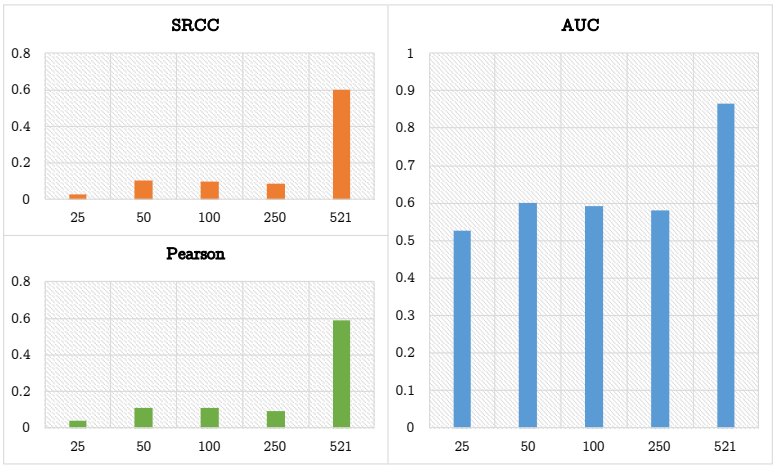


Figure 3.9 Comparison of performance of models with different number of convolutional filters on allele HLA-A*02:01.

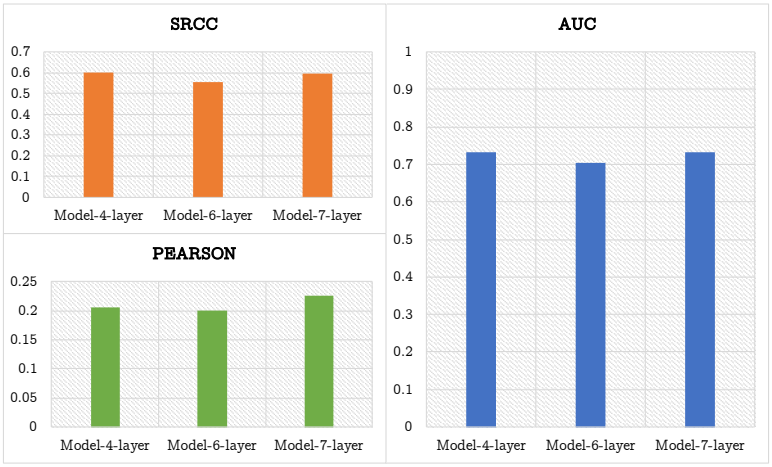


Figure 3.10 Comparison of performance of models with different depth on allele HLA-A*02:01.

3.5 Chapter Summary

In this chapter, we proposed an allele-specific CNN model on peptide-MHC I binding prediction and showed its leading performance comparing with existing methods. We also analyzed few factors that may affect performance of our model. We found that a wider neural network is good for our targeting problem and argued that 20-channel encoding model is more stable comparing with 20-row encoding model.

Table 3.1 Training Datasets from IEDB

MHC	Data Type	Length	Seq Count	Pos	Neg	5-fold CV Pearson
HLA-A*02:01	IC_{50}	9	9051	3273	5778	0.91
HLA-A*03:01	IC_{50}	9	5488	1378	4110	0.87
HLA-A*11:01	IC_{50}	9	4544	1363	3181	0.89
HLA-A*02:03	IC_{50}	9	4428	1543	2885	0.90
HLA-B*15:01	IC_{50}	9	4101	1187	2914	0.86
HLA-A*31:01	IC_{50}	9	3945	1015	2930	0.89
HLA-A*01:01	IC_{50}	9	3902	539	3363	0.81
HLA-B*07:02	IC_{50}	9	3868	1061	2807	0.80
HLA-A*26:01	IC_{50}	9	3766	409	3357	0.87
HLA-A*02:06	IC_{50}	9	3733	1522	2211	0.88
HLA-A*68:02	IC_{50}	9	3672	835	2837	0.90
HLA-B*08:01	IC_{50}	9	3027	715	2312	0.86
HLA-B*58:01	IC_{50}	9	2984	655	2329	0.88
HLA-B*40:01	IC_{50}	9	2824	478	2346	0.89
HLA-B*27:05	IC_{50}	9	2811	443	2368	0.92
HLA-A*30:01	IC_{50}	9	2565	736	1829	0.89
HLA-A*69:01	IC_{50}	9	2558	248	2310	0.82
HLA-B*57:01	IC_{50}	9	2529	384	2145	0.84
HLA-B*35:01	IC_{50}	9	2514	821	1693	0.92
HLA-A*02:02	IC_{50}	9	2465	1119	1346	0.90
HLA-A*24:02	IC_{50}	9	2395	496	1899	0.89
HLA-B*18:01	IC_{50}	9	2315	230	2085	0.81
HLA-B*51:01	IC_{50}	9	2239	233	2006	0.82
HLA-A*29:02	IC_{50}	9	2110	532	1578	0.86
HLA-A*68:01	IC_{50}	9	2036	830	1206	0.90
HLA-A*33:01	IC_{50}	9	1929	387	1542	0.82
HLA-A*23:01	IC_{50}	9	1915	410	1505	0.86
HLA-A*02:01	IC_{50}	10	2753	1150	1603	0.79
HLA-A*03:01	IC_{50}	10	1694	720	974	0.67
HLA-A*11:01	IC_{50}	10	1680	755	925	0.79
HLA-A*68:01	IC_{50}	10	1651	740	911	0.73
HLA-A*31:01	IC_{50}	10	1640	576	1064	0.65
HLA-A*02:06	IC_{50}	10	1623	679	944	0.52
HLA-A*68:02	IC_{50}	10	1617	481	1136	0.68
HLA-A*02:03	IC_{50}	10	1614	811	803	0.51
HLA-A*33:01	IC_{50}	10	1560	315	1245	0.48
HLA-A*02:02	IC_{50}	10	1445	658	787	0.73

Table 3.2 Evaluation datasets with binary affinity labels

IEDB reference	MHC allele	Measure type	Peptide length	Count	Positive count	Negative count
1027079, 1026941, 1027588, 1028928, 1029824, 1026840	HLA-A*02-01	binary	9	491	165	326
1026891, 1026840	HLA-A*24-02	binary	9	378	65	313
1026840	HLA-A*30-01	binary	9	349	8	341
315312	HLA-A*31-01	binary	9	10	4	6
1026840	HLA-A*68-01	binary	9	436	43	393
1028928, 1026840	HLA-B*07-02	binary	9	308	35	273
1029125	HLA-B*27-05	binary	9	21	14	7
1026897, 1026891	HLA-B*40-01	binary	9	38	14	24
1026897, 1026891, 1026840	HLA-B*58-01	binary	9	482	63	419
1026891	HLA-A*11:01	binary	9	22	19	3

Table 3.3 Evaluation datasets with binary affinity labels

IEDB reference	MHC allele	Measure type	Peptide length	Count	Positive count	Negative count
1027079, 1026941, 1027588, 1028928, 1029824, 1026840	HLA-A*02-01	binary	9	491	165	326
1026891, 1026840	HLA-A*24-02	binary	9	378	65	313
1026840	HLA-A*30-01	binary	9	349	8	341
315312	HLA-A*31-01	binary	9	10	4	6
1026840	HLA-A*68-01	binary	9	436	43	393
1028928, 1026840	HLA-B*07-02	binary	9	308	35	273
1029125	HLA-B*27-05	binary	9	21	14	7
1026897, 1026891	HLA-B*40-01	binary	9	38	14	24
1026897, 1026891, 1026840	HLA-B*58-01	binary	9	482	63	419
1026891	HLA-A*11:01	binary	9	22	19	3

Table 3.4 Performance of DeepMHC on binding affinity Prediction (IC_{50}) compared to other algorithms (9-length, methods without winning on any dataset are omitted)

MHC allele	Measure type	DeepMHC			NetMHCpan			SMM			ANN			ARB			IEDBConsensus			NetMHCcons			PickPocket		
		AUC	SRCC	Pearson	AUC	SRCC	Pearson	AUC	SRCC	Pearson	AUC	SRCC	Pearson	AUC	SRCC	Pearson	-	-	-	AUC	SRCC	Pearson	AUC	SRCC	Pearson
HLA-A*02:01	IC_{50}	**0.73**	**0.60**	**0.21**	0.71	0.56	0.2	0.69	0.52	0.14	0.7	0.53	0.2	0.69	0.51	0.18	-	-	-	-	-	-	-	-	-
HLA-A*02:01	t1/2	**0.82**	**0.58**	**0.23**	0.81	0.56	0.16	0.81	0.55	0.07	0.8	0.56	0.18	0.8	0.56	0.05	0.73	0.60	**-0.01**	-	-	-	-	-	-
HLA-A*02:02	IC_{50}	0.73	**0.66**	-0.03	0.71	0.59	-0.03	0.7	0.55	-0.01	0.74	0.65	-0.04	0.74	0.5	-0.01	0.66	0.52	0.40	**0.75**	0.63	-0.04	0.71	0.52	-0.02
HLA-A*02:03	IC_{50}	0.66	0.49	**0.62**	0.69	**0.54**	0.46	0.66	0.51	0.26	0.66	0.51	0.39	0.69	0.5	0.13	0.76	0.64	0.30	0.69	**0.54**	0.43	**0.76**	0.48	0.28
HLA-A*02:06	IC_{50}	**0.79**	**0.69**	**0.37**	0.78	0.65	0.21	0.75	0.62	0.11	0.77	0.64	0.2	0.74	0.6	0.24	-	-	-	**0.79**	0.67	0.2	0.74	0.51	0.35
HLA-A*24:02	IC_{50}	0.68	0.37	0.73	0.60	0.21	0.64	**0.74**	**0.4**	0.71	0.59	0.21	0.58	0.44	0.05	**0.85**	-	-	-	-	-	-	-	-	-
HLA-B*35:01	IC_{50}	**0.80**	**0.4**	**0.30**	0.68	0.36	0.26	0.59	0.21	0.27	0.57	0.27	0.26	0.64	0.26	0.25	-	-	-	-	-	-	-	-	-
HLA-B*58:01	IC_{50}	0.56	0.17	0.02	**0.72**	**0.36**	0.07	0.67	0.32	0.06	0.65	0.27	**0.08**	0.55	0.21	0.01	-	-	-	-	-	-	-	-	-
HLA-B*58:01	t1/2	**0.69**	**0.31**	0.05	0.55	0.16	0.12	0.61	0.15	0.11	0.63	0.22	**0.16**	0.59	0.18	0.06	-	-	-	-	-	-	-	-	-
HLA-A*68:01	IC_{50}	0.75	0.60	0.08	**0.84**	0.63	0.10	0.79	0.62	0.03	**0.84**	**0.65**	**0.18**	0.77	0.53	0.14	-	-	-	-	-	-	-	-	-
HLA-A*68:01	t1/2	0.23	-0.50	-0.48	**0.32**	**-0.32**	-0.33	0.25	-0.42	-0.45	0.27	-0.41	-0.37	0.31	-0.39	**-0.29**	-	-	-	-	-	-	-	-	-
HLA-A*68:02	IC_{50}	**0.89**	**0.63**	0.46	0.82	0.54	**0.47**	0.82	0.53	0.39	0.84	0.56	0.43	0.77	0.43	0.32	0.53	0.45	0.84	0.84	0.55	0.45	0.83	0.49	0.24
HLA-B*07:02	IC_{50}	0.88	0.66	0.52	0.8	0.71	0.63	0.86	0.74	0.38	**0.89**	**0.77**	**0.73**	0.77	0.69	0.41	-	-	-	-	-	-	-	-	-
HLA-B*07:02	t1/2	0.94	0.78	**0.56**	0.95	**0.86**	0.45	**0.96**	0.79	0.4	**0.96**	0.84	0.42	0.78	0.53	0.27	-	-	-	-	-	-	-	-	-
HLA-B*57:01	IC_{50}	0.75	0.33	0.37	0.82	**0.5**	**0.4**	0.73	0.26	0.34	**0.85**	0.4	0.35	0.59	0.1	0.38	-	-	-	-	-	-	-	-	-
Count Of Best		**6**	**7**	**6**	3	5	2	2	1	0	4	2	4	0	0	1	0	0	1	2	1	0	1	0	0
Avgerage		**0.73**	0.45	**0.27**	0.72	**0.46**	0.25	0.71	0.42	0.16	0.72	0.45	0.24	0.66	0.35	0.19	-	-	-	-	-	-	-	-	-

Table 3.5 Prediction Performance of DeepMHC on the binding affinity (IC_{50}) problems against other algorithms (10-length, Omitted some methods having no best reults)

MHC allele	Measure type	DeepMHC			NetMHCpan			SMM			SMMPMBEC			IEDBCconsensus		
		AUC	SRCC	Pearson	AUC	SRCC	Pearson	AUC	SRCC	Pearson	AUC	SRCC	Pearson	AUC	SRCC	Pearson
HLA-A*02-01	t1/2	**0.76**	**0.4**	**0.07**	0.57	0.15	-0.13	0.56	0.14	-0.10	-	-	-	-	-	-
HLA-A*02-01	IC_{50}	**0.69**	**0.58**	0.09	0.68	0.46	0.09	0.66	0.50	**0.16**	0.62	0.5	0.15	0.66	0.50	0.09
HLA-A*02-03	IC_{50}	**0.80**	**0.39**	-0.05	0.75	0.25	-0.06	0.73	0.32	0.00	0.7	0.31	-0.01	0.73	0.30	**0.02**
HLA-A*02-06	IC_{50}	0.82	**0.66**	0.14	0.81	0.64	0.15	0.83	0.64	0.21	**0.84**	**0.66**	**0.22**	-	-	-
HLA-A*68-02	IC_{50}	**0.94**	**0.78**	**0.51**	0.66	0.31	0.12	0.75	0.41	0.13	0.73	0.38	0.11	0.73	0.40	0.25
Count Of Best		4	5	1	0	0	0	0	0	1	1	1	1	0	0	0
Avgerage		**0.80**	**0.56**	**0.15**	0.69	0.36	0.04	0.71	0.40	0.08	-	-	-	-	-	-

Table 3.6 Prediction Performance of DeepMHC on binary peptide binding prediction

MHC allele	DeepMHC			NetMHCpan			SMM			ANN		
	AUC	SRCC	Pearson	AUC	SRCC	Pearson	AUC	SRCC	Pearson	AUC	SRCC	Pearson
HLA-A*02-01	0.87	0.60	0.59	**0.89**	**0.63**	**0.66**	0.88	0.62	0.14	0.88	0.61	0.64
HLA-A*11-01	**0.63**	**0.16**	**0.53**	0.58	0.09	0.49	0.60	0.11	0.37	0.61	0.14	0.4
HLA-A*24-02	0.81	0.41	0.24	**0.89**	**0.51**	0.63	0.87	0.48	0.23	0.89	0.51	**0.65**
HLA-A*30-01	**0.89**	**0.20**	0.15	0.80	0.16	0.16	0.79	0.15	0.06	0.77	0.14	**0.17**
HLA-A*31-01	0.88	0.64	0.55	**0.92**	**0.71**	0.74	**0.92**	**0.71**	0.34	**0.92**	**0.71**	0.7
HLA-A*68-01	0.85	0.36	0.27	**0.87**	**0.38**	0.47	0.86	0.37	0.11	0.87	0.38	**0.52**
HLA-B*07-02	0.90	0.44	0.42	0.92	0.46	0.57	**0.93**	**0.47**	0.09	0.93	0.47	**0.71**
HLA-B*27-05	0.88	0.62	0.70	**0.96**	**0.75**	**0.79**	0.91	0.67	0.69	0.94	0.72	0.78
HLA-B*40-01	**0.89**	0.65	0.35	**0.89**	**0.66**	0.42	0.88	0.63	0.26	0.88	0.64	**0.48**
HLA-B*58-01	**0.89**	**0.45**	0.39	0.88	0.44	0.53	**0.89**	**0.45**	0.08	0.87	0.44	**0.59**
Count of Best	4	3	1	**6**	**6**	2	3	3	0	1	1	**6**
Avgerage	0.85	0.45	0.42	**0.86**	**0.48**	0.55	0.85	0.47	0.24	**0.86**	**0.48**	**0.56**

Table 3.7 Prediction Performance of DeepMHC V.S. HONN on BLIND Dataset (AUC scores)

Allele	DeepMHC	HONN	Allele	DeepMHC	HONN
A*01:01-BLIND	0.89	0.89	A*30:01-BLIND	0.91	0.91
A*02:01-BLIND	0.92	0.92	A*30:02-BLIND	0.72	**0.78**
A*02:02-BLIND	**0.90**	0.86	A*31:01-BLIND	0.86	**0.88**
A*02:03-BLIND	0.96	0.96	A*33:01-BLIND	0.91	0.91
A*02:06-BLIND	0.87	0.87	A*68:01-BLIND	0.92	0.92
A*03:01-BLIND	0.89	**0.92**	A*68:02-BLIND	**0.96**	0.95
A*11:01-BLIND	0.94	0.94	A*69:01-BLIND	**0.93**	0.92
A*23:01-BLIND	0.85	**0.86**	B*07:02-BLIND	**0.90**	0.88
A*24:02-BLIND	0.79	**0.81**	B*08:01-BLIND	**0.95**	0.94
A*26:01-BLIND	0.92	0.92	B*15:01-BLIND	0.90	**0.92**
A*29:02-BLIND	0.86	**0.89**	B*27:05-BLIND	0.91	0.91
B*44:02-BLIND	0.84	**0.91**	B*35:01-BLIND	**0.86**	0.85
B*51:01-BLIND	0.92	0.92	B*39:01-BLIND	0.95	**0.96**
B*57:01-BLIND	0.95	0.95	B*40:01-BLIND	0.93	0.93
B*58:01-BLIND	0.96	0.96			

