Deep Learning Models for Vaccinology: Predicting T-cell Epitopes in C57BL/6 Mice

Zitian Zhen[1], Yuhe Wang[1], Derin B. Keskin[1,2], Vladimir Brusic[1,3], Lou Chitkushev[1], and Guang Lan Zhang[1(✉)]

[1] Department of Computer Science, Metropolitan College, Boston University, Boston, MA 02215, USA
guanglan@bu.edu
[2] Department of Medical Oncology, Dana-Farber Cancer Institute, Boston, MA 02215, USA
[3] School of Economics, University of Nottingham Ningbo China, Ningbo 315100, China

Abstract. The C57 Black 6 (C57BL/6) mice are one the earliest and most widely used inbred laboratory animals in biomedical research and vaccine development. We propose developing a bioinformatics system for the identification of T-cell epitopes in C57BL/6 mice by integrating multiple contributing factors critical to the antigen processing and recognition pathway. The interaction between peptides and MHC molecules is a highly specific step in the antigen processing pathway and T-cell mediated immunity. As the first step of the project, we built a computational tool for predicting MHC class I binding peptides for the C57BL/6 mice. Utilizing deep learning methods, we trained and rigorously validated the prediction models using naturally eluted MHC ligands. The prediction models are of high accuracy.

Keywords: Bioinformatics System · Deep Learning · Prediction Tool · T-cell Epitope · MHC Binding · C57BL/6 Mice

1 Introduction

The design of vaccines is an intricate, multidimensional task. Because of the growing population and extremely high mobility of the human population, traditional vaccine development technologies are insufficient to address the challenges of pandemics and rapid spread of highly contagious and rapidly mutating viruses. Major challenges are emerging for developing novel vaccines to target pathogens that are difficult to control [1]. Most of newly developed vaccines for protection against COVID-19 are based on vaccine technologies that target antigen-presenting cells that were considered experimental as recently as 2021 [1]. Computational approaches are essential for the rapid development of vaccines because they significantly speed up the vaccine development by rapidly detecting vaccine targets [2]. An effective subunit vaccine must elicit strong immune responses against targets in vaccinated individuals, both B-cell and T-cell responses [3].

Z. Zhen and Y. Wang—These authors contributed equally.

T. Zlateva and G. Tuparov (Eds.): CSECS 2023, LNICST 514, pp. 182–192, 2023.
https://doi.org/10.1007/978-3-031-44668-9_14

The human adaptive immune response, characterized by specificity, involves specialized cells that detect non-self antigens and orchestrate targeted immune reactions [4]. Cytotoxic T cells, a vital component of the adaptive immune system, actively search for short antigenic peptides presented by major histocompatibility complex (MHC) class I molecules. The MHC class I antigen processing pathway involves several key steps: proteins are cleaved into shorter peptides by the proteasome, peptides are translocated into the endoplasmic reticulum (ER) via the transporter associated with antigen processing (TAP), further degradation of peptides may occur in the ER by aminopeptidases, peptides bind MHC molecules, and finally, peptide-MHC complexes are transported to the cell surface for recognition by CD8+T cells [5]. MHC binding is considered the most selective stage in T cell recognition. Peptides presented by MHC and recognized by T cells are referred to as T-cell epitopes. Accurately identifying T cell epitopes speeds up the design and development of epitope-based vaccines [6, 7].

Computational approaches have been successful in designing population-based vaccines by precisely identifying vaccine targets that offer broad protection across population. However, given that COVID-19 is highly contagious and rapidly mutating, there were several practical issues with available vaccines due to viral and host factors [8]: 1) individuals have different immunological profiles, and there will be antigenic mismatch in subpopulations; 2) increased transmissibility reduces vaccine efficiency; and 3) rapid viral mutations make optimal vaccine a moving target. Computational vaccinology needs to address the issues of high host diversity vs. rapid mutation of pathogen, with highly dynamic environment of optimal vaccine targeting. Deep learning has been proposed as an approach for effective vaccine design [9]. It was even suggested that deep learning models cannot be run on personal computers, but they require supercomputing facilities with extensive mass spectrometry support [10, 11]. These approaches are top-down, and they do not resolve the host diversity and target mutability problem. We propose a bottom-up approach whereby deep learning is applicable to individual immunological profiles and the results provide individualized vaccine targets. The trained models then can be used to rapidly assess viral mutations and potential immune escape of viral variants. To explore this goal, we developed a model that targets a specific combination of MHC alleles and, for simplicity, deployed it on a mouse model.

Mouse models are widely utilized in immunological research due to their relatively less complex composition of the immunopeptidome [12]. Studies have shown that vaccines composed of synthetic peptides resembling cancer neoepitopes have resulted in efficient T-cell activities and killed cancer cells in both mouse models and human patients [13–16]. The C57 Black 6 (C57BL/6) mice are one the earliest and most widely used inbred laboratory animals in biomedical research and vaccine development. C57BL/6 mouse expresses two MHC class I alleles, H2-D^b and -K^b [12].

Reverse immunology approaches involve the identification of immune targets through extensive bioinformatics screening of complete pathogenic genomes, followed by experimental validation [17].. Multiple online bioinformatics systems have been developed to predict peptides binding MHC alleles, including H2-D^b and H2-K^b alleles [6, 18, 19]. Many of them were trained mainly based on MHC binding peptides identified using *in vitro* binding assays. MHC-peptide *in vitro* binding assays assess the binding of

peptides to specific MHC molecules, a prerequisite for T-cell activation. With technological advancement, liquid chromatography-tandem mass spectrometry (LC-MS/MS) has been employed to physically detect naturally processed and presented peptides on the cell surface [20–22]. We refer to these experimentally determined, naturally processed peptides as eluted ligands. As more and more datasets of eluted ligands are becoming available, this gives us opportunities to build more accurate prediction models using more biologically relevant data while incorporating the antigen processing steps simultaneously [20–23].

Moreover, most existing tools only model a single step in the MHC class I antigen processing pathway, the binding between MHC molecules and peptides. We propose a computational system for the identification of T-cell epitopes in C57BL mice by integrating other contributing factors critical to the antigen processing and recognition pathway, including proteasome cleavage, gene expression, and antigenicity. As the first step, we built a bioinformatics tool that predicts binding peptides of the MHC class I molecules H2-D^b and -K^b, of the C57BL/6 mice. Utilizing deep learning methods, we trained and rigorously validated the prediction models using naturally eluted MHC ligands. The prediction models are of high accuracy. This system is a prototype for exploring personalized targeting of vaccines for highly contagious and rapidly mutating pathogens. Because of high combinatorial complexity of host-pathogen interaction of such viruses, deep learning represents a promising platform for improving personalized vaccine targeting.

2 Materials and Methods

2.1 Data Collection and Transformation

The Immune Epitope Database and Analysis Resource (IEDB) compiles manually curated information on experimentally discovered B Cell and T cell epitopes found on various species, MHC binding peptides, and accompanying experimental settings [24]. Its contents were gathered primarily from literature from 1952 to now. We assembled our training data sets by collecting MHC ligand elution assay data from IEDB and enriched it with information from peptide processing tools [23]. When performing searches in IEDB, the following search criteria were adopted, Epitope: Linear peptide, Assay: MHC ligand, Assay Type: MHC Ligand Elution Assay, Outcome: Positive, and MHC Restriction: H2-D^b (or H2-K^b) protein complex. The search result contained 96,838 assay entries, with 44,565 for H2-D^b and 52,273 for H2-K^b.

We discarded ligands identified only by one positive assay to increase data quality. Ligands with two or more positive assay records were kept in the data sets as MHC binders. Our final dataset included 3,395 H2-D^b ligands and 3,961 H2-K^b ligands of length 8–11. Additional binding peptides were extracted from the NetMHCpan 4.1 training datasets [23]. After comparing the two data sets and removing duplicates, 885 H2-D^b ligands and 1,187 H2-K^b ligands from the NetMHCpan 4.1 dataset were added to the training dataset. Table 1 summarizes the numbers of MHC ligands collected. We also extracted random natural peptides from the NetMHCpan 4.1 datasets to use as non-binders. As the first step, we focused on building prediction models for H2-D^b and -K^b 9-mer ligands.

Table 1. Eluted H2-D^b and H2-K^b ligands collected from IEDB and NetMHCpan 4.1 dataset.

Length	H2-D^b ligands			H2-K^b ligands		
	IEDB	NetMHCpan	Sum	IEDB	NetMHCpan	Sum
8	210	14	224	2411	646	3057
9	**2357**	**585**	**2942**	**1292**	**395**	**1687**
10	481	151	632	175	68	243
11	347	135	482	83	78	161
Sum	3395	885	4280	3961	1187	5148

2.2 The Deep Learning Model

Deep learning, a branch of machine learning, is highly proficient in unveiling patterns from vast, multifaceted labeled data. This proficiency comes from its capacity to model high-level abstractions using architectural constructs known as neural networks [25]. Neural networks consist of interconnected layers, including an input layer, at least one hidden layer, and an output layer. The hidden layers are internal to the network and learn complex features from the inputs, whereas the output layer produces the final predictions. We developed deep neural network (DNN) models with several layers to learn patterns and characteristics within our training data. These layers serve as processing stages - akin to filters - each adding more complexity. We employed the Python Keras library [26] in this study. An Application Programming Interface (API) enables software applications to establish communication and interact with one another. The high-level neural network API of the Keras library can run on top of lower-level libraries like TensorFlow, Theano, or CNTK [25]. Keras provides predefined layers, such as dense (fully connected) layers, convolutional layers, and recurrent layers, which allow for a high degree of flexibility in designing custom neural network architectures. Due to its simplicity and effectiveness, we implemented a sequential model with dense (fully connected) layers using the backpropagation algorithm.