Table 3.8 Prediction Performance of DeepMHC V.S. JHU Proposed CNN model

Allele	Measure Type	JHU-CNN			DeepMHC		
		AUC	SRCC	Pearson	AUC	SRCC	Pearson
HLA-A*02:01	IC_{50}	**0.75**	**0.66**	**0.28**	0.73	0.60	0.21
HLA-A*02:01	binary	0.80	0.49	0.49	**0.87**	**0.60**	**0.59**
HLA-A*02:01	t1/2	0.73	0.46	**0.27**	**0.82**	**0.58**	0.23
HLA-A*02:02	IC_{50}	**0.75**	**0.69**	-0.04	0.73	0.66	**0.03**
HLA-A*02:03	IC_{50}	**0.68**	**0.54**	**0.70**	0.66	0.49	0.62
HLA-A*02:06	IC_{50}	**0.86**	**0.77**	**0.42**	0.79	0.69	0.37
HLA-A*24:02	IC_{50}	0.58	0.32	0.43	**0.68**	**0.37**	**0.73**
HLA-A*24:02	Binary	0.75	0.32	**0.38**	**0.81**	**0.41**	0.24
HLA-A*68:01	IC_{50}	0.58	0.22	**0.14**	**0.75**	**0.60**	0.08
HLA-A*68:01	Binary	0.74	0.25	0.26	**0.85**	**0.36**	**0.27**
HLA-A*68:01	t1/2	**0.34**	-0.39	-0.33	0.23	**0.50**	**0.48**
HLA-A*68:02	IC_{50}	**0.92**	**0.77**	**0.61**	0.89	0.63	0.46
HLA-B*07:02	IC_{50}	0.75	0.47	0.51	**0.88**	**0.66**	**0.52**
HLA-B*07:02	Binary	0.84	0.37	**0.45**	**0.90**	**0.44**	0.42
HLA-B*07:02	t1/2	0.78	0.51	0.39	**0.94**	**0.78**	**0.56**
HLA-B*35:01	IC_{50}	0.65	0.29	0.20	**0.80**	**0.40**	**0.30**
HLA-B*57:01	IC_{50}	0.66	0.27	0.25	**0.75**	**0.33**	**0.37**
HLA-B*58:01	IC_{50}	**0.66**	**0.34**	**0.32**	0.56	0.17	0.02
HLA-B*57:01	Binary	0.82	0.37	**0.40**	**0.89**	**0.45**	**0.39**
HLA-B*58:01	t1/2	**0.70**	**0.37**	**0.29**	0.69	0.31	0.05

Table 3.9 Prediction Performance of DeepMHC V.S. HLA-CNN

Allele	Measure Type	Peptide Length	Peptide Count	DeepMHC		HLA-CNN	
				AUC	SRCC	AUC	SRCC
HLA-B*57:01	IC_{50}	9	26	**0.82**	**0.65**	0.81	0.44
HLA-A*68:02	IC_{50}	9	55	0.88	0.62	**0.99**	**0.72**
HLA-A*02:06	IC_{50}	9	55	0.78	**0.68**	**0.82**	0.58
HLA-A*02:03	IC_{50}	9	55	0.66	**0.48**	**0.75**	0.37
HLA-A*02:01	IC_{50}	9	55	0.57	**0.62**	**0.68**	0.58
HLA-A*02:01	IC_{50}	10	35	**0.66**	**0.56**	0.59	0.33
HLA-A*02:03	IC_{50}	10	35	0.80	**0.35**	**0.84**	0.31
HLA-A*02:06	IC_{50}	10	35	0.81	0.60	**0.92**	**0.64**
HLA-A*68:02	IC_{50}	10	35	0.92	**0.74**	**0.99**	0.72
Average				0.77	**0.59**	0.82	**0.52**

Chapter 4

Pan-specific model on peptide-HLA class I binding affinity prediction

4.1 Introduction

In Chapter 3, we introduced an allele-specific CNN-based model on the prediction of peptide-HLA Class I binding. As discussed in Section 3.4 of last chapter, a major con of allele-specific methods is that they could not be trained and be used to predict on HLA alleles with few or zero samples since the peptide is the only input. Due to the high polymorphism of HLA genes, as March 2018, there are more than 17,000 HLA alleles deposited in the IMGT/HLA database [81]. For many HLA alleles, there are actually only a few or no experimentally determined binding affinity data available. In contrast, a pan-specific method takes a pair of HLA allele and its binding peptide as an input. In this way, binding pairs across all alleles can be integrated as one training dataset. To achieve this goal, besides encoding the peptide, the peptide-HLA binding context should also be represented in a proper way such that the chosen machine learning model could utilize all pairs of binding samples. The key idea behind pan-specific models is that besides encoding the peptide in a proper way for the prediction model, the peptide-HLA binding context/environment is also represented so that the machine learning models could be trained on all available peptide-HLA binding samples [110]. In Figure 4.1, we show the difference between allele-specific methods and pan-specific methods.

In this chapter, we will first discuss the existing pan-specific models and their limitations in section 4.2. Then our proposed method is presented in section 4.3. We test our model on benchmark dataset and compare the results with other methods in section 4.4.

4.2 Related work

As of now, a number of pan-specific models have been proposed for both HLA class I and class II alleles [110]. Pseudo-sequence based methods are currently most suc-

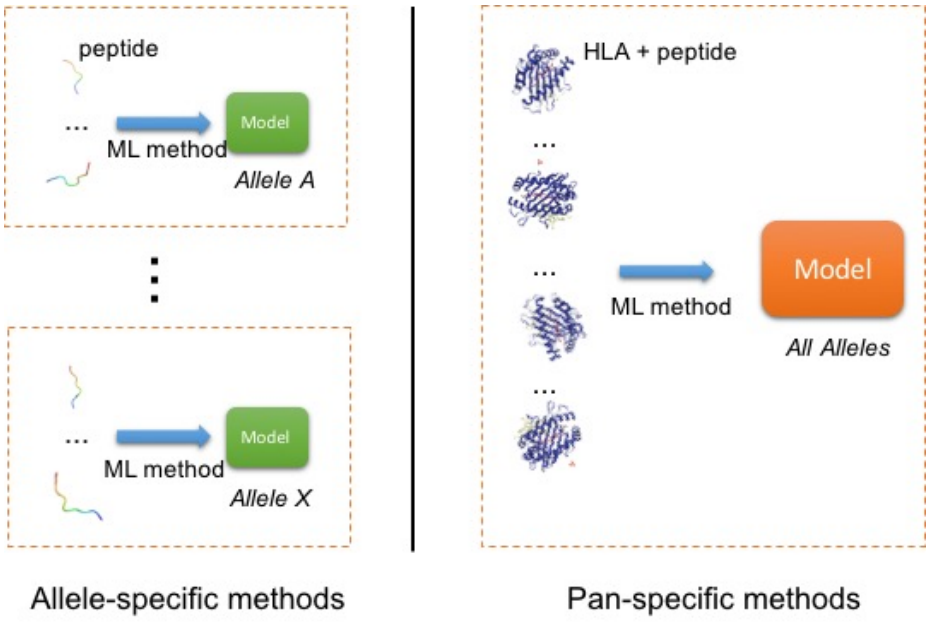


Figure 4.1 Illustration of difference between allele-specific framework and pan-specific framework. Left panel show the allele-specific framework where for each allele, a model needs to be trained on available peptides. In the right panel, a pan-specific model could takes into binding complex of all alleles together and a universal model will be obtained.

cessful ones with high performance when trained on a large amount of HLA class I binding affinity data. NetMHCPan [26] is the first pan-specific binding affinity prediction algorithm that trains on a large number of peptide-HLA binding samples of different HLA alleles. It proposed the pseudo sequence encoding method to represent the binding context, in which an HLA sequence is reduced to a pseudo amino acid sequence of length 34 based on a representative set of HLA structures with nonamer peptides. Each amino acid in this pseudo sequence is selected if it is in contact with the peptide within 4.0 $\mathring{A}$ (0.4 nm). The extracted interaction map is shown in Figure 4.2. this extracted 34-length pseudo sequence is a fixed list of location indexes of amino acids. For any given HLA allele's sequence, the corresponding 34 residues are extracted and used to represent this HLA allele. In NetMHCPan, a HLA-peptide binding sample is represented as a 43-length residue sequence: 9 from peptide and 34

from HLA. This sequence is then encoded in three different ways: one-hot encoding, BLOSUM50 and a mixture of both. The encoded input is then used to train multiple feed forward neural networks with 22 to 86 hidden neurons. Then the network with the highest prediction performance (lowest square error) on the test set was selected as the final prediction model [26]. NetMHCPan later has been improved several times by training with additional training data. The latest version is NetMHCPan 4.0 [36].

This pseudo sequence encoding approach has also been used in PickPocket [108] and Kim's algorithm [23], but the two methods use different machine learning algorithms for model training. In PickPocket, position-specific scoring matrices (PSSMs) are first derived from peptides data. Then extract the position-specific vectors from the PSSMs in association with pseudo-sequence to construct a pocket library. Each pocket library entry is characterized by nine pairs, where each pair consists of a list of pocket amino acids and a specificity vector. Kim et al. proposed a pan-specific DCNN model for peptide-MHC class I binding prediction, in which pseudo sequence encoding is adopted for HLA sequences and the DCNN model is setup as a 26-layer classifier [23].

The pseudo sequence encoding is currently the dominating binding context encoding method in pan-specific peptide-HLA class I binding prediction. This encoding method has its potential limitations: 1) its interaction map extraction step relies on available MHC-peptide bound complex structures,which can not cover all alleles, especially considering the high polymorphism of HLA proteins; 2) the 34 contact residues of the encoding is empirical and only covers part of the whole HLA sequence.

In next section, we propose DeepSeqPan, a deep neural network trained on pairs of one-hot encoded raw peptides and HLA sequences. Our method does not require 3-D structural data during training stage and it could obtain state-of-the-art performance on standard benchmark.

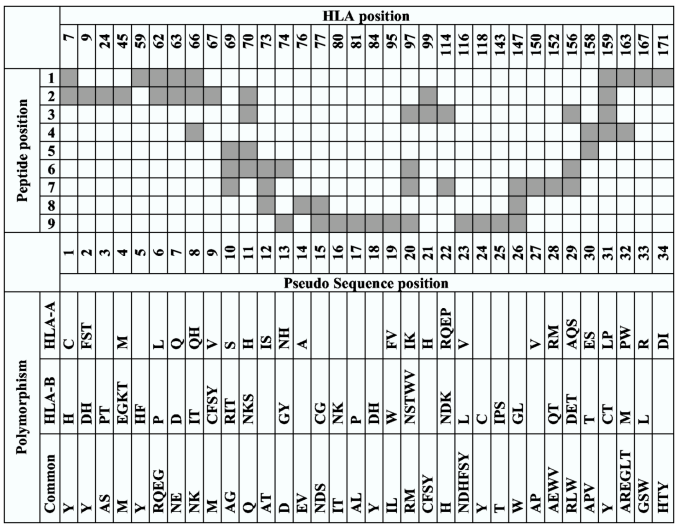


Figure 4.2 Interaction map of the HLA pseudo sequence in NetMHCPan. Reproduced from original paper.

4.3 Method

4.3.1 Dataset

We use same training dataset BD2013 from IEDB as we did for DeepMHC. All training samples are labeled with IC_{50} binding affinity values. The testing dataset is downloaded from IEDB's weekly benchmark dataset ranging from 2014-03-21 to 2018-01-26 (http://tools.iedb.org/auto_bench/mhci/weekly). To address the concern that duplicate peptides may exist in both the training and testing data downloaded from IEDB, we removed all duplicate peptides from the testing dataset. The alignment-ready HLA sequences were obtained from IMGT/HLA database [81]. We trained our model on 9-length peptides binding to HLA-A, HLA-B and HLA-C alleles with available HLA sequences. Totally, the training dataset contains 121,787 peptide-HLA binding peptides covering 42 HLA-A alleles (72618 samples), 49 HLA-B alleles (46915 samples) and 10 HLA-C alleles (2254 samples). The detailed information of the training dataset are listed in Table 3.1

4.3.2 Sequence Encoding

In this model, we use one-hot encoding for both peptide and HLA sequences. A 9-length peptide sequence is encoded into a 2D tensor with dimension $1 \times 9 \times 20$ where the last dimension is the number of channels and each channel represents one of 20 amino acids. Figure 4.3 illustrates the encoded peptide HLNPNKTKR as a 2D tensor with dimension $1 \times 9 \times 20$. Since HLA class I sequences downloaded from IMGT/HLA database have variable lengths, we chose the maximum length 372 as the fixed dimension. Then we encode each aligned HLA sequence into a 2D tensor with dimension $1 \times 372 \times 21$. The extra channel represents gaps in HLA sequences shorter than 372.

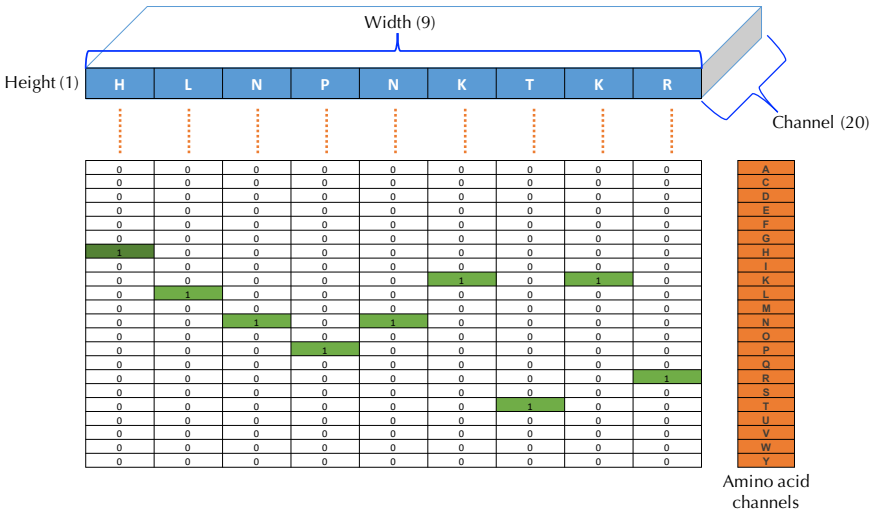


Figure 4.3 Peptide encoding example. Sequence HLNPNKTKR is encoded into a 2D tensor with dimension 1 (height) × 9 (width) × 20 (channel). Each of 20 channels represents one amino acid type and we set a channel value to 1 if the corresponding amino acid appears at this location of the input sequence.

4.3.3 DeepSeqPan

Architecture. As shown in Figure 4.4, the DeepSeqPan network consists of three parts:

(i) **Peptide encoder and HLA encoder.** The peptide and HLA encoders convert a pair of one-hot encoded peptide and HLA sequences into two tensors with a unified dimension $1 \times 9 \times 10$. The output tensors of two encoders are concatenated along the channel axis to generate an encoded feature tensor with dimension $1 \times 9 \times 20$. Then this concatenated tensor will be fed into the binding context extractor in (ii). The purpose of these two encoders is extracting high-level features from raw sequences and encoding them into a feature tensor. Different from the 34 pseudo amino acid sequence encoding approach in [26], the features and information stored inside this feature tensor are learned by the deep neural network automatically with its end-to-end training framework. The encoder of the peptide consists of two blocks of convolutional, batch normalization and LeakyReLu layers. As for the HLA encoder whose input sequence is much longer than the peptide, we used a network configuration similar to the VGG network [89].

(ii) **Binding context extractor.** The extractor takes into the encoded feature tensor from (i) and outputs a 2560-dimension vector. This vector is actually the binding context between a peptide and a HLA. This binding context extractor will be optimized automatically in the training stage through the back-propagation algorithm and the extraction of the binding context is done by the network itself without human involvement. Especially, in this extractor we use Locally Connected layers (as illustrated LCBlock in Figure 4.4) instead of standard convolutional layers with weights sharing. The reason is that the encoded high-level features in the feature tensor is position related, i.e. in the encoded feature tensor with length 9, an extracted feature $\mathcal{A}$ located at position 1 should have different effect as it appears at position 7. Locally connected layers have the capability to capture features at specific locations since its filters at different locations do not share weights, which has been proved to be powerful in

DeepFace [93].

(iii) **Affinity predictor and binding probability predictor.** Another novel design of DeepSeqPan is that at the output layer, both the binding probabilities and the IC_{50} value are used as output in final stage (iii). This is different from all other DCNN based MHC binding prediction algorithms[23, 99] which outputs either the binding probabilities or IC_{50} values. This design is not a captain's call. Actually, at first when we were training the DeepSeqPan that only predicts IC_{50} values, we found it was very hard to train the network with very slow convergence. So we added the binding probability predictor as an additional source of supervision signal with the expectation that the backpropagation algorithm can train the network easier by taking advantage of two types of losses: the classification loss and regression loss. Note that we can calculate the binary binding probability for the training samples from their IC_{50} binding affinity values via Equation (4.4). The underlying relationship between regression outputs and classification outputs is built up naturally. In the training stage, the network needs to learn this underlying relationship in order to reduce the total loss. In that case, we argue that the classification predictor plays as a regularizer by forcing the network to predict a more accurate IC_{50} values.

Layer configuration details. In Figure 4.4, there are several layers we need to give more details:

- **FC *N*:** A fully connected layer with N hidden units.
- **Dropout:** We used 0.5 as the dropout rate.
- **ConvBlock *N*:** A ConvBlock N consists of 4 layers in following order:
  1. A 2D convolutional layer with N filters of size 1×3
  2. A Batch normalization layer

3. A 2D convolutional layer with N filters of size 1×3
4. A Max pooling with kernel size 1×2

- **LCBlock *N*:** A LCBlock N consists of 3 layers in following order:
  1. A 2D Locally Connected layer with N filters of size 1×2
  2. A Batch normalization layer
  3. A LeakyReLU activation layer

Loss function. The overall loss $\mathcal{L}$ is the sum of the regression loss $\mathcal{L}_{\mathcal{R}}$ and the classification loss $\mathcal{L}_{\mathcal{C}}$ as illustrated in Equation (4.1).

$$\mathcal{L} = \mathcal{L}_{\mathcal{R}} + \mathcal{L}_{\mathcal{C}} \tag{4.1}$$

For IC_{50} predictor, we use mean squared error (MSE, Equation (4.2)) as the loss function and for the binary binding predictor, the binary cross entropy loss (Equation (4.3)) is used.

$$\mathcal{L}_{\mathcal{R}} = \frac{1}{N} \sum ||Y_{IC_{50}} - Y'_{IC_{50}}||^2 \tag{4.2}$$

$$\mathcal{L}_{\mathcal{C}} = -Plog(P') - (1-P)log(1-P') \tag{4.3}$$

To get binary binding labels, we use standard 500 nM threshold to convert a IC_{50} value label into a binding label:

$$P = \begin{cases} 1, & \text{if } IC_{50} < 500. \\ 0, & \text{otherwise.} \end{cases} \tag{4.4}$$

Training. We randomly split all training samples into a training set and a validation set following 4:1 ratio. Stochastic gradient descent (SGD) is employed as the optimization algorithm enabled with momentum and learning rate decay. The initial learning rate is 0.001 and the momentum factor is 0.8. It is scheduled to halve

the learning rate if validation loss hasn't improved within 5 epochs. The minimum learning rate is set to 0.00001. The training process stops if the validation loss has not improved within 15 epochs. We used Keras [14] deep learning framework to implement our DeepSeqPan algorithm.

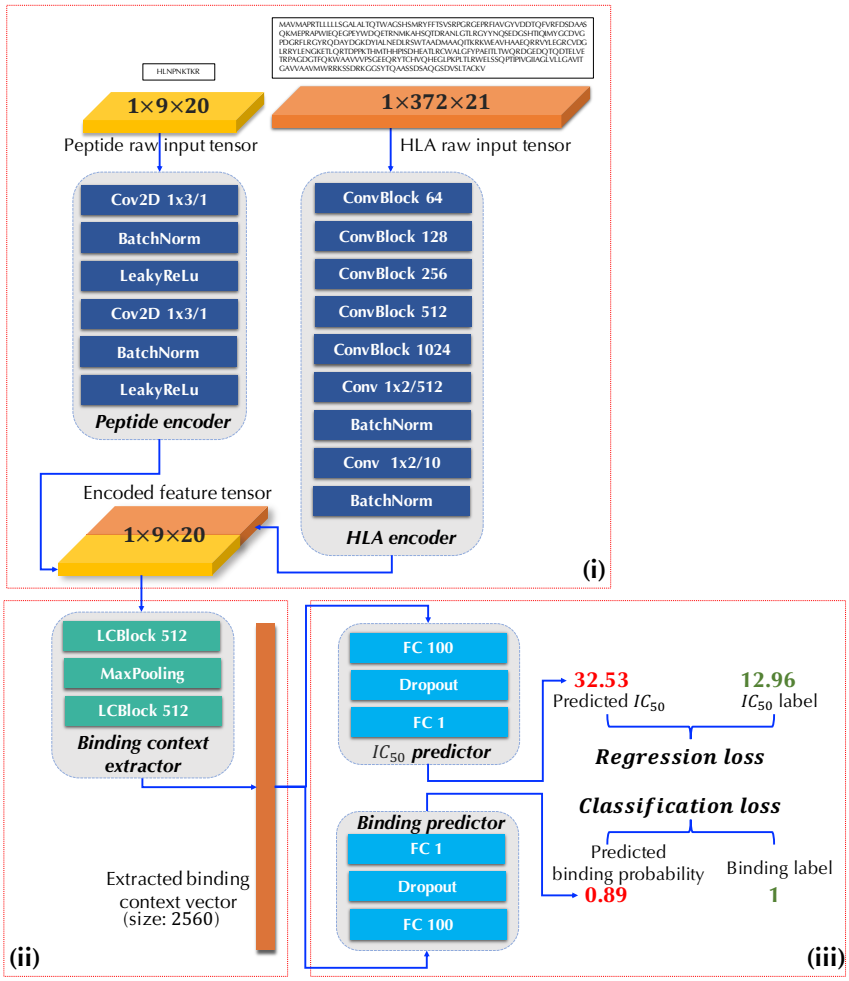


Figure 4.4 DeepSeqPan Network Structure. (i) Peptide and HLA encoders. (ii) Binding context extractor. (iii) Affinity and binding predictors.

4.3.4 Metrics and label processing

Area under the curve (AUC) and Spearman's rank correlation coefficient (SRCC) are used as evaluation metrics to compare with the public benchmark results at IEDB

website [96]. In pan-specific binding prediction modeling, the IC_{50} values of the peptides span a large range $[0, 80000]$. To avoid gradient explosion issue in neural network training, we convert IC_{50} to $\log IC_{50}$ via Equation 4.5. The $\log IC_{50}$ are then used as labels during training. During inference stage, we convert the prediction results back to IC_{50} values.

$$\log IC_{50} = log_e IC_{50} \tag{4.5}$$

4.4 Experiments and results

4.4.1 Cross-validation on the training dataset

Standard five-fold cross-validation experiments are applied over the training dataset. Since our network outputs both IC_{50} affinity values and binding probabilities, we evaluated the performances on both outputs separately. We use AUC and SRCC discussed in section 4.3.4 to measure the performance.

In table 4.1, *All* rows show the 5-fold cross-validation results of our algorithm on the original training dataset. When the calculated the metrics on all samples, DeepSeqPan achieved a high AUC of 0.94 for regression on IC_{50} and an AUC of 0.94 for binary binding classification. The corresponding SRCC is 0.73 (IC_{50}) and 0.70 (binary binding) respectively. When evaluated over the samples of HLA-A, -B and -C alleles separately, all the AUC scores are around or above 0.90 and all the SRCC scores are above 0.60. More comprehensive metrics evaluated on each allele are reported in Table 4.2. In this table, some AUC scores are N/A because of that in training dataset, there are few alleles coming with 0 positive samples. Thus, it is not applicable to compute AUC score. Also, we can find that some SRCC scores are also N/A in Table 4.2. These happen if the number of an allele's samples is < 2. It is not applicable to calculate SRCC score without at least 2 data points. For allele HLA-

A*02:10, the Spearmanr function gives N/A result because all samples are labeled as same values which is also a non-calculable situation.

Besides performing cross validation on the original training dataset, we also did a cross validation on a CD-HIT [52] filtered training dataset. The reason behind this is that in the original training dataset, some peptides have high similarities. When do cross validation on the original training dataset, these similar peptides may lead to over estimating of our model's performance. CD-HIT is a widely used tool to cluster protein sequences based on their alignment similarities. We first use it to group all peptides in our training dataset with sequence identity threshold 0.7, i.e. if two peptides have similaritiy > 0.7, they will be clustered into on group. After this step, 20,148 unique peptide sequences are grouped into 14,812 clusters. Then for each cluster, we only keep peptide with the greatest number of samples. In this way, we got a new training dataset for cross-validation, we name it *training-cd-hit*. Then we did the normal 5-fold cross-validation on this training set. In Table 4.2, we listed the performance of model trained on this filtered dataset (*CD-HIT filtered* rows). As shown in the table, AUC scores are much better than model trained on all dataset with all achieved 1.0. SRCC scores are slightly worse than the model trained on all dataset. We can see that our model keeps the performance on the training-cd-hit dataset.

4.4.2 Evaluation on benchmark dataset

To evaluate how our DeepSeqPan performs compared to other HLA-peptide binding prediction algorithms, we applied it to the public IEDB weekly benchmark dataset upon which a set of top algorithms have been evaluated with published results.

We trained a single DeepSeqPan model on all 9-length peptides in the training dataset that bind to HLA-A, -B and -C alleles. Then this trained model was evaluated on all available IEDB weekly benchmark dataset [96]. As we inform before, the IEDB

benchmark dataset has been filtered by removing duplicate samples. We compared the performance of DeepSeqPan with those of pan-specific models: NetMHCPan (2.8) [25], NetMHCPan (3.0) [68] and PickPocket [108], the performances of allele-specific models: SMM [76], NetMHC (3.4) [59], NetMHC (4.0) [2], ARB [12], MHCflurry [73] and AMMPMBEC [96], and those of ensemble models (results are based on several different models): IEDB Consensus [96] and NetMHCcons [38]. Metrics of compared models are listed in Table 4.6. This table summarizes the performance of different algorithms on 64 testing datasets from IEDB benchmark database. For each dataset, we highlighted the highest AUC scores in yellow and highest SRCC scores in pink and then counted the number of datasets upon which each algorithm achieved the highest scores and put them at the last row of the table. We found that DeepSeqPan achieved the highest AUC scores in 19 records out of total 64 testing records. In 45 records that DeepSeqPan didn't achieve the highest AUC scores, there are 28 records on which the AUC scores of DeepSeqPan are very close to the highest AUC scores within a small margin around 0.1. In terms of SRCC, DeepSeqPan obtained the highest scores on 13 records.

From Table 4.6, it can be found that different pan-specific and allele-specific methods have the best performance on datasets of various alleles, which implies the good performance of the ensemble methods such as NetMHCcons since they make prediction via combining results from multiple methods [96]. Our proposed DeepSeqPan could thus be a complementary tool for existing pan-specific models and it is promising to include it into the state-of-the-art ensemble prediction models to improve their performance.

4.4.3 Comparison with other DCNN models

To the best of our knowledge, Kim et lt.'s work [15] is the only pan-specific model that employs DCNN architecture beside our proposed DeepSeqPan. It uses NetMHC-

Pan's pseudo sequence encoding for binding context modeling, in which a pair of peptide-HLA binding sample is encoded into a 9 (height) × 34 (width) × 18 (channel) 2D tensor. Each "pixel" in this 2D tensor represents a contacting pair of a peptide reside and a HLA residue. For two contacting residues, 9 physicochemical properties are used for each one and in total 18 values are encoded in channels. Their network structure is VGG-like and consists of 26 layers. They trained their model with binding samples on HLA-A and HLA-B alleles and it used the same dataset BD2013 as we did. To compare the performance of our DeepSeqPan with Kim's method, we evaluated the benchmark dataset with their online server (http://jumong.kaist.ac.kr:8080/convmhc) on all its supporting alleles (HLA-A and HLAB alleles). In total, we evaluated 54 benchmark datasets on Kim's server and compared with ours obtained in previous benchmark evaluation and the binary prediction outputs were used to compare. Since Kim's model was trained as a classifier, we calculated AUC scores for each testing dataset and in Table 2 we showed the average AUC scores measured based on all HLA-A or HLA-B testing dataset respectively (Detailed performance on each dataset is listed in Supplementary Files). Out of all 54 benchmark datasets, Kim's model and our model both got an average AUC of 0.76. For HLA-A datasets, two model also obtained same average AUC of 0.74. Our model slightly out performed Kim's model on HLA-B alleles with an average AUC of 0.80. Overall, two models achieved similar performance and in terms of performance on each allele as shown in Table 4.3, two models obtained better performances on different sets of HLA alleles and none can dominate the other model.

4.4.4 Generalization of DeepSeqPan to binding predictions of new HLA alleles

One major advantage of pan-specific models over allele-specific models for HLA-peptide binding prediction is that it can make predictions on HLA alleles that are not

included in the training dataset. This is especially useful for HLA alleles without any samples with known binding affinity values. In order to evaluate this extrapolation capability of DeepSeqPan, we setup a 3-fold blind cross validation. We first grouped all alleles into 3 groups: HLA-A, -B and -C. In this blind test, each of 3 models were trained on 2 groups of alleles and then tested on another group of alleles. We call those trained model Blind models since they didn't see any alleles from testing. All samples are taken from the CD-HIT training dataset such that we can compare these 3 model's performances with that of 5-fold cross-validation. The CD-HIT training dataset was obtained from previous cross-validation experiment.

The comparison results are shown in Table 4.4. The columns under 3-fold blind test are performance results of the Blind models. Columns under 5-fold cross validations are results obtained from previous 5-fold crossvalidation on same training dataset. Overall, the blind models perform worse than 5-fold cross validation as expected since in both 3 folds, the training datasets are much smaller comparing with that of 5-fold cross validations. For example, when tested on HLA-A group alleles, the model was trained on 43,462 samples while it was tested on 60,987 samples. The number of testing samples is even larger than the number of training samples. Same situations for HLA-B and HLA-C groups. However, as shown in Table 4.4, thereâĂŹre several records in groups B and C where blind model achieved better or similar performance (HLA-B*15:42, HLA-B*27:02, HLAB*42:02, HLA-B*45:06, HLA-B*52:01, HLA-B*57:03, HLA-B*58:02 and HLA-C*04:01). We think this happened because when the network trained on A-B dataset or A-C dataset, it could learned the basic cross allele features with a number of samples (102,347 of A-B, 63,088 of A-C) even only fed with raw sequences.

4.4.5 The binding context vector: consistency and capability

One of key design features of our DeepSeqPan model (Figure 4.4 (ii)) is the binding context vector aiming to capture high-level features that determine whether a peptide and a HLA bind or not and if so, how strong the binding is. Another key feature of our model is the dual outputs of the model: the binding affinity output and the binary binding probability output.

Since the binding context vector is used as input for both predicted outputs, it should be consistent for both the IC_{50} predictor and the binary binding predictor in (iii): for the same binding context vector, both predictors should give consistent outputs. In other words, higher binding probability should correspond to higher binding affinity values. To verify this consistency, we inspected IC_{50} and binary prediction outputs of all samples from previous cross-validation and benchmark evaluation experiments. The analysis results are shown in Table 4.5. First row of the table lists the numbers of evaluated samples in the cross-validation and benchmark evaluation experiments with 121,787 in cross-validation and 19,741 in benchmark evaluation. The second row shows the number of consistent outputs. Given a sample pair of peptide and HLA, we marked its predicted IC_{50} value and the predicted binding probability as consistent if both values indicate binding or not binding. An IC_{50} value of < 500 or a binding probability of 0.5 or greater indicates the binding state. From the table we can observe that high consistency exists between the regression and classification outputs. For cross-validation experiments, the percentage of consistent outputs is 95.81% and for benchmark evaluation experiments, this percentage is 86.14%.

In last two rows of Table 4.5, we reported the number of correct predictions measured with the IC_{50} predictions and the binary prediction respectively. A predicted IC_{50} value or a predicted binary binding probability value is marked as a correct prediction if its real label and the prediction value indicate the same binding state: binding or not binding. Given a sample, it will be marked as binding if its IC50 value

is less than the threshold (500 nM). If the sample binding affinity is labelled with t1/2 type, measured minutes less than 120 indicates the binding state For binary binding labels, a binary label of 1 means it is binding while a value of 0 means no binding. For the predicted binary binding probability, a probability value of > 0.5 means binding. From Table 4.5, we found that both affinity and binary binding outputs obtained accuracies greater than 88% in cross-validation experiments. In benchmark experiments, the accuracy rate is 59% for IC_{50} predictions and binary predictions have an accuracy of 53%. The results showed that the consistency between IC_{50} predictions and binary predictions is high, which means that the binding context vector extracted by DeepSeqPan contains common effective features for determining binding states.

In Figure 4.5 we plotted the correlation between binary values and regression values predicted on benchmark dataset. Each dot in this plot represents a testing sampleâĂŹs two prediction values by DeepSeqPan. The x axis value is the predicted logIC50 and the y axis value is the predicted binary binding probability. The Pearson correlation value calculated is -0.97. We also fitted a linear function into these correlation data and the fitted linear function is $y = -0.1x + 1.0$. From the figure, it shows that two output values have very strong correlation when both indicate very strong binding or very weak binding (upper left part and lower right part). It displays weak correlation when the two predicted values indicate the binding is neither strong or weak.

Though the results show that the two output values have a pretty strong correlation overall. There are some cases they will give contradicting predictions. We think the major reason behind this is that in the training dataset, thereâĂŹre some edge samples. Following the 500 nM hard convention value, for samples whose IC_{50} values are slightly above 500 (e.g. 502.03), their binary labels will be 0. And for those samples whose IC50 values are slightly below 500 (e.g. 498.33), their binary labels will be 1. After training, when the model does prediction on those similar

edge samples in testing dataset, the predicted regression values and binary labels are easy to contradict to each other. It can be seen that around hard convention line ($x = 6.20$) in Figure 4.5, the correlation is almost weakest. In practical usage, users should be careful on predicted regression values around 500 and measure the result based on two outputs. But if users only care strong binding samples, according to our correlation analysis on benchmark testing, the two values show strong correlation in strong binding cases. That will not be a problem.

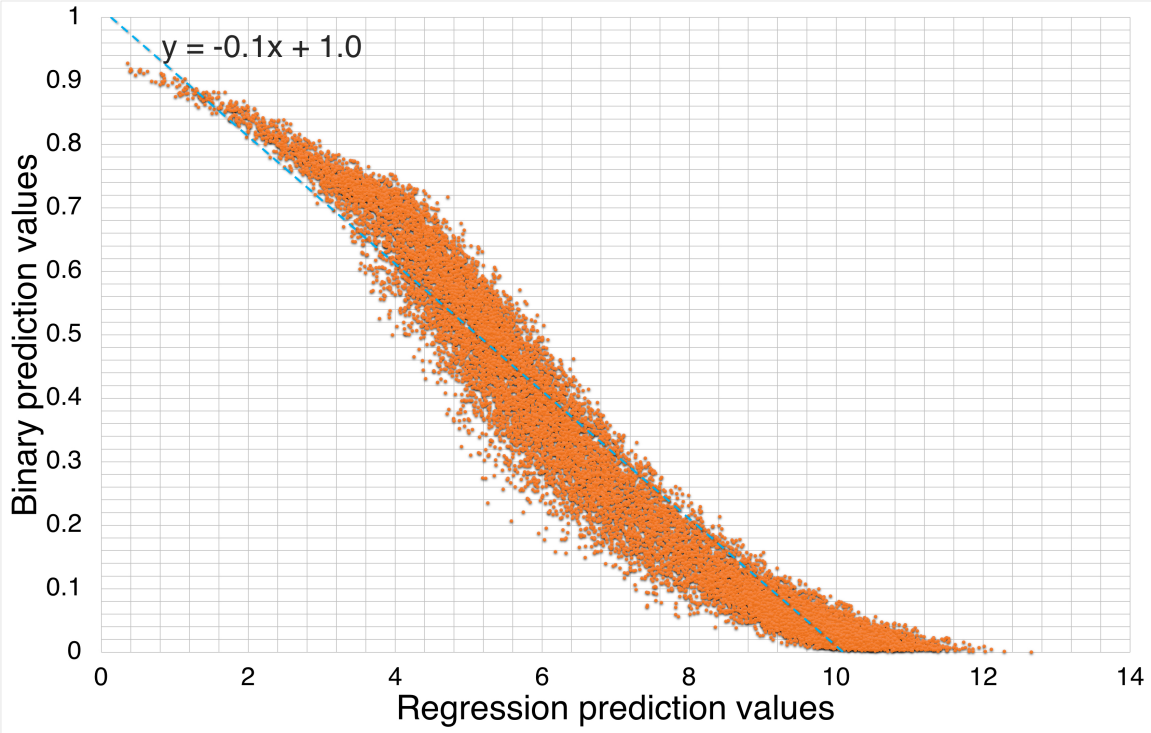


Figure 4.5 Correlation analysis between binary prediction values and regression prediction values on benchmark dataset.