2.3 The Study Design

The overall structure of our study consisted of the following steps:

1. The 9-mer data of both alleles were divided into training and testing datasets using a stratified method to maintain an approximate 80/20 ratio. To ensure reproducibility, we set the random seed to 42. Details of the datasets are shown in Table 2.
2. We encoded each 9-mer ligand in the training data into two NumPy arrays. The first array represents the nine amino acids in the ligand, as shown in Fig. 1. After evaluating several amino acid encoding methods, we selected the one-hot encoding for simplicity and effectiveness [27]. The second array contains a binary label indicating binding (1) or non-binding (0).
3. We then constructed dense neural network models. The final model consists of one input layer and two hidden layers. The first hidden layer contains 128 neurons, and

Table 2. Training and testing datasets for A) H2-D^b and B) H2-K^b. The binder/non-binder ratio is 19/81 and 11/89 for H2-D^b and -K^b training and testing datasets.

Datasets	# of Binders (Percentage)	# of Non-binders (Percentage)	Total
H2-D^b			
Training	2349 (19%)	9992 (81%)	12341
Testing	593 (19%)	2493 (81%)	3086
sum	2942	12485	15427
H2-K^b			
Training	1345 (11%)	10706 (89%)	12051
Testing	342 (11%)	2671 (89%)	3013
sum	1687	13377	15064

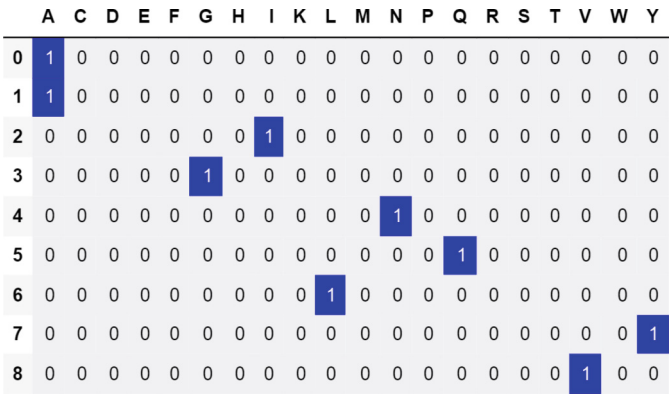


Fig. 1. The array representing the 9-mer peptide, AAIGNQLYV, using One-hot encoding.

the second has 64 neurons with a Leaky ReLU activation function to introduce non-linearity and allow the learning of more complex patterns [28]:

$$LeakyReLU\,(x) = \begin{cases} x & if \quad x > 0 \\ \alpha x & if \quad x \leq 0 \end{cases} \tag{1}$$

where α is a small positive constant, typically a small fraction. In our model, we set $\alpha = 0.01$ to allow a small gradient for negative values, allowing learning to occur even for negative inputs to avoid the vanishing and exploding gradients problem. The output layer has a sigmoid activation function, guaranteeing that the output is between 0 and 1, making it useful for binary classification [29].

$$Sigmoid\,(x) = \frac{1}{1 + e^{-2}} \tag{2}$$

MHC binders tend to produce values close to 1(binding), while non-binders tend to yield values close to 0 (non-binding). We compiled this model using the binary cross-entropy loss function [30] since both the sigmoid activation and binary cross-entropy loss functions are constructed for binary classification problems [25].

$$L = -[y * \log(p) + (1 - y) * \log(1 - p)] \tag{3}$$

where y represents the actual class label (either 0 or 1) of the binary classification problem, and p represents the predicted probability of the positive class. The loss function computes the logarithmic loss between p and y. When y is 1, the loss penalizes the model more if it predicts a low probability. When y is 0, the loss penalizes the model more for predicting a high probability for the positive class. The negative sign at the beginning of the equation converted it into a minimization problem. The goal is to minimize this loss function during the training to update the model's parameters and improve its predictive performance.