4.5 Chapter summary

In this chapter, we proposed, DeepSeqPan, a novel deep convolutional neural network model for pan-specific HLA-peptide binding affinity prediction. This model is

characterized by its capability of binding prediction with only the raw amino acid sequences of the peptide and the HLA, which makes it applicable to HLA-peptide binding prediction for HLA alleles without structural information. This is achieved by a novel sequence-based encoding of the peptide-HLA binding context, a binding context feature extractor, and the dual outputs with both binding affinity and binding probability predictions. Extensive evaluation of DeepSeqPan on public benchmark experiments showed that our model achieves state-of-the-art performance on a variety of HLA allele datasets.

Our model contributes to the study of MHC-peptide binding prediction in a few special ways. First, our experiments showed that it is possible to extrapolate the binding prediction capability to unseen HLA alleles, which is important for pan-specific models. Second, our sequence-only based binding context encoding is complementary to the pseudo sequence encoding, which is currently the only encoding method used in pan-specific models for class I MHC-peptide binding affinity prediction. This has the potential to further improve the state-of-the-art prediction models such as the pan-specific model NetMHCSpan. It showed the importance of sufficient amount of training data to achieve high prediction performance for deep learning models. Moreover, our proposed sequence-based DCNN architecture for protein-peptide binding is universal and can be adapted to other similar binding problems such as protein-DNA, protein-RNA and protein-ligand/drug bindings.

Table 4.1 Cross validation results on original training data (All) and CD-HIT filtered training data (CD-HIT filtered)

TrainingDataset	Alleles	Seq Count	IC50		Binary Binding	
			AUC	SRCC	AUC	SRCC
All	All alleles	121,787	0.94	0.73	0.94	0.70
	HLA-A	72,618	0.94	0.75	0.94	0.73
	HLA-B	46,915	0.94	0.68	0.94	0.64
	HLA-C	2,254	0.89	0.70	0.89	0.69
CD-HIT filtered	All alleles	104,449	1.00	0.71	1.00	0.68
	HLA-A	60,987	1.00	0.73	1.00	0.71
	HLA-B	41,360	1.00	0.66	1.00	0.62
	HLA-C	2,102	1.00	0.69	1.00	0.68

Table 4.2 5-fold cross validation measured for each allele (All training)

HLA	IC_{50}		Binary		HLA	IC_{50}		Binary	
	AUC	SRCC	AUC	SRCC		AUC	SRCC	AUC	SRCC
A*01:01	0.93	0.57	0.94	0.55	B*15:09	0.88	0.47	0.89	0.47
A*02:01	0.95	0.81	0.95	0.81	B*15:17	0.93	0.69	0.94	0.69
A*02:02	0.93	0.86	0.93	0.86	B*15:42	0.82	0.04	0.65	-0.01
A*02:03	0.95	0.82	0.95	0.82	B*18:01	0.90	0.52	0.89	0.48
A*02:04	N/A	N/A	N/A	N/A	B*27:01	0.00	-1.00	0.00	-1.00
A*02:05	0.99	0.84	0.98	0.81	B*27:02	N/A	-0.50	N/A	-0.50
A*02:06	0.91	0.80	0.91	0.79	B*27:03	N/A	0.03	N/A	-0.01
A*02:07	0.76	0.75	0.79	0.79	B*27:04	N/A	1.00	N/A	1.00
A*02:10	N/A	N/A	N/A	N/A	B*27:05	0.94	0.63	0.94	0.59
A*02:11	0.96	0.80	0.96	0.80	B*27:06	N/A	-1.00	N/A	-1.00
A*02:12	0.97	0.77	0.97	0.77	B*27:10	N/A	N/A	N/A	N/A
A*02:16	0.98	0.68	0.98	0.68	B*27:20	0.56	0.52	0.54	0.56
A*02:17	0.68	0.41	0.69	0.41	B*35:01	0.90	0.72	0.91	0.71
A*02:19	0.96	0.66	0.96	0.66	B*35:03	0.87	0.50	0.90	0.54
A*02:50	1.00	0.88	0.99	0.87	B*37:01	0.48	0.01	0.61	0.13
A*03:01	0.93	0.73	0.93	0.72	B*38:01	0.98	0.81	0.98	0.80
A*03:02	0.69	0.57	0.68	0.57	B*39:01	0.93	0.62	0.93	0.61
A*03:19	0.87	0.55	0.87	0.56	B*40:01	0.97	0.63	0.97	0.59
A*11:01	0.95	0.77	0.95	0.76	B*40:02	0.90	0.76	0.90	0.75
A*11:02	1.00	0.77	1.00	0.77	B*40:13	0.64	0.48	0.61	0.46
A*23:01	0.93	0.71	0.92	0.68	B*42:01	0.94	0.77	0.94	0.77
A*24:02	0.90	0.67	0.90	0.66	B*42:02	0.84	0.64	0.82	0.63
A*24:03	0.96	0.70	0.96	0.68	B*44:02	0.95	0.60	0.94	0.55
A*25:01	0.98	0.47	0.99	0.47	B*44:03	0.94	0.82	0.95	0.82
A*26:01	0.93	0.52	0.93	0.49	B*45:01	0.92	0.67	0.91	0.66
A*26:02	0.96	0.74	0.96	0.74	B*45:06	0.81	0.14	0.71	0.17
A*26:03	0.93	0.52	0.94	0.54	B*46:01	0.92	0.44	0.93	0.44
A*29:02	0.88	0.65	0.87	0.63	B*48:01	0.92	0.49	0.92	0.50
A*30:01	0.91	0.70	0.92	0.70	B*51:01	0.92	0.58	0.92	0.55
A*30:02	0.82	0.63	0.82	0.62	B*52:01	0.58	0.29	0.50	0.18
A*31:01	0.93	0.74	0.93	0.74	B*53:01	0.92	0.78	0.93	0.77
A*32:01	0.85	0.71	0.85	0.72	B*54:01	0.90	0.73	0.90	0.73
A*32:07	0.81	0.53	0.82	0.53	B*57:01	0.96	0.62	0.96	0.59
A*32:15	0.51	0.38	0.50	0.36	B*57:02	0.84	0.66	0.80	0.62
A*33:01	0.92	0.73	0.92	0.73	B*57:03	0.97	0.74	0.97	0.75
A*66:01	0.83	0.41	0.84	0.35	B*58:01	0.95	0.69	0.96	0.68
A*68:01	0.90	0.79	0.90	0.79	B*58:02	0.55	0.51	0.59	0.55
A*68:02	0.92	0.70	0.92	0.69	B*73:01	0.64	0.36	0.66	0.37
A*68:23	0.77	0.53	0.78	0.54	B*81:01	0.92	0.76	0.93	0.76
A*69:01	0.94	0.51	0.94	0.53	B*83:01	0.93	0.53	0.93	0.53
A*74:01	0.70	0.45	0.61	0.33	C*03:03	0.80	0.59	0.79	0.58
A*80:01	0.94	0.55	0.95	0.55	C*04:01	0.52	-0.09	0.58	-0.15
B*07:02	0.95	0.72	0.95	0.72	C*05:01	0.91	0.74	0.92	0.75
B*08:01	0.91	0.66	0.91	0.65	C*06:02	0.89	0.74	0.89	0.74
B*08:02	0.96	0.42	0.96	0.44	C*07:01	0.84	0.61	0.84	0.61
B*08:03	0.92	0.35	0.95	0.38	C*07:02	0.74	0.42	0.73	0.41
B*14:01	0.74	0.43	0.74	0.44	C*08:02	0.64	0.28	0.63	0.25
B*14:02	0.84	0.51	0.85	0.43	C*12:03	0.62	0.28	0.61	0.29
B*15:01	0.90	0.67	0.90	0.67	C*14:02	0.67	0.28	0.67	0.29
B*15:02	0.77	0.53	0.77	0.53	C*15:02	0.79	0.52	0.80	0.55
B*15:03	0.90	0.75	0.90	0.75					

Table 4.3 Evaluation results of Kim's DCNN and DeepSeqPan

MHC	IEDB Ref	Measure Type	Count	AUC	
				Kim	DeepSeqPan (Binary)
A*01-01	1028282	t1/2	6	1.00	1.00
A*02-01	1026371	t1/2	34	0.70	0.76
A*02-01	1026840	Binary	341	0.85	0.85
A*02-01	1026840	IC_{50}	22	0.79	0.55
A*02-01	1026840	t1/2	22	0.69	0.58
A*02-01	1027079	Binary	15	0.80	0.80
A*02-01	1027471	Binary	43	0.79	0.88
A*02-01	1027588	Binary	18	0.70	0.82
A*02-01	1028285	t1/2	135	0.75	0.72
A*02-01	1028553	IC_{50}	22	0.85	0.95
A*02-01	1028554	IC_{50}	44	0.75	0.91
A*02-01	1028928	Binary	11	0.92	0.94
A*02-01	1029824	Binary	77	0.59	0.58
A*03-01	1028288	t1/2	221	0.84	0.85
A*03-01	1031253	IC_{50}	14	0.96	1.00
A*11-01	1026891	Binary	16	0.71	0.67
A*11-01	1028287	t1/2	219	0.79	0.76
A*24-02	1026840	Binary	346	0.84	0.83
A*24-02	1026840	IC_{50}	19	0.62	0.61
A*24-02	1026891	Binary	19	0.55	0.68
A*24-02	1028289	t1/2	423	0.73	0.74
A*30-01	1026840	Binary	347	0.87	0.85
A*30-02	1026840	Binary	360	0.73	0.72
A*30-02	1026840	IC_{50}	56	0.51	0.55
A*30-02	1026840	t1/2	56	0.48	0.49
A*31-01	315312	Binary	8	0.94	0.88
A*66-01	315312	Binary	16	0.39	0.14
A*68-01	1026840	Binary	436	0.86	0.84
A*68-01	1026840	IC_{50}	35	0.84	0.78
A*68-01	1026840	t1/2	35	0.42	0.33
B*07-02	1026371	t1/2	33	0.89	0.91
B*07-02	1026840	Binary	288	0.86	0.81
B*07-02	1028291	t1/2	136	0.82	0.82
B*07-02	1028553	IC_{50}	22	0.84	0.90
B*07-02	1028554	IC_{50}	52	0.80	0.80
B*07-02	1028928	Binary	11	1.00	1.00
B*07-02	1031253	IC_{50}	13	1.00	1.00
B*15-01	1028293	t1/2	570	0.74	0.62
B*15-02	1027131	Binary	14	1.00	1.00
B*27-05	1029125	Binary	21	0.97	0.95
B*27-05	1031253	IC_{50}	12	0.60	0.63
B*35-01	1028292	t1/2	363	0.81	0.74
B*35-01	1028554	IC_{50}	56	0.47	0.58
B*40-01	1026891	Binary	19	0.83	0.81
B*40-01	1026897	Binary	15	0.80	0.80
B*44-03	1028554	IC_{50}	46	0.54	0.80
B*57-01	1028554	IC_{50}	53	0.87	0.89
B*57-01	1029061	IC_{50}	17	0.90	0.95
B*58-01	1026840	Binary	433	0.87	0.85
B*58-01	1026840	IC_{50}	34	0.77	0.53
B*58-01	1026840	t1/2	34	0.44	0.58
B*58-01	1026891	Binary	20	0.66	0.67
B*58-01	1026897	Binary	22	0.81	0.90
B*27-05	1031959	Binary	13540	0.60	0.60

Table 4.4 Comparision results of 3-fold blind testing and 5-fold corss validations on each alleles in trainign data with CD-HIT filtered. Highlited cells are higher scores in that record.

Allele	5-fold cross validation				3-fold blind test				Allele	5-fold cross validation				3-fold blind test			
	IC50		AUC		IC50		AUC			IC50		AUC		IC50		AUC	
	AUC	SRCC	AUC	SRCC	AUC	SRCC	AUC	SRCC		AUC	SRCC	AUC	SRCC	AUC	SRCC	AUC	SRCC
A*01:01	1.00	0.57	1.00	0.55	0.55	0.13	0.55	0.10	B*15:09	1.00	0.47	0.99	0.49	0.37	-0.11	0.37	-0.10
A*02:01	1.00	0.81	1.00	0.80	0.68	0.35	0.72	0.41	B*15:17	1.00	0.69	1.00	0.69	0.73	0.38	0.73	0.39
A*02:02	1.00	0.85	1.00	0.85	0.71	0.43	0.72	0.46	B*15:42	1.00	0.06	1.00	0.00	0.59	0.00	0.60	0.01
A*02:03	1.00	0.80	1.00	0.79	0.69	0.35	0.71	0.40	B*18:01	1.00	0.50	1.00	0.46	0.45	0.05	0.44	0.01
A*02:04	-	-	-	-	-	-	-	-	B*27:01	-	-	-	-	-	-	-	-
A*02:05	1.00	0.85	1.00	0.81	0.79	0.47	0.86	0.55	B*27:02	1.00	0.60	1.00	0.60	-	0.80	-	0.80
A*02:06	1.00	0.79	1.00	0.77	0.67	0.34	0.70	0.40	B*27:03	1.00	-0.05	1.00	-0.01	-	-0.01	-	-0.01
A*02:07	1.00	0.79	1.00	0.81	0.73	0.08	0.77	0.17	B*27:04	1.00	1.00	1.00	1.00	-	-0.50	-	-0.50
A*02:10	-	-	-	-	-	-	-	-	B*27:05	1.00	0.60	1.00	0.57	0.55	0.12	0.50	0.06
A*02:11	1.00	0.80	1.00	0.80	0.65	0.26	0.69	0.32	B*27:06	-	1.00	-	1.00	-	-1.00	-	-1.00
A*02:12	1.00	0.77	1.00	0.77	0.67	0.24	0.71	0.31	B*27:10	-	-	-	-	-	-	-	-
A*02:16	1.00	0.68	1.00	0.69	0.60	0.11	0.65	0.17	B*27:20	1.00	0.50	1.00	0.47	0.36	-0.19	0.38	-0.29
A*02:17	1.00	0.38	1.00	0.38	0.61	0.26	0.61	0.27	B*35:01	1.00	0.69	1.00	0.68	0.58	0.13	0.60	0.16
A*02:19	1.00	0.65	1.00	0.65	0.65	0.20	0.68	0.22	B*35:03	1.00	0.50	1.00	0.54	0.48	-0.15	0.51	-0.10
A*02:50	1.00	0.86	1.00	0.87	0.44	-0.05	0.44	-0.06	B*37:01	1.00	0.37	1.00	0.25	0.53	-0.02	0.53	0.06
A*03:01	1.00	0.72	1.00	0.71	0.45	-0.08	0.42	-0.13	B*38:01	1.00	0.59	1.00	0.59	0.20	-0.29	0.24	-0.27
A*03:02	1.00	0.54	1.00	0.53	0.61	0.44	0.67	0.40	B*39:01	1.00	0.62	1.00	0.62	0.42	-0.09	0.42	-0.10
A*03:19	1.00	0.20	1.00	0.23	0.57	0.13	0.52	0.10	B*40:01	1.00	0.59	1.00	0.55	0.40	-0.03	0.40	-0.05
A*11:01	1.00	0.77	1.00	0.75	0.43	-0.09	0.42	-0.14	B*40:02	1.00	0.73	1.00	0.69	0.54	0.04	0.53	0.02
A*11:02	1.00	0.79	0.96	0.94	0.52	0.13	0.33	-0.19	B*40:13	1.00	0.56	1.00	0.56	0.44	-0.07	0.43	-0.10
A*23:01	1.00	0.68	1.00	0.64	0.60	0.21	0.61	0.22	B*42:01	1.00	0.83	1.00	0.83	0.41	-0.20	0.38	-0.21
A*24:02	1.00	0.67	1.00	0.66	0.61	0.20	0.62	0.21	B*42:02	1.00	0.43	1.00	0.41	0.97	0.48	0.91	0.35
A*24:03	1.00	0.67	1.00	0.65	0.67	0.29	0.68	0.30	B*44:02	1.00	0.56	1.00	0.51	0.36	0.02	0.37	0.01
A*25:01	1.00	0.48	1.00	0.48	0.58	0.10	0.62	0.14	B*44:03	1.00	0.79	0.99	0.79	0.45	-0.03	0.43	-0.06
A*26:01	1.00	0.50	1.00	0.47	0.64	0.19	0.66	0.18	B*45:01	1.00	0.65	0.99	0.63	0.49	-0.06	0.47	-0.09
A*26:02	1.00	0.76	1.00	0.75	0.61	0.20	0.62	0.20	B*45:06	1.00	0.09	1.00	0.08	0.45	0.09	0.43	0.09
A*26:03	1.00	0.52	1.00	0.53	0.64	0.18	0.66	0.20	B*46:01	1.00	0.41	1.00	0.42	0.74	0.21	0.75	0.21
A*29:02	1.00	0.62	1.00	0.59	0.54	0.11	0.54	0.10	B*48:01	1.00	0.54	1.00	0.54	0.57	0.06	0.60	0.10
A*30:01	1.00	0.67	1.00	0.67	0.50	0.06	0.51	0.07	B*51:01	1.00	0.54	1.00	0.51	0.52	0.05	0.53	0.05
A*30:02	1.00	0.64	0.99	0.62	0.51	0.04	0.51	0.02	B*52:01	1.00	0.36	1.00	0.04	0.67	0.07	0.75	0.29
A*31:01	1.00	0.73	1.00	0.73	0.45	-0.09	0.45	-0.08	B*53:01	1.00	0.70	1.00	0.70	0.47	0.03	0.49	0.05
A*32:01	1.00	0.67	1.00	0.68	0.72	0.43	0.72	0.43	B*54:01	1.00	0.70	1.00	0.69	0.52	-0.04	0.52	-0.03
A*32:07	1.00	0.33	1.00	0.31	0.38	-0.08	0.41	-0.03	B*57:01	1.00	0.59	1.00	0.56	0.77	0.35	0.78	0.35
A*32:15	1.00	0.27	1.00	0.24	0.71	0.26	0.71	0.24	B*57:02	1.00	0.84	1.00	0.82	0.67	0.64	0.67	0.65
A*33:01	1.00	0.71	0.99	0.71	0.36	-0.18	0.38	-0.18	B*57:03	1.00	0.46	1.00	0.40	0.64	0.51	0.61	0.49
A*66:01	1.00	0.42	1.00	0.39	0.46	0.10	0.50	0.10	B*58:01	1.00	0.64	1.00	0.63	0.70	0.34	0.71	0.34
A*68:01	1.00	0.78	1.00	0.77	0.43	-0.15	0.43	-0.19	B*58:02	1.00	0.43	0.98	0.47	0.86	0.50	0.85	0.48
A*68:02	1.00	0.66	1.00	0.66	0.66	0.27	0.67	0.29	B*73:01	1.00	0.51	1.00	0.48	0.54	0.05	0.55	0.12
A*68:23	1.00	0.54	1.00	0.54	0.49	-0.07	0.49	-0.07	B*81:01	1.00	0.81	1.00	0.78	0.37	0.03	0.43	0.15
A*69:01	1.00	0.50	1.00	0.51	0.69	0.22	0.73	0.27	B*83:01	1.00	0.55	1.00	0.55	0.37	-0.18	0.39	-0.15
A*74:01	1.00	0.68	1.00	0.68	0.46	0.07	0.54	0.20	C*03:03	1.00	0.60	0.99	0.58	0.42	-0.10	0.43	-0.09
A*80:01	1.00	0.57	1.00	0.57	0.49	-0.06	0.50	-0.05	C*04:01	-	-0.06	-	-0.10	0.58	0.10	0.57	0.10
B*07:02	1.00	0.71	1.00	0.71	0.42	-0.08	0.42	-0.08	C*05:01	1.00	0.75	1.00	0.74	0.50	0.00	0.52	0.03
B*08:01	1.00	0.66	1.00	0.65	0.47	-0.02	0.47	-0.02	C*06:02	1.00	0.67	1.00	0.67	0.63	0.21	0.64	0.21
B*08:02	1.00	0.47	1.00	0.48	0.47	-0.07	0.49	-0.06	C*07:01	1.00	0.65	1.00	0.65	0.39	-0.29	0.39	-0.29
B*08:03	1.00	0.37	1.00	0.39	0.56	0.04	0.61	0.07	C*07:02	1.00	0.51	0.99	0.46	0.38	-0.15	0.38	-0.16
B*14:01	1.00	0.30	1.00	0.31	0.57	-0.13	0.55	-0.15	C*08:02	1.00	0.20	0.97	0.15	0.48	0.13	0.47	0.10
B*14:02	1.00	0.41	1.00	0.33	0.52	0.18	0.48	0.11	C*12:03	1.00	0.29	0.99	0.27	0.59	0.16	0.59	0.19
B*15:01	1.00	0.66	1.00	0.66	0.67	0.30	0.67	0.30	C*14:02	1.00	0.09	0.99	0.07	0.43	-0.02	0.45	-0.04
B*15:02	1.00	0.56	1.00	0.55	0.66	0.30	0.67	0.34	C*15:02	1.00	0.50	1.00	0.49	0.29	-0.42	0.28	-0.43
B*15:03	1.00	0.69	1.00	0.69	0.66	0.21	0.65	0.19									

Table 4.5 Consistency inspection results

	Cross Validation	Benchmark Evaluation
Total samples	121787	19741
Consistent pred.	116688 (95.81%)	17004 (86.14%)
Correct IC_{50} pred.	108064 (88.73%)	11690 (59.21%)
Correct Binary pred.	107239 (88.05%)	10487 (53.12%)

Table 4.6 Evaluation on benchmark database. Red/Yellow highlighted number(s) are best AUC/SRCC scores in that record. Sorted by IEDB Ref ID.

HLA	Type	Pan-specific								Allele-specific												Ensemble			
		DeepSeqPan		NetMHCpan 2.8		NetMHCpan 3.0		PickPocket		SMM		NetMHC 3.4		NetMHC 4.0		ARB		SMMPMBEC		MHCflurry		IEDB Consensus		NetMHCcons	
		AUC	SRCC	AUC	SRCC	AUC	SRCC	AUC	SRCC	AUC	SRCC	AUC	SRCC	AUC	SRCC	AUC	SRCC	AUC	SRCC	AUC	SRCC	AUC	SRCC	AUC	SRCC
B*27-03	Binary	0.58	0.14	0.92	0.71	-	-	0.88	0.64	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.92	0.71
A*66-01	Binary	0.11	-0.45	-	-	0.54	0.04	0.68	0.20	0.68	0.20	-	-	0.25	-0.29	-	-	1.00	0.57	0.86	0.41	0.64	0.16	0.61	0.12
A*31-01	Binary	0.88	0.65	-	-	0.88	0.65	0.88	0.65	0.88	0.65	-	-	0.88	0.65	-	-	0.88	0.65	0.88	0.65	0.88	0.65	0.88	0.65
B*07-02	t1/2	0.93	0.66	0.93	0.81	-	-	-	-	0.93	0.72	0.93	0.77	-	-	0.72	0.44	-	-	-	-	-	-	-	-
A*02-01	t1/2	0.75	0.41	0.75	0.42	-	-	-	-	0.77	0.45	0.76	0.45	-	-	0.75	0.45	-	-	-	-	-	-	-	-
B*07-02	Binary	0.83	0.29	0.87	0.32	-	-	-	-	0.88	0.33	0.88	0.33	-	-	0.85	0.31	-	-	-	-	-	-	-	-
C*07-01	Binary	0.67	0.15	0.78	0.25	-	-	-	-	0.65	0.13	0.76	0.23	-	-	-	-	-	-	-	-	-	-	-	-
C*07-01	IC_{50}	0.32	-0.29	0.54	-0.18	-	-	-	-	0.39	-0.01	0.61	0.17	-	-	-	-	-	-	-	-	-	-	-	-
A*30-01	Binary	0.83	0.17	0.81	0.16	-	-	-	-	0.79	0.15	0.77	0.14	-	-	0.71	0.11	-	-	-	-	-	-	-	-
B*58-01	Binary	0.84	0.36	0.86	0.38	-	-	-	-	0.87	0.40	0.85	0.37	-	-	0.84	0.36	-	-	-	-	-	-	-	-
B*58-01	IC_{50}	0.52	0.19	0.72	0.38	-	-	-	-	0.70	0.38	0.68	0.33	-	-	0.57	0.26	-	-	-	-	-	-	-	-
B*58-01	t1/2	0.56	0.03	0.54	0.15	-	-	-	-	0.53	0.08	0.54	0.16	-	-	0.50	0.12	-	-	-	-	-	-	-	-
A*30-02	Binary	0.72	0.35	0.77	0.42	-	-	-	-	0.73	0.36	0.75	0.40	-	-	0.65	0.25	-	-	-	-	-	-	-	-
A*30-02	IC_{50}	0.64	0.21	0.47	0.01	-	-	-	-	0.54	0.12	0.59	0.13	-	-	0.64	0.27	-	-	-	-	-	-	-	-
A*30-02	t1/2	0.43	0.15	0.50	0.05	-	-	-	-	0.50	0.07	0.55	0.19	-	-	0.52	0.15	-	-	-	-	-	-	-	-
A*68-01	Binary	0.82	0.33	0.87	0.38	-	-	-	-	0.86	0.37	0.87	0.38	-	-	0.78	0.34	-	-	-	-	-	-	-	-
A*68-01	IC_{50}	0.76	0.44	0.84	0.63	-	-	-	-	0.79	0.62	0.84	0.65	-	-	0.77	0.53	-	-	-	-	-	-	-	-
A*68-01	t1/2	0.40	-0.13	0.32	-0.32	-	-	-	-	0.25	-0.42	0.27	-0.41	-	-	0.31	-0.39	-	-	-	-	-	-	-	-
A*24-02	Binary	0.81	0.34	0.85	0.39	-	-	-	-	0.82	0.36	0.85	0.38	-	-	0.83	0.37	-	-	-	-	-	-	-	-
A*24-02	IC_{50}	0.54	0.16	0.63	0.18	-	-	-	-	0.76	0.39	0.61	0.19	-	-	0.45	0.03	-	-	-	-	-	-	-	-
A*02-01	Binary	0.82	0.42	0.89	0.52	-	-	-	-	0.88	0.51	0.87	0.49	-	-	0.88	0.51	-	-	-	-	-	-	-	-
A*02-01	IC_{50}	0.69	0.31	0.72	0.36	-	-	-	-	0.68	0.37	0.63	0.29	-	-	0.74	0.45	-	-	-	-	-	-	-	-
A*02-01	t1/2	0.74	0.33	0.73	0.43	-	-	-	-	0.74	0.37	0.67	0.29	-	-	0.69	0.46	-	-	-	-	-	-	-	-
C*07-02	Binary	0.60	0.17	0.65	0.25	-	-	-	-	0.74	0.39	0.75	0.41	-	-	-	-	-	-	-	-	-	-	-	-
A*11-01	Binary	0.21	-0.33	0.43	-0.08	-	-	-	-	0.50	0.00	0.46	-0.04	-	-	0.14	-0.41	-	-	-	-	-	-	-	-
B*58-01	Binary	0.59	0.16	0.88	0.64	-	-	-	-	0.76	0.44	0.88	0.64	-	-	0.84	0.58	-	-	-	-	-	-	-	-
B*40-01	Binary	0.81	0.54	0.92	0.73	-	-	-	-	0.86	0.62	0.91	0.71	-	-	0.87	0.64	-	-	-	-	-	-	-	-
A*24-02	Binary	0.50	0.00	0.63	0.19	-	-	-	-	0.57	0.09	0.53	0.05	-	-	0.55	0.07	-	-	-	-	-	-	-	-
B*58-01	Binary	0.77	0.32	0.74	0.28	-	-	-	-	0.81	0.37	0.75	0.30	-	-	0.67	0.20	-	-	-	-	-	-	-	-
B*40-01	Binary	0.82	0.49	0.75	0.38	-	-	-	-	0.84	0.52	0.75	0.38	-	-	0.75	0.38	-	-	-	-	-	-	-	-
A*02-01	Binary	0.77	0.46	0.77	0.46	-	-	-	-	0.68	0.31	0.75	0.43	-	-	0.68	0.31	-	-	-	-	-	-	-	-
B*15-02	Binary	1.00	0.71	1.00	0.71	-	-	1.00	0.71	0.91	0.58	1.00	0.71	-	-	1.00	0.72	1.00	0.71	1.00	0.71	1.00	0.71	1.00	0.71
A*02-01	Binary	0.94	0.49	0.84	0.38	-	-	-	-	0.83	0.36	0.81	0.34	-	-	0.81	0.35	-	-	-	-	-	-	-	-
A*02-01	Binary	0.87	0.63	0.83	0.56	-	-	-	-	0.84	0.58	0.82	0.55	-	-	0.75	0.43	-	-	-	-	-	-	-	-
C*03-03	IC_{50}	1.00	0.77	0.68	0.68	-	-	0.64	0.47	0.68	0.46	0.68	0.46	-	-	-	-	0.76	0.67	-	-	0.68	0.46	0.68	0.56
A*01-01	t1/2	1.00	0.84	1.00	0.90	-	-	1.00	0.90	1.00	0.81	1.00	0.90	-	-	1.00	0.75	1.00	0.90	-	-	1.00	0.79	1.00	0.90
A*02-01	t1/2	0.78	0.49	0.76	0.52	-	-	0.82	0.59	0.79	0.56	0.74	0.51	-	-	0.79	0.49	0.78	0.55	-	-	0.78	0.56	0.76	0.53
A*11-01	t1/2	0.80	0.58	0.90	0.76	-	-	0.80	0.51	0.90	0.75	0.90	0.74	-	-	0.87	0.67	0.90	0.75	-	-	0.91	0.76	0.91	0.76
A*03-01	t1/2	0.81	0.50	0.85	0.58	-	-	0.77	0.48	0.86	0.56	0.85	0.57	-	-	0.85	0.54	0.86	0.57	-	-	0.86	0.58	0.86	0.59
A*24-02	t1/2	0.74	0.48	0.79	0.56	-	-	0.81	0.60	0.80	0.57	0.78	0.57	-	-	0.76	0.50	0.80	0.58	-	-	0.80	0.59	0.80	0.60
A*26-01	t1/2	-	-0.5	-	-0.5	-	-	-	0.50	-	0.50	-	-0.5	-	-	-	0.50	-	0.50	-	-	-	0.50	-	-0.50
B*07-02	t1/2	0.83	0.67	0.89	0.78	-	-	0.81	0.69	0.88	0.76	0.87	0.76	-	-	0.80	0.63	0.88	0.76	-	-	0.88	0.76	0.89	0.78
B*35-01	t1/2	0.79	0.52	0.83	0.62	-	-	0.83	0.60	0.81	0.58	0.83	0.62	-	-	0.74	0.45	0.81	0.58	-	-	0.82	0.59	0.84	0.64
B*15-01	t1/2	0.59	0.19	0.74	0.43	-	-	0.61	0.22	0.71	0.40	-	-	-	-	0.59	0.20	0.71	0.40	-	-	0.70	0.38	0.74	0.43
B*40-01	t1/2	-	0.98	-	0.96	-	-	-	0.93	-	0.86	-	0.98	-	-	-	0.73	-	0.93	-	-	-	-	-	0.98
B*07-02	IC_{50}	0.89	0.81	0.89	0.66	-	-	0.89	0.62	0.89	0.62	0.92	0.76	-	-	0.88	0.55	0.88	0.62	-	-	0.92	0.72	0.91	0.72
A*02-01	IC_{50}	0.88	0.52	0.92	0.62	-	-	0.83	0.58	0.85	0.59	0.91	0.70	-	-	0.90	0.69	0.83	0.62	-	-	0.84	0.62	0.92	0.64
B*07-02	IC_{50}	0.71	0.61	0.77	0.62	-	-	0.74	0.70	0.85	0.66	0.88	0.70	-	-	0.76	0.65	0.86	0.70	-	-	0.88	0.70	0.86	0.73
B*57-01	IC_{50}	0.80	0.43	0.86	0.62	-	-	0.85	0.44	0.77	0.33	0.94	0.52	-	-	0.63	0.12	0.77	0.29	-	-	0.87	0.43	0.92	0.56
B*35-01	IC_{50}	0.64	0.20	0.68	0.36	-	-	0.50	0.28	0.59	0.21	0.57	0.27	-	-	0.64	0.26	0.53	0.20	-	-	0.65	0.30	0.64	0.36
B*44-03	IC_{50}	0.77	0.31	0.61	0.46	-	-	0.64	0.43	0.75	0.47	0.65	0.56	-	-	0.56	0.25	0.76	0.55	-	-	0.64	0.51	0.64	0.55
A*02-01	IC_{50}	0.75	0.44	0.89	0.70	-	-	0.88	0.51	0.89	0.58	0.82	0.62	-	-	0.76	0.51	0.86	0.55	-	-	0.84	0.58	0.89	0.69
B*07-02	Binary	1.00	0.50	1.00	0.50	-	-	1.00	0.50	1.00	0.50	1.00	0.50	-	-	1.00	0.50	1.00	0.50	-	-	1.00	0.52	1.00	0.50
A*02-01	Binary	1.00	0.67	0.94	0.60	-	-	0.92	0.56	0.94	0.60	0.94	0.60	-	-	0.94	0.60	0.94	0.60	-	-	0.94	0.60	0.94	0.60
B*57-01	IC_{50}	0.83	0.63	0.88	0.60	-	-	0.86	0.72	0.88	0.64	0.81	0.64	-	-	0.40	0.12	0.81	0.65	0.90	0.67	0.83	0.64	0.83	0.57
B*27-04	Binary	0.89	0.63	0.94	0.72	-	-	0.83	0.53	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.94	0.72
B*27-06	Binary	0.75	0.42	0.75	0.42	-	-	0.80	0.50	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.75	0.42
B*27-05	Binary	0.89	0.63	0.96	0.75	-	-	0.91	0.67	0.91	0.67	0.94	0.72	-	-	0.88	0.62	0.96	0.75	0.89	0.63	0.91	0.68	0.96	0.75
A*02-01	Binary	0.55	0.07	0.55	0.07	-	-	0.55	0.08	0.57	0.10	0.57	0.11	-	-	0.52	0.04	0.57	0.10	0.57	0.11	0.57	0.11	0.56	0.09
B*38-01	IC_{50}	-	-0.70	-	0.00	-	-	-	-0.10	-	-0.10	-	-0.10	-	-	-	-0.50	-	-0.10	-	0.30	-	-0.10	-	0.00
B*07-02	IC_{50}	1.00	0.93	-	-	1.00	0.97	1.00	0.87	1.00	0.96	-	-	1.00	0.98	-	-	1.00	0.97	1.00	0.91	1.00	0.97	1.00	0.97
A*03-01	IC_{50}	0.93	0.87	-	-	0.93	0.78	0.96	0.71	0.87	0.69	-	-	0.96	0.79	-	-	0.91	0.75	0.76	0.56	0.91	0.74	0.91	0.76
B*27-05	IC_{50}	0.77	0.67	-	-	0.63	0.59	0.60	0.49	0.63	0.44	-	-	0.69	0.58	-	-	0.60	0.46	0.57	0.48	0.63	0.49	0.66	0.62
B*27-05	Binary	0.60	0.16	-	-	0.60	0.16	0.60	0.16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.60	0.16
Highlighted		19	13	23	19	3	2	10	7	16	10	16	14	3	2	4	6	8	5	5	3	10	5	16	15

Chapter 5

Attention-like LSTM-CNN network on peptide-HLA class II binding affinity and binding core prediction

5.1 Introduction

As we introduced before, besides HLA class I, another import subgroup of HLA is HLA class II. The HLA class II are also heterodimers, like HLA class I. But in class II, both α and β chains are encoded by HLA genes and are interaction with binding peptides. Figure 5.1 illustrates the structure of HLA class II. The α_1 and β_1 regions of the chains come together to make a membrane-distal peptide-binding domain [103]. The peptide binding groove, is made up of two α-helixes walls and β-sheet [34]. Since the binding groove of HLA class II is open at both ends (while class I is closed at both ends), the peptides binding to HLA class II are longer, between 15 and 24 amino acid residues long [53].

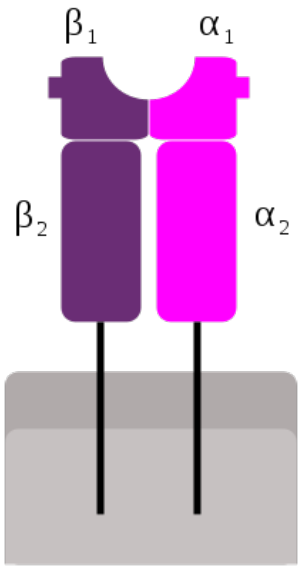


Figure 5.1 Schematic representation of HLA class II. Reproduced from wikipedia

Comparing with HLA class I, there are much less methods proposed on peptide-HLA class II binding prediction, both allele-specific and pan-specific. And the performance of those models are considerably inferior to that of HLA class I methods [4]. There are several reasons make the class II prediction model more challenging:

- The peptides binding to HLA class II are longer and thus have more dynamic lengths. This requires models have a capability to encode variant lengths input sequences.

- In HLA class I, an HLA allele has only one protein sequence and different alleles have same lengths. But in class II, an HLA allele has two protein sequences and each sequence may have very variant length.

- As we know the peptides binding to HLA class II are much longer. However, only a small part (usually 9-length amino acid residues, called *binding core*) of it will be fitted into the binding groove [110]. Given a peptide, researchers are also interested in the binding core prediction besides affinity prediction. Currently, only few methods offer identification of the binding core given a peptide.

In this chapter, we propose an attention LSTM model addressing on the peptide-HLA class II binding prediction problem. Our model is a pan-specific model which could give both affinity prediction and binding core prediction. In Section 5.2, we first introduce related works in this area. In next Section 5.2, we give the details of our proposed model. We compare our method on benchmark dataset and show the result in Section 5.4. Finally, we summarize our work in Section 5.5.

5.2 Related work

Like prediction models in HLA class I, existing models on class II binding prediction can be grouped into either allele-specific or pan-specific. NN-align [69], TEPITOPE [91], ARB [12], SVRMHC [100] and MHCpred [21] are some allele-specific methods. As for pan-specific methods, TEPITOPEpan [109] and NetMHCIIpan [70] are currently available models.

ARB, as a quantitative statistical method, uses a derived Average Relative Binding (ARB) matrix that directly predict IC50 values allowing combination of searches

involving different peptide sizes and alleles into a single global prediction. In ARB, the author assumes individual side chains of a peptide's amino acids contribute to binding independently. If a residue r appears at position i in a peptide, it's assumed to contribute a constant amount of $E_{i,r}$ to the free energy of binding of this peptide. $E_{i,r}$ was estimated as:

$$E_{i,r} = 10^{(P_{i,r} - Q_{i,r})} \tag{5.1}$$

where $P_{i,r}$ is the geometric average binding affinity of peptides containing residue r at position i and $Q_{i,r}$ is the geometric average binding affinity (IC_{50}) of the remaining peptides [12]. Given a training dataset, an ARB matrix could be derived. With this ARB matrix, the binding affinity score of a peptide is estimated through:

$$S = \prod_{i=1}^{L} E_{i,r} \tag{5.2}$$

In above equation. L is the length of this peptide.

In SVRMHC, the author developed a support vector machine regression (SVR) model to predict binding affinities. For peptide sequence encoding, user tried one-hot encoding and 11-dimension feature vector introduced in [54]. On exploring different configuration combinations of SVR kernels and encoding method, it chose the best performance model as the final model [100].

MHCpred introduced a partial least square (PLS) based additive model. An additive model assumes that the binding affinity of a peptide could be presented as a sum of the contributions of the amino acids at each position and certain interactions between them [21]. In MHCpred, each peptide was represented by a binary bit string of 180 bins (9 positions $\times$ 20 amino acids). One-hot encoding was used to encode each amino acid of a peptide. After construction of the matrix, an iterative self-consistent PLS-based algorithm was used to fit a scoring function.

A vanilla feed-forward artificial neural network was introduced in NN-align. It could give predictions of both binding cores and binding affinities. In NN-align,

each peptide was encoded with Blosum matrix. During training, a peptide will be processed to generate all possible 9-length subsequences. The core of a given peptide was identified as the highest scoring of all 9 mers contained within the peptide. Given a binding core, the weights were updated to lower the sum of squared errors between the predicted binding score and the measured binding affinity target value [69]. An ensemble method was employee by NN-align in which a group of neural networks with 2, 10, 20, 40 and 60 hidden neurons was built. And for each type of network, 10 duplicates with different starting weights were trained. In total, 50 networks were created. The author chose 10 networks with highest performance to assemble the final prediction model. The binding core of a given peptide was assigned by a majority vote of the networks in the ensemble [69].

TEPITOPEpan is a pan-specific method which utilized position-specific weight matrix (PSSM) and 3D structure to compute the binding affinities. It first generated pseudo sequences of all HLA-DR alleles based on 32 structures of HLA-peptide complexes. For each complex, it extracted the contacting residue positions on HLA alleles. Totally, 45 residue positions were extracted to represent an HLA allele. Then the pocket similarity and weight between any two alleles' pseudo sequences were calculated via method introduced in [108]. Finally, given any pair of a peptide and an HLA allele, the binding affinities was estimated via a weighted summary based on 11 available alleles' PSSMs [109].

Another pan-specific model, NetMHCIIpan also used pseudo sequence method to encode HLA alleles. Similar to TEPITOPEpan, the pseudo sequence were extracted from a series of 3D complexes. In NetMHCIIpan, the author extracted 34 amino acid residue locations to represent an allele: 15 from the α-chain and 19 from β-chain. The input sequences were encoded with three methods:(i) one-hot encoding, (ii) Blosum encoding, and (iii) a mixture of two. A group of networks with 22, 44, 56 and 66 hidden units were trained and the binding affinity were scored as the average of all

networks [70]. NetMHCIIpan preprocessed all peptides with alignment matrix and assigned each nonamer a normalized binding affinity. In this way, it could predict the binding core given a long peptide.

In this chapter, we propose a pan-specific model which only requires raw inputs of peptides and HLA alleles sequences. With the help of attention LSTM, each peptide-HLA sample only needs to be fed into the network one time to predict both affinity and binding core. In next section, we give the details of our proposed model.

5.3 Method

5.3.1 Attention LSTM

Initial long short-term memory (LSTM) model was proposed by Hochreiter and Schmidhuber [24]. The LSTM has been found extremely successful in many areas especially in natural language process [6, 20, 19]. A simplified LSTM cell is illustrated in Figure 5.2. The most critical part of the cell is the state unit which has a self-loop. By making the weight of this self-loop gated, the time scale of integration can be changed dynamically [18]. The forward process is calculated as follow:

$$i_t = \sigma(W_{ii}x_t + W_{if}h_{t-1} + b_i) \tag{5.3}$$

$$f_t = \sigma(W_{fi}x_t + W_{ff}h_{t-1} + b_f) \tag{5.4}$$

$$o_t = \sigma(W_{oi}x_t + W_{of}h_{t-1} + b_o) \tag{5.5}$$

$$c_t = f_t \bullet c_{t-1} + i_t \bullet tanh(W_{ci}x_t + W_{cf}h_{t-1} + b_c) \tag{5.6}$$

$$h_t = o_t \bullet tanh(c_t) \tag{5.7}$$

Even LSTM show its power in many NLP problems, researchers found that using a fixed-size representation, usually the last output of a LSTM model, to capture the semantic information of a very long sentence is very difficult [18]. One way to solve this problem is to train a sufficiently deep LSTM model and train it for a long time,

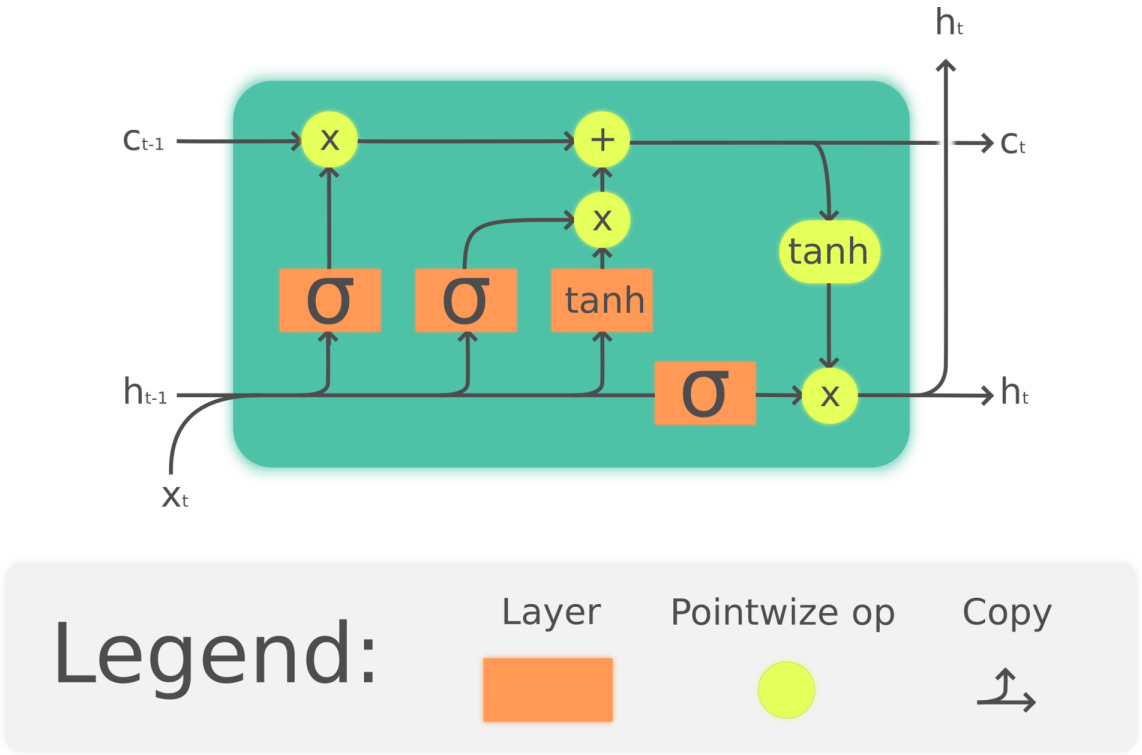
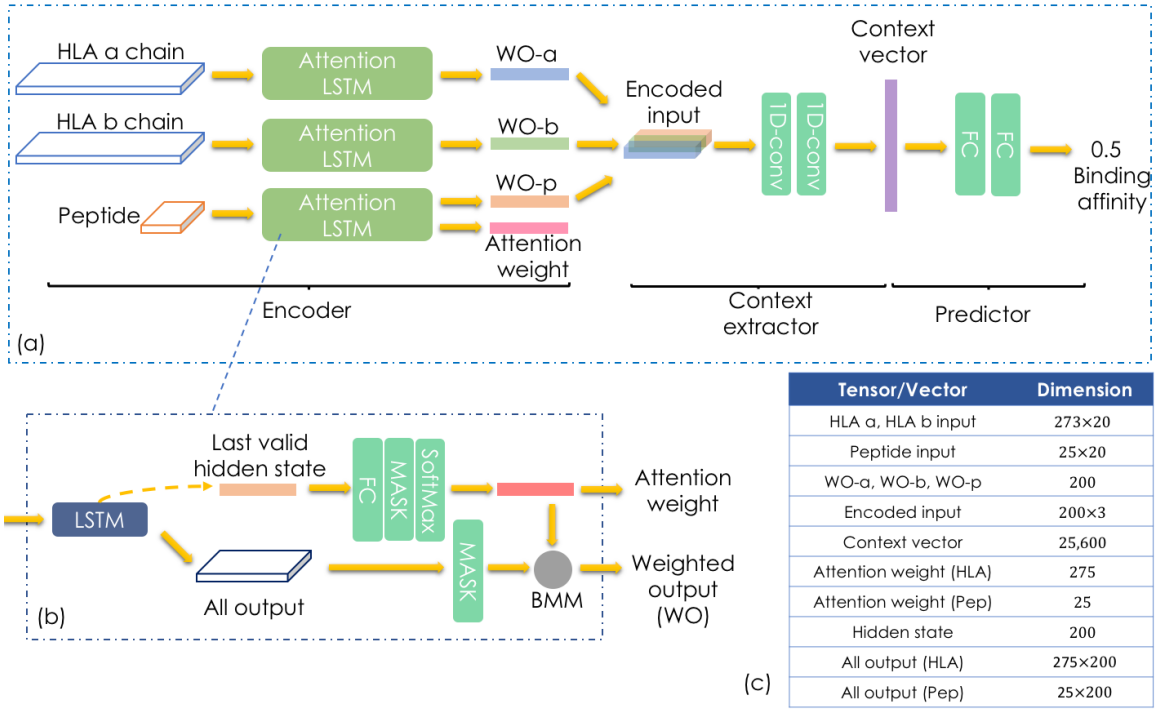


Figure 5.2 Simplified illustration of a LSTM cell. Reproduced from wikipedia

which has been demonstrated in [15, 7]. In 2015, Bahdanau et al. introduced attention mechanism on machine translation problem. In original attention mechanism, the LSTM reads the whole input first and then outputs the translated word one at a time while each time it focusing on a different part of the input. The model was designed such that it could automatically learn which part to focus. To achieve this goal, several implementations have been proposed to calculation the attention weight vector. Another benefit of attention LSTM is that by visualizing this attention weight vector, we could have a directly feeling about which part of the input the model is focusing on for the prediction.



Tensor/Vector	Dimension
HLA a, HLA b input	273×20
Peptide input	25×20
WO-a, WO-b, WO-p	200
Encoded input	200×3
Context vector	25,600
Attention weight (HLA)	275
Attention weight (Pep)	25
Hidden state	200
All output (HLA)	275×200
All output (Pep)	25×200

(c)

Figure 5.3 Architecture of our proposed model. (a) The overall structure. (b) The detail illustration of the attention LSTM module. (c) Dimensions of inputs and intermediate tensors/vectors.

5.3.2 Proposed method

We propose a attention-like LSTM model to predict the binding affinities and binding cores of peptide-HLA class II. Since our problem is a sequence-to-value problem, which is totally different from machine translation's sequence-to-sequence problem. We design a novel network architecture to calculate the peptide's attention weight vector from a LSTM module and output binding affinity in the final stage. The overall architecture of our proposed model is shown in Figure 5.3.

Sequence encoding

We still use one-hot encoding as we did in Chapter 3 and Chapter 5. Figure 4.3 illustrates the one-hot encoding. A peptide or protein sequence is encoded as a

tensor with shape: $1 \times L \times 20$. L is the max length of HLA sequences or peptide sequences. In our model, the max length of HLAs is 275 and that of peptides is 25. If a sequence's length is shorter than L, we pad 0 at the end.

Architecture

Our model consists of three modules as shown in Figure 5.3 (a):

- **Encoder.** A input of our model contains three parts: HLA α-chain sequence, HLA β-chain sequence and peptide sequence. Each sequence was encoded using one-hot encoding. For each input sequence, we construct an attention-like LSTM encoder of which the detail is shown in Figure 5.3 (b). We setup the LSTM with 100 hidden states and behave in a bi-directional way. As shown in the figure, we collect two outputs: the last hidden state and output values of all amino acids residues. The output tensor has dimension $L \times 200$ where L is the max length of HLA chains or peptides. The last hidden state (a 200-dim vector) is then fed into a fully connected (FC) layer with L hidden units where L is the max length of HLA chains or peptides. After FC layer, we apply SoftMax activation function on this L-length vector. In this way, we could obtain an attention weight vector with same length as input sequence and the sum of all values equal to 1. Here, we want to point out several processing we did: (i) we masked the all output by assigning all padding positions with 0 manually (shown as a MASK layer after All output in Figure 5.3 (b)), (ii) the hidden state we use was *valid* final hidden state, i.e. the hidden state of the last valid input instead of padded final input, and (iii) before applying SoftMax, we mask the output vector from FC layer by assigning all padding elements with 0 manually (shown as a MASK layer after FC layer in Figure 5.3 (b)). An example for (ii), if a peptide has 15 amino acids, it would be encoded as a 25×20 tensor. We use the 15th hidden state as the output hidden state instead of the 25th one

since the 15th one is the valid one. We did these processing because we want the LSTM only focus on valid input and ignore the padding noises. After we have the attention weight, we do a batch matrix-multiply (BMM) between all output and attention weight. Then we can get our weighted output with size L. For all three input sequences, after going through their encoders, we will get three weighted output vectors with same dimension 200 and three (we only plotted peptide's one in the figure) attention weight vectors of which two have dimension 273 for HLA chains and one has dimension 25 for peptide.

- **Context extractor** With three output vectors obtained from encoder, we first concatenate three vectors as a tensor with dimension 200×3. Then we feed this tensor into a CNN network consisting of two 1D convolutional layers. The first layer is configured with 256 filters of size 3 and the second CNN layer has 64 filters of size 3. For both two layers, we use LeakyRelu [106] as the activation function. Finally, we flatten the output of 2nd CNN layer as 25,600 dimension vector.

- **Predictor** In the last stage of our model, we use two vanilla fully connected layers to output the final IC_{50} value. First layer has 200 hidden units and second one has 1 hidden unit. We place dropout layer [90] between two FC layers. For the 2nd FC layer, we use tanh as the activation function.

Training Strategy

We fed the network with a batch of 512 samples and employed Early Stop to control the training. The samples are randomly split into training group and validation group following 9:1 ratio. We stopped training if validation loss hasn't improve in 5 continuous epochs. Mean squared error (MSE) was used as the loss function. For optimizer, we used basic stochastic gradient descent (SGD) algorithm. We implemented our model with PyTorch Framework [75].

5.3.3 Dataset

The training dataset is collected by [3]. This dataset contains about 130,000 training samples over 72 HLA class II alleles. The IC_{50} labels were normalized in the original dataset and all of them are in a scale of $0.0 \sim 1.0$. For testing, we use the standard weekly benchmark dataset [4] from 2016-12-31 to 2017-12-29 which can be accessible at `http://tools.iedb.org/auto_bench/mhcii/weekly/`. We group testing samples by its allele and measure type. Totally, we have 44 testing groups. For each group, we use area under the curve (AUC) and Spearman's rank correlation coefficient (SRCC) as metrics by following the convention.

5.4 Experiment

5.4.1 Evaluation on standard benchmark

We trained a model following the description in Section 5.3.2. Then we tested our model on the standard benchmark dataset which contains 44 testing groups. The evaluation results of other methods are included in the original benchmark dataset. We calculated AUC and SRCC for each testing group. The results are listed in Table 5.1. Our proposed method obtained highest AUC scores in 19 testing groups and highest SRCC scores in 17 testing groups. Another pan-specific method NetMHCIIpan-3.1 obtained highest AUC and SRCC scores in 18 and 20 testing groups, respectively. All other allele-specific methods and the ensemble method did not perform good enough to compare. Specially, we found that our method performs very well on alleles: HLA-DRA*01:01/DRB1*15:02, HLA-DRA*01:01/DRB1*13:01, HLA-DQA1*03:03/DQB1*04:02 and HLA-DQA1*01:02/DQB1*05:01.

The results showed that our proposed method could outperform existing methods on certain alleles. For the rest alleles, our model also could achieve competitive performance. This benchmark testing demonstrate that the model we proposed could

capture the complex pattern relationship between HLA chains and peptide based on one-hot encoded information. Since two pan-specific methods outperform each other on different alleles, it's intuitive to ensemble these two methods in practical usage.

5.4.2 Estimate the binding core

A benefit of our proposed model could directly estimate the binding core based on attention weight vector obtained from attention LSTM (as shown in Figure 5.3 (a)). We use Algorithm 1 to identify the binding core from attention weight vector. Basically, we find the 9-length subsequence with maximum attention weight and we think this subsequence is the binding core. Comparing with NetMHCIIpan which predicts on every possible 9-length subsequence of a peptide and select the one with highest predicted binding affinity as binding core, our model is a neat one-shot prediction.

Algorithm 1 Binding core identification algorithm

```
1: procedure BINDING_CORE(attnVec, L)          ▷ L: length of the peptide
2:     maxWeight ← 0
3:     core ← 0
4:     i ← 0
5:     while i < L − 9 + 1 do
6:         coreWeight ← ∑ attnVec[i : i + 9]
7:         if coreWeight > maxWeight then
8:             maxWeight ← coreWeight
9:             core ← i
10:    return core          ▷ Return the starting index of predicted binding core
```

To verify our model's binding core prediction capability, we tested our trained model obtained in Section 5.4.1 on 47 HLA class II - peptide complexes structures. These structures and observed binding cores were collected by [31]. The prediction results are shown in Figure 5.4. For each peptide, we underscored the observed binding core portion. The correctly predicted amino acids are green highlighted. For false positive predicted amino acids, they are red highlighted. And false negative predicted amino acids are blue highlighted. Of 47 records, 24 binding cores are

correctly predicted. In 10 structures, our predicted binding cores are either right or left offset with one amino acid. The predicted binding cores are either right or left offset with two amino acids in 9 records. Two predicted binding cores show 3 amino acids offset and two show 4 amino acids offset.

5.5 Summary

In this chapter, we proposed an attention-like LSTM-CNN model on class II HLA-peptide binding prediction problem. We tested it on the benchmark dataset and obtained state-of-the-art performance. Among 44 testing groups, we outperformed other methods on 19 groups for AUC and 17 groups for SRCC. Also, as a sequence based method, our method could be applied on all alleles with known sequences. With attention mechanism, our method could directly output the binding core prediction without feeding all possible 9-mers of a peptide. As currently the only non pseudo sequence based pan-specific method, we successfully applied attention LSTM network in this area and obtained one of the best performances. We argue that by ensemble with existing method, researchers could get much better prediction results.

PDB	Alpha	Beta	Peptide
1A6A	DRB1*03:01	DRB1*03:01	PVSKMRMATPLLMQA
1AQD	DRB1*01:01	DRB1*01:01	VGSDWRFLRGYHQYA
1BX2	DRB1*15:01	DRB1*15:01	ENPVVHFFKNIVTPR
1FV1	DRB5*01:01	DRB5*01:01	NPVVHFFKNIVTPRTPPPSQ
1FYT	DRB1*01:01	DRB1*01:01	PKYVKQNTLKLAT
1H15	DRB5*01:01	DRB5*01:01	GGVYHFVKKHVHES
1HQR	DRB5*01:01	DRB5*01:01	VHFFKNIVTPRTP
1J8H	DRB1*04:01	DRB1*04:01	PKYVKQNTLKLAT
1JK8	DQA1*03:03	DQB1*03:02	LVEALYLVCGERGG
1KLG	DRB1*01:01	DRB1*01:01	GELIGTLNAAKVPAD
1PYW	DRB1*01:01	DRB1*01:01	XFVKQNAAALX
1S9V	DQA1*05:05	DQB1*02:01	LQPFPQPELPY
1SJE	DRB1*01:01	DRB1*01:01	PEVIPMFSALSEGATP
1SJH	DRB1*01:01	DRB1*01:01	PEVIPMFSALSEG
1T5X	DRB1*01:01	DRB1*01:01	AAYSDQATPLLLSPR
1UVQ	DQA1*01:02	DQB1*06:02	MNLPSTKVSWAAVGGGGSLV
1YMM	DRB1*15:01	DRB1*15:01	ENPVVHFFKNIVTPRGGSGGGGG
1ZGL	DRB5*01:01	DRB5*01:01	VHFFKNIVTPRTPGG
2FSE	DRB1*01:01	DRB1*01:01	AGFKGEQGPKGEPG
2IPK	DRB1*01:01	DRB1*01:01	XPKVVKQNTLKLAT
2Q6W	DRB3*01:01	DRB3*01:01	AWRSDEALPLGS
2SEB	DRB1*04:01	DRB1*04:01	AYMRADAAAGGA
3C5J	DRB3*03:01	DRB3*03:01	QVIILNHPGQISA
3L6F	DRB1*01:01	DRB1*01:01	APPAYEKLSAEQSPP
3LQZ	DPA1*01:03	DPB1*02:01	RKFHYLPFLPSTGGS
3PDO	DRB1*01:01	DRB1*01:01	KPVSKMRMATPLLMQALPM
3PGD	DRB1*01:01	DRB1*01:01	KMRMATPLLMQALPM
3QXA	DRB1*01:01	DRB1*01:01	PVSKMRMATPLLMQA
3WEX	DPA1*02:01	DPB1*05:01	KVTVAFNQFGGS
4AEN	DRB1*01:01	DRB1*01:01	MPLAQMLLPTAMRMKM
4D8P	DQA1*03:01	DQB1*02:01	PQPEQPEQPFPQP
4GG6	DQA1*03:01	DQB1*03:02	QQYPSGEGSFQPSQENPQ
4H1L	DRB3*03:01	DRB3*03:01	QHIRCNIPKRISA
4H25	DRB3*03:01	DRB3*03:01	QHIRCNIPKRIGPSKVATLVPR
4H26	DRB3*03:01	DRB3*03:01	QWIRVNIPKRI
4I5B	DRB1*01:01	DRB1*01:01	VVKQNCLKLATK
4IS6	DRB1*04:01	DRB1*04:01	WNRQLYPEWTEAQRLD
4MCY	DRB1*04:01	DRB1*04:01	SAVRLRSSVPGVR
4MCZ	DRB1*04:01	DRB1*04:01	GVYATRSSAVRLR
4MD4	DRB1*04:01	DRB1*04:01	ATEYRVRVNSAYQDK
4MD5	DRB1*04:04	DRB1*04:04	SAVRLRSSVPGVR
4MDI	DRB1*04:02	DRB1*04:02	SAVRLRSSVPGVR
4OV5	DRB1*01:01	DRB1*01:01	GSDARFLRGYHLYA
4OZG	DQA1*05:05	DQB1*02:01	APQPELPYPQPGS
4P4K	DPA1*01:03	DPB1*02:01	QAFWIDLFETIG
4P57	DPA1*01:03	DPB1*02:01	QAFWIDLFETIGGGSLV
4P5M	DPA1*01:03	DPB1*02:01	QAYDGKDYIALKG

Figure 5.4 Binding core prediction results on 47 HLA class II - peptide complex structures. The observed binding core from 3D structure is underscored. For the binding core predicted by our model, the correctly predicted amino acids are green highlighted, the false positive amino acids are red highlighted, and the false negative amino acids are blue highlighted.

Table 5.1 Benchmark evaluations results. Highlighted cells are highest scores in that test group.

Allele (Alpha/Beta)	Count	Type	Pan specific				Allele specific								Ensemble	
			Our Method		NetMHCIIpan-3.1		NN-align		Comblib matrices		SMM-align		Tepitope (Sturniolo)		Consensus IEDB	
			AUC	SRCC	AUC	SRCC	AUC	SRCC	AUC	SRCC	AUC	SRCC	AUC	SRCC	AUC	SRCC
DQA1*01:02/DQB1*05:01	825	IC_{50}	0.778	0.554	0.596	0.215	-	-	-	-	-	-	-	-	-	-
DQA1*01:02/DQB1*06:02	10	IC_{50}	0.438	-0.115	0.813	0.224	0.813	0.273	0.250	-0.236	1.000	0.382	-	-	0.750	0.224
DQA1*01:03/DQB1*06:03	357	IC_{50}	0.842	0.425	0.809	0.424	-	-	-	-	-	-	-	-	-	-
DQA1*02:01/DQB1*03:01	818	IC_{50}	0.777	0.532	0.814	0.588	-	-	-	-	-	-	-	-	-	-
DQA1*02:01/DQB1*03:03	759	IC_{50}	0.823	0.631	0.760	0.537	-	-	-	-	-	-	-	-	-	-
DQA1*02:01/DQB1*04:02	765	IC_{50}	0.758	0.453	0.519	0.039	-	-	-	-	-	-	-	-	-	-
DQA1*03:01/DQB1*03:02	18	IC_{50}	0.500	0.149	0.973	0.550	0.964	0.622	0.830	0.423	0.946	0.722	-	-	0.991	0.702
DQA1*03:03/DQB1*04:02	567	IC_{50}	0.692	0.299	0.482	-0.080	-	-	-	-	-	-	-	-	-	-
DQA1*05:01/DQB1*03:02	834	IC_{50}	0.803	0.582	0.771	0.574	-	-	-	-	-	-	-	-	-	-
DQA1*05:01/DQB1*03:03	564	IC_{50}	0.822	0.581	0.812	0.614	-	-	-	-	-	-	-	-	-	-
DQA1*05:01/DQB1*04:02	747	IC_{50}	0.718	0.435	0.577	0.140	-	-	-	-	-	-	-	-	-	-
DQA1*06:01/DQB1*04:02	565	IC_{50}	0.726	0.343	0.497	-0.056	-	-	-	-	-	-	-	-	-	-
DRA*01:01/DRB1*01:01	69	Binary	0.809	0.479	0.837	0.521	0.833	0.515	0.661	0.248	0.786	0.443	0.229	-0.420	0.787	0.444
DRA*01:01/DRB1*01:01	1070	IC_{50}	0.827	0.644	0.799	0.635	0.781	0.596	0.696	0.428	0.737	0.507	0.282	-0.457	0.748	0.535
DRA*01:01/DRB1*03:01	1001	IC_{50}	0.775	0.509	0.855	0.710	0.789	0.598	-	-	0.784	0.576	0.247	-0.516	0.797	0.606
DRA*01:01/DRB1*03:01	79	Binary	0.537	0.062	0.624	0.208	0.587	0.146	-	-	0.618	0.196	0.432	-0.113	0.600	0.168
DRA*01:01/DRB1*04:01	104	Binary	0.640	0.242	0.763	0.455	0.770	0.466	-	-	0.692	0.333	0.222	-0.481	0.694	0.336
DRA*01:01/DRB1*04:01	179	IC_{50}	0.732	0.379	0.838	0.579	0.805	0.503	-	-	0.667	0.339	0.439	-0.193	0.676	0.337
DRA*01:01/DRB1*04:04	861	IC_{50}	0.865	0.681	0.861	0.711	0.797	0.592	-	-	0.783	0.582	0.173	-0.632	0.802	0.615
DRA*01:01/DRB1*07:01	1071	IC_{50}	0.855	0.687	0.878	0.761	0.871	0.741	0.787	0.553	0.858	0.712	0.190	-0.607	0.861	0.719
DRA*01:01/DRB1*07:01	69	Binary	0.811	0.535	0.876	0.648	0.863	0.626	0.659	0.274	0.847	0.598	0.260	-0.413	0.840	0.587
DRA*01:01/DRB1*08:01	889	IC_{50}	0.808	0.637	0.863	0.719	-	-	-	-	-	-	0.201	-0.592	0.798	0.591
DRA*01:01/DRB1*08:02	142	IC_{50}	0.752	0.471	0.736	0.439	0.705	0.320	-	-	0.385	0.188	0.721	0.411	0.315	-0.154
DRA*01:01/DRB1*09:01	873	IC_{50}	0.853	0.617	0.868	0.697	0.845	0.669	0.595	0.157	0.791	0.574	-	-	0.810	0.595
DRA*01:01/DRB1*09:01	34	Binary	0.802	0.523	0.837	0.583	0.839	0.586	0.623	0.219	0.795	0.511	-	-	0.759	0.448
DRA*01:01/DRB1*11:01	66	Binary	0.678	0.299	0.749	0.419	0.722	0.373	-	-	0.690	0.319	0.294	-0.347	0.742	0.408
DRA*01:01/DRB1*11:01	1006	IC_{50}	0.851	0.694	0.890	0.778	0.870	0.748	-	-	0.849	0.714	0.204	-0.602	0.840	0.696
DRA*01:01/DRB1*12:02	17	Binary	0.886	0.659	0.800	0.512	-	-	-	-	-	-	-	-	-	-
DRA*01:01/DRB1*13:01	18	Binary	0.988	0.846	0.864	0.632	-	-	-	-	-	-	0.228	-0.471	0.772	0.471
DRA*01:01/DRB1*13:01	866	IC_{50}	0.846	0.652	0.772	0.532	-	-	-	-	-	-	0.210	-0.551	0.790	0.551
DRA*01:01/DRB1*13:02	134	IC_{50}	0.632	0.048	0.903	0.620	0.910	0.606	-	-	0.756	0.422	0.720	0.281	0.765	0.527
DRA*01:01/DRB1*14:54	854	IC_{50}	0.867	0.683	0.889	0.713	-	-	-	-	-	-	-	-	-	-
DRA*01:01/DRB1*15:01	167	IC_{50}	0.866	0.621	0.758	0.498	0.744	0.525	-	-	0.514	0.057	0.636	0.176	0.469	0.019
DRA*01:01/DRB1*15:01	79	Binary	0.537	0.057	0.578	0.120	0.610	0.169	-	-	0.446	-0.082	0.504	0.006	0.517	0.026
DRA*01:01/DRB1*15:02	17	Binary	0.846	0.510	1.000	0.736	-	-	-	-	-	-	0.173	-0.481	0.827	0.481
DRA*01:01/DRB1*15:02	18	IC_{50}	0.933	0.683	0.667	0.409	-	-	-	-	-	-	0.444	-0.387	0.544	0.384
DRA*01:01/DRB3*01:01	852	IC_{50}	0.644	0.293	0.838	0.597	0.827	0.546	0.677	0.300	0.808	0.527	-	-	0.800	0.484
DRA*01:01/DRB3*02:02	771	IC_{50}	0.725	0.422	0.740	0.432	-	-	-	-	-	-	-	-	-	-
DRA*01:01/DRB3*03:01	854	IC_{50}	0.794	0.568	0.781	0.563	-	-	-	-	-	-	-	-	-	-
DRA*01:01/DRB4*01:01	14	IC_{50}	0.725	0.472	0.800	0.516	0.600	0.402	0.725	0.548	0.725	0.460	-	-	0.650	0.465
DRA*01:01/DRB4*01:01	18	Binary	0.615	0.179	0.723	0.347	0.738	0.371	0.492	-0.012	0.554	0.084	-	-	0.646	0.227
DRA*01:01/DRB4*01:03	839	IC_{50}	0.832	0.644	0.786	0.539	-	-	-	-	-	-	-	-	-	-
DRA*01:01/DRB5*01:01	18	Binary	0.889	0.636	0.958	0.750	0.917	0.681	-	-	0.847	0.568	0.306	-0.318	0.778	0.454
DRA*01:01/DRB5*01:01	762	IC_{50}	0.785	0.596	0.843	0.743	0.806	0.660	-	-	0.778	0.609	0.260	-0.526	0.775	0.607
Best Count			19	17	18	20	5	4	0	1	1	2	0	0	1	0

Chapter 6

Conclusions

6.1 Conclusions

In this dissertation proposal, we introduced our work on mislocation related cancer genes indentification and peptide-MHC I binding prediction. For the first work, we showed that our pipeline has the capability to capture misolcation related cancer genes and gave a list of potential cancers genes that are related to proteins mislocation. Comparing with other methods, our pipeline only relies on gene expression data. This makes our pipeline much easier to be applied on other cancers.

In the second part work, our proposed an allele-specific CNN model (DeepMHC) works better than all existing models. Also we explored on how network structure and data encoding will effect the performance for peptide-MHC I binding prediction. In our next work, we presented a pan-specific model (DeepSeqPan) which takes into the raw sequences of HLA alleles and peptides. In this proposed model, we successfully let the model learn to encode the protein sequences instead of using knowledge and structure based pseudo sequence encoding method. Our proposed method offers a novel encoding and modeling solution on this problem. The benchmark test results showed that DeepSeqPan can outperform other methods on certain alleles. We argue that the DeepSeqPan model could be integrated into existing ensemble methods to improved the overall capabilities. In our final work, we proposed a novel attention-like LSTM-CNN model on class II HLA binding problem. Our model is the only available non pseudo sequence based pan-specific model in thia area. Testing with benchmark dataset, our model achieve state-of-the-art performance and outperformed other methods on several certain alleles. With the help of attention mechanism, our model could directly output the binding core prediction. Comparing with existing binding core prediction, our model predicts based on the whole peptide in one-shot. Also, it could offer attention weight on each individual position.

6.2 Future work

6.2.1 Extending DeepSeqPan on dynamic length peptides

In current DeepSeqPan model, we only focus on 9-mers. However, HLA class I could also bind to 10-, 11-length peptides though not so common as 9-length peptides. Considering this, in future we plan to extend the DeepSeqPan model to handle variant peptides input. Right now, we have two options in hand: (i) padding all peptides as a fixed length input, i.e. 11-length, and (ii) using LSTM to encode peptides input. We plan to explore both options and test which one offers better performance.

6.2.2 Other binding problems

As we mentioned before, protein related bindings are very common and important in cell activities. In our work, we focused on HLA-peptide bindings. Other binding problems, such as DNA-binding protein, protein-protein binding and protein-ligand binding etc., also need reliable and effective tools to help researchers design their bench experiment. Since all our proposed method are sequence based without relying on any structure or domain information, we plan to extend these models to other protein related binding problems.

Bibliography

[1] Valsamo Anagnostou et al. "Evolution of Neoantigen Landscape during Immune Checkpoint Blockade in Non–Small Cell Lung Cancer". In: *Cancer discovery* 7.3 (2017), pp. 264–276.

[2] Massimo Andreatta and Morten Nielsen. "Gapped sequence alignment using artificial neural networks: application to the MHC class I system". In: *Bioinformatics* 32.4 (2015), pp. 511–517.

[3] Massimo Andreatta et al. "Accurate pan-specific prediction of peptide-MHC class II binding affinity with improved binding core identification". In: *Immunogenetics* 67.11-12 (2015), pp. 641–650.

[4] Massimo Andreatta et al. "An automated benchmarking platform for MHC class II binding prediction methods". In: *Bioinformatics* 34.9 (2017), pp. 1522–1528.

[5] Michael Ashburner et al. "Gene Ontology: tool for the unification of biology". In: *Nature genetics* 25.1 (2000), p. 25.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[8] Elisabet Ognedal Berge et al. "Identification and characterization of retinoblastoma gene mutations disturbing apoptosis in human breast cancers". In: *Molecular cancer* 9.1 (2010), p. 173.

[9] Daniel Beltrán-Valero de Bernabé et al. "The molecular basis of alkaptonuria". In: *Nature genetics* 14 (1996), p. 19.

[10] Rohit Bhattacharya et al. "Prediction of peptide binding to MHC Class I proteins in the age of deep learning". In: *bioRxiv* (2017), p. 154757.

[11] Janos X Binder et al. “COMPARTMENTS: unification and visualization of protein subcellular localization evidence”. In: *Database* 2014 (2014), bau012.

[12] Huynh-Hoa Bui et al. “Automated generation and evaluation of specific MHC binding predictive tools: ARB matrix applications”. In: *Immunogenetics* 57.5 (2005), pp. 304–314.

[13] Min Chen, Shiwen Mao, and Yunhao Liu. “Big data: A survey”. In: *Mobile Networks and Applications* 19.2 (2014), pp. 171–209.

[14] François Chollet et al. “Keras: Deep learning library for theano and tensorflow”. In: *URL: https://keras. io/k* 7.8 (2015).

[15] Junyoung Chung et al. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. In: *arXiv preprint arXiv:1412.3555* (2014).

[16] Jürgen Drews. “Drug discovery: a historical perspective”. In: *Science* 287.5460 (2000), pp. 1960–1964.

[17] Nivit Gill, Shailendra Singh, and Trilok C Aseri. “Computational disease gene prioritization: an appraisal”. In: *Journal of Computational Biology* 21.6 (2014), pp. 456–465.

[18] Ian Goodfellow et al. *Deep learning*. Vol. 1. MIT press Cambridge, 2016.

[19] Alex Graves and Navdeep Jaitly. “Towards end-to-end speech recognition with recurrent neural networks”. In: *International Conference on Machine Learning*. 2014, pp. 1764–1772.

[20] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. “Speech recognition with deep recurrent neural networks”. In: *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE. 2013, pp. 6645–6649.

[21] Pingping Guan et al. “MHCPred: a server for quantitative prediction of peptide–MHC binding”. In: *Nucleic acids research* 31.13 (2003), pp. 3621–3624.

[22] Emre Guney and Baldo Oliva. “Analysis of the robustness of network-based disease-gene prioritization methods reveals redundancy in the human interactome and functional diversity of disease-genes”. In: *PloS one* 9.4 (2014), e94686.

[23] Youngmahn Han and Dongsup Kim. “Deep convolutional neural networks for pan-specific peptide-MHC class I binding prediction”. In: *BMC bioinformatics* 18.1 (2017), p. 585.

[24] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[25] I Hoof et al. "NetMHCpan: MHC class I binding prediction beyond HLA-A and-B". In: *Tissue Antigens* 72.3 (2008), pp. 251–251.

[26] Ilka Hoof et al. "NetMHCpan, a method for MHC class I binding prediction beyond humans". In: *Immunogenetics* 61.1 (2009), p. 1.

[27] Ziwei Huang. *Drug discovery research: new frontiers in the post-genomic era*. John Wiley & Sons, 2007.

[28] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. "What makes ImageNet good for transfer learning?" In: *arXiv preprint arXiv:1608.08614* (2016).

[29] Mien-Chie Hung and Wolfgang Link. "Protein localization in disease and therapy". In: *J Cell Sci* 124.20 (2011), pp. 3381–3392.

[30] AN Jain. "Virtual screening in lead discovery and optimization." In: *Current opinion in drug discovery & development* 7.4 (2004), pp. 396–403.

[31] Kamilla Kjærgaard Jensen et al. "Improved methods for predicting peptide binding affinity to MHC class II molecules". In: *Immunology* (2018).

[32] Rui Jiang, Mengmeng Wu, and Lianshuo Li. "Pinpointing disease genes through phenomic and genomic data fusion". In: *BMC genomics*. Vol. 16. Suppl 2. BioMed Central Ltd. 2015, S3.

[33] Shihui Jiao et al. "Identification of the causative gene for Simmental arachnomelia syndrome using a network-based disease gene prioritization approach". In: *PloS one* 8.5 (2013), e64468.

[34] E Yvonne Jones et al. "MHC class II proteins and disease: a structural perspective". In: *Nature Reviews Immunology* 6.4 (2006), p. 271.

[35] Johann de Jong et al. "Computational identification of insertional mutagenesis targets for cancer gene discovery". In: *Nucleic acids research* 39.15 (2011), e105–e105.

[36] Vanessa Jurtz et al. "NetMHCpan-4.0: Improved peptide–MHC class I interaction predictions integrating eluted ligand and peptide binding affinity data". In: *The Journal of Immunology* (2017), ji1700893.

[37] Maricel G Kann. “Advances in translational bioinformatics: computational approaches for the hunting of disease genes”. In: *Briefings in bioinformatics* 11.1 (2009), pp. 96–110.

[38] Edita Karosiene et al. “NetMHCcons: a consensus method for the major histocompatibility complex class I predictions”. In: *Immunogenetics* 64.3 (2012), pp. 177–186.

[39] TS Keshava Prasad et al. “Human protein reference database-2009 update”. In: *Nucleic acids research* 37.suppl_1 (2008), pp. D767–D772.

[40] Yohan Kim et al. “Derivation of an amino acid similarity matrix for peptide: MHC binding and its application as a Bayesian prior”. In: *BMC bioinformatics* 10.1 (2009), p. 394.

[41] Diederik Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[42] Pavel P Kuksa et al. “High-order neural networks and kernel methods for peptide-MHC binding prediction”. In: *Bioinformatics* 31.22 (2015), pp. 3600–3607.

[43] Kirsti Laurila and Mauno Vihinen. “Prediction of disease-related mutations affecting protein localization”. In: *BMC genomics* 10.1 (2009), p. 122.

[44] Michael S Lawrence et al. “Discovery and saturation analysis of cancer genes across 21 tumor types”. In: *Nature* 505.7484 (2014), p. 495.

[45] Michael S Lawrence et al. “Discovery and saturation analysis of cancer genes across 21 tumor types”. In: *Nature* 505.7484 (2014), p. 495.

[46] Michael S Lawrence et al. “Discovery and saturation analysis of cancer genes across 21 tumor types”. In: *Nature* 505.7484 (2014), p. 495.

[47] Yann LeCun et al. “A tutorial on energy-based learning”. In: *Predicting structured data* 1 (2006), p. 0.

[48] Hyunju Lee et al. “Diffusion kernel-based logistic regression models for protein function prediction”. In: *Omics: a journal of integrative biology* 10.1 (2006), pp. 40–55.

[49] Hyunju Lee et al. “Diffusion kernel-based logistic regression models for protein function prediction”. In: *Omics: a journal of integrative biology* 10.1 (2006), pp. 40–55.

[50] KiYoung Lee et al. “Protein networks markedly improve prediction of subcellular localization in multiple eukaryotic species”. In: *Nucleic acids research* 36.20 (2008), e136–e136.

[51] KiYoung Lee et al. “Proteome-wide discovery of mislocated proteins in cancer”. In: *Genome research* 23.8 (2013), pp. 1283–1294.

[52] Weizhong Li and Adam Godzik. “Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences”. In: *Bioinformatics* 22.13 (2006), pp. 1658–1659.

[53] John D Lippolis et al. “Analysis of MHC class II antigen processing by quantitation of peptides that constitute nested sets”. In: *The Journal of Immunology* 169.9 (2002), pp. 5089–5097.

[54] Wen Liu et al. “Quantitative prediction of mouse class I MHC peptide binding affinity using support vector machine regression (SVR) models”. In: *BMC bioinformatics* 7.1 (2006), p. 182.

[55] Paul M Lizardi, Matteo Forloni, and Narendra Wajapeyee. “Genome-wide approaches for cancer gene discovery”. In: *Trends in biotechnology* 29.11 (2011), pp. 558–568.

[56] Yong-Chen Lu and Paul F Robbins. “Cancer immunotherapy targeting neoantigens”. In: *Seminars in immunology*. Vol. 28. 1. Elsevier. 2016, pp. 22–27.

[57] Ole Lund. *Immunological Bioinformatics*. MIT press, 2005.

[58] Claus Lundegaard et al. “NetMHC-3.0: accurate web accessible predictions of human, mouse and monkey MHC class I affinities for peptides of length 8–11”. In: *Nucleic acids research* 36.suppl_2 (2008), W509–W512.

[59] Claus Lundegaard et al. “NetMHC-3.0: accurate web accessible predictions of human, mouse and monkey MHC class I affinities for peptides of length 8–11”. In: *Nucleic acids research* 36.suppl_2 (2008), W509–W512.

[60] Jenny Mattison et al. “Cancer gene discovery in mouse and man”. In: *Biochimica et Biophysica Acta (BBA)-Reviews on Cancer* 1796.2 (2009), pp. 140–161.

[61] Rebecca E McIntyre, Louise van der Weyden, and David J Adams. “Cancer gene discovery in the mouse”. In: *Current opinion in genetics & development* 22.1 (2012), pp. 14–20.

[62] Dmytro Mishkin and Jiri Matas. “All you need is a good init”. In: *arXiv preprint arXiv:1511.06422* (2015).

[63] Ananda Mondal and Jianjun Hu. "Network based prediction of protein localisation using diffusion Kernel". In: *International journal of data mining and bioinformatics* 9.4 (2014), pp. 386–400.

[64] Magdalini Moutaftsi et al. "A consensus epitope prediction approach identifies the breadth of murine TCD8+-cell responses to vaccinia virus". In: *Nature biotechnology* 24.7 (2006), p. 817.

[65] KD Munkres et al. "Genetically induced subcellular mislocation of Neurospora mitochondrial malate dehydrogenase". In: *Proceedings of the National Academy of Sciences* 67.1 (1970), pp. 263–270.

[66] Tal Nawy. "Systems biology: Cancer gene discovery goes viral". In: *Nature methods* 9.9 (2012), pp. 868–868.

[67] Simarjeet Negi et al. "LocSigDB: a database of protein localization signals". In: *Database* 2015 (2015).

[68] Morten Nielsen and Massimo Andreatta. "NetMHCpan-3.0; improved prediction of binding to MHC class I molecules integrating information from multiple receptor and peptide length datasets". In: *Genome medicine* 8.1 (2016), p. 33.

[69] Morten Nielsen and Ole Lund. "NN-align. An artificial neural network-based alignment algorithm for MHC class II peptide binding prediction". In: *BMC bioinformatics* 10.1 (2009), p. 296.

[70] Morten Nielsen et al. "Quantitative predictions of peptide binding to any HLA-DR molecule of known sequence: NetMHCIIpan". In: *PLoS computational biology* 4.7 (2008), e1000107.

[71] Morten Nielsen et al. "Reliable prediction of T-cell epitopes using neural networks with novel sequence representations". In: *Protein Science* 12.5 (2003), pp. 1007–1017.

[72] Daniela Nitsch et al. "Candidate gene prioritization by network analysis of differential expression using machine learning approaches". In: *BMC bioinformatics* 11.1 (2010), p. 460.

[73] Timothy J O'Donnell et al. "MHCflurry: open-source class I MHC binding affinity prediction". In: *Cell systems* 7.1 (2018), pp. 129–132.

[74] Giorgio Parmiani et al. "Cancer immunotherapy with peptide-based vaccines: what have we achieved? Where are we going?" In: *Journal of the National Cancer Institute* 94.11 (2002), pp. 805–818.

[75] Adam Paszke et al. *PyTorch*. 2017.

[76] Bjoern Peters and Alessandro Sette. "Generating quantitative models describing the sequence specificity of biological processes with the stabilized matrix method". In: *BMC bioinformatics* 6.1 (2005), p. 132.

[77] Gabriella Pinto, Abdulrab Ahmed M Alhaiek, and Jasminka Godovac Zimmermann. "Proteomics reveals the importance of the dynamic redistribution of the subcellular location of proteins in breast cancer cells". In: *Expert review of proteomics* 12.1 (2015), pp. 61–74.

[78] Tom A Rapoport. "Protein translocation across the eukaryotic endoplasmic reticulum and bacterial plasma membranes". In: *Nature* 450.7170 (2007), p. 663.

[79] Ellis L Reinherz. "$\alpha\beta$ TCR-mediated recognition: relevance to tumor-antigen discovery and cancer immunotherapy". In: *Cancer immunology research* 3.4 (2015), pp. 305–312.

[80] David N Reshef et al. "Detecting novel associations in large data sets". In: *science* 334.6062 (2011), pp. 1518–1524.

[81] James Robinson et al. "IMGT/HLA and IMGT/MHC: sequence databases for the study of the major histocompatibility complex". In: *Nucleic acids research* 31.1 (2003), pp. 311–314.

[82] James Robinson et al. "The IPD and IMGT/HLA database: allele variant databases". In: *Nucleic Acids Research* 43.D1 (2014), pp. D423–D431.

[83] Judith M Rollinger, Hermann Stuppner, and Thierry Langer. "Virtual screening for the discovery of bioactive natural products". In: *Natural Compounds as Drugs Volume I* (2008), pp. 211–249.

[84] L Salwinski et al. "The Database of Interacting Proteins: 2004 update Nucleic Acids Res,. 32". In: *D449 D451* (2004).

[85] Oliver Schmidt, Nikolaus Pfanner, and Chris Meisinger. "Mitochondrial protein import: from proteomics to functional mechanisms". In: *Nature reviews. Molecular cell biology* 11.9 (2010), p. 655.

[86] Danny J Schnell and Daniel N Hebert. "Protein translocons: multifunctional mediators of protein translocation across membranes". In: *Cell* 112.4 (2003), pp. 491–505.

[87] John Sidney et al. "HLA class I supertypes: a revised and updated classification". In: *BMC immunology* 9.1 (2008), p. 1.

[88] John Sidney et al. “HLA class I supertypes: a revised and updated classification”. In: *BMC immunology* 9.1 (2008), p. 1.

[89] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[90] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[91] Tiziana Sturniolo et al. “Generation of tissue-specific and promiscuous HLA ligand databases using DNA microarrays and virtual HLA class II matrices”. In: *Nature biotechnology* 17.6 (1999), p. 555.

[92] Mythili Suntharalingam and Susan R Wente. “Peering through the pore: nuclear pore complex structure, assembly, and function”. In: *Developmental cell* 4.6 (2003), pp. 775–789.

[93] Yaniv Taigman et al. “Deepface: Closing the gap to human-level performance in face verification”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1701–1708.

[94] Laura J Terry, Eric B Shows, and Susan R Wente. “Crossing the nuclear envelope: hierarchical regulation of nucleocytoplasmic transport”. In: *Science* 318.5855 (2007), pp. 1412–1416.

[95] Scott A Tomlins and Arul M Chinnaiyan. “Of mice and men: cancer gene discovery using comparative oncogenomics”. In: *Cancer cell* 10.1 (2006), pp. 2–4.

[96] Thomas Trolle et al. “Automated benchmarking of peptide-MHC class I binding predictions”. In: *Bioinformatics* 31.13 (2015), pp. 2174–2181.

[97] Olga Troyanskaya et al. “Missing value estimation methods for DNA microarrays”. In: *Bioinformatics* 17.6 (2001), pp. 520–525.

[98] Mathias Uhlen et al. “Towards a knowledge-based human protein atlas”. In: *Nature biotechnology* 28.12 (2010), pp. 1248–1250.

[99] Yeeleng S Vang and Xiaohui Xie. “HLA class I binding prediction via convolutional neural networks”. In: *Bioinformatics* (2017), btx264.

[100] Ji Wan et al. “SVRMHC prediction server for MHC-binding peptides”. In: *BMC bioinformatics* 7.1 (2006), p. 463.

[101] Rong-Fu Wang and Helen Y Wang. “Immune targets and neoantigens for cancer immunotherapy and precision medicine”. In: *Cell research* 27.1 (2017), pp. 11–37.

[102] William Wickner and Randy Schekman. “Protein translocation across biological membranes”. In: *science* 310.5753 (2005), pp. 1452–1456.

[103] Wikipedia contributors. *Human leukocyte antigen — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Human_leukocyte_antigen&oldid=862591541`. [Online; accessed 15-October-2018]. 2018.

[104] Janina Wolf, Wolfgang Schliebs, and Ralf Erdmann. “Peroxisomes as dynamic organelles: peroxisomal matrix protein import”. In: *The FEBS journal* 277.16 (2010), pp. 3268–3278.

[105] Bingqing Xie et al. “Disease gene prioritization using network and feature”. In: *Journal of Computational Biology* 22.4 (2015), pp. 313–323.

[106] Bing Xu et al. “Empirical evaluation of rectified activations in convolutional network”. In: *arXiv preprint arXiv:1505.00853* (2015).

[107] Lars Zender et al. “Cancer gene discovery in hepatocellular carcinoma”. In: *Journal of hepatology* 52.6 (2010), pp. 921–929.

[108] 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”. In: *Bioinformatics* 25.10 (2009), pp. 1293–1299.

[109] Lianming Zhang et al. “TEPITOPEpan: extending TEPITOPE for peptide binding prediction covering over 700 HLA-DR molecules”. In: *PloS one* 7.2 (2012), e30483.

[110] Lianming Zhang et al. “Toward more accurate pan-specific MHC-peptide binding prediction: a review of current methods and tools”. In: *Briefings in bioinformatics* 13.3 (2011), pp. 350–364.

[111] Zhi-Qin Zhao et al. “Laplacian normalization and random walk on heterogeneous networks for disease-gene prioritization”. In: *Computational biology and chemistry* 57 (2015), pp. 21–28.