4. Batch normalization, a technique in deep learning, was applied to improve a neural network's performance. This is achieved by normalizing the mini-batch, and then scaling and shifting the normalized values using learned parameters. Batch normalization stabilizes the distribution of each layer's inputs during training, leading to faster and more stable convergence [31]. After trying various sizes for batch normalization, we chose batch size 32 as it produced the best performance.
5. We also used early stopping, a form of regularization to avoid overfitting [32]. If the model's prediction performance on the validation set does not demonstrate improvement for a predefined number of epochs, the training process stops. We started training by running 100 epochs, and the model stopped in less than 20 epochs. After implementing the early stopping function with hyper-parameter patience being 10, we changed the epoch to 30. We trained the networks with 30 epochs and a batch size 32 and evaluated the model using the testing datasets.
6. We then fine-tuned the hyper-parameters to optimize the performance, including adjusting the number of hidden layers, the size each layer, and the optimizer learning rate. We also tried various activation functions in hidden layers. We defined a callback function, such as Model Checkpoint, to record the best-performing model [25]. This way, we ensured that the best-performing model was recorded during training, which is helpful when trying out multiple sets of parameters and working with large datasets or when training takes a long time. We tried various optimizers and settled on Adam as it produced the best performance (Fig. 2) [33]. In stochastic gradient descent, the learning rate needs to be manually tuned. Adam computes individual learning rates for different parameters, resulting in adaptive learning rates.

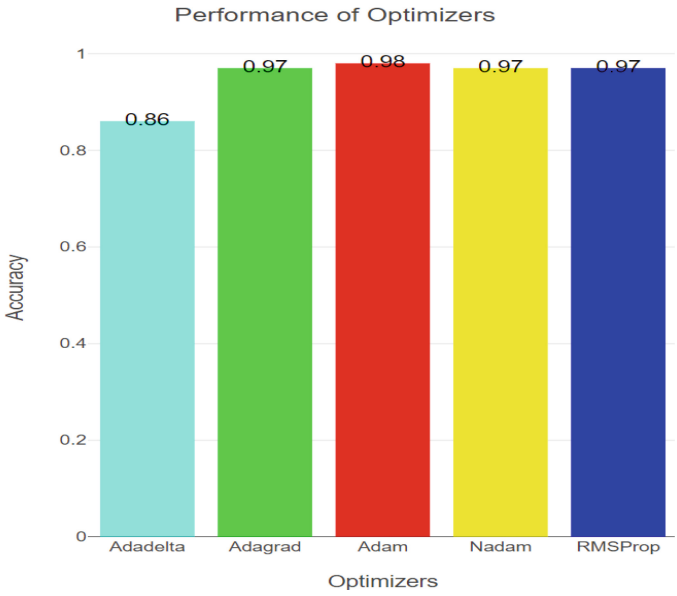


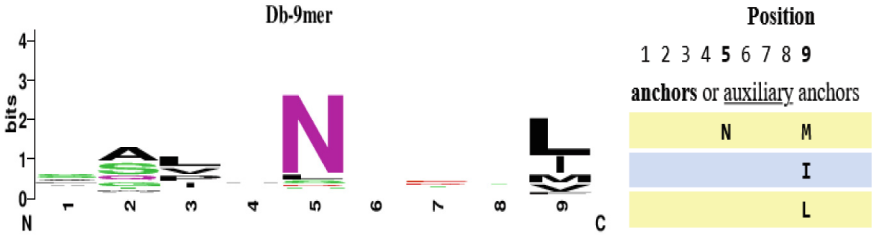
Fig. 2. The performance of various optimizers.

2.4 The Validation Dataset

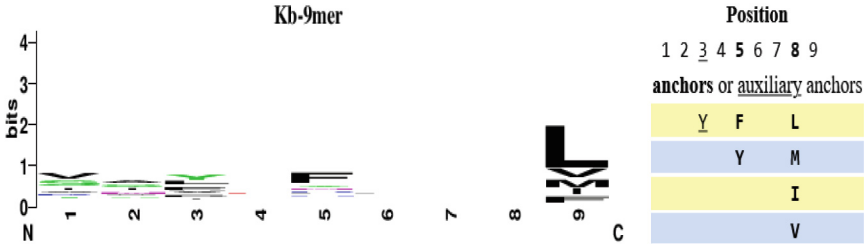
In 2020, Paul et al. benchmarked the publicly available MHC binding prediction models using a set of H2-D^b and H2-K^b restricted T-cell epitopes derived from the Vaccinia virus (VACV)" [19]. All models demonstrated reasonable prediction performance, and the NetMHCpan-4.0 and MHCflurry, based on deep learning models trained on MHC binding affinity and eluted ligand data, were the top performers among the 17 models evaluated. Many of these models were developed over a decade ago, while NetMHCpan-4.0 and MHCflurry are the newest additions. We validated the developed prediction model using the same published dataset [19]. It contains naturally processed and eluted peptides from VACV-infected mice cells [34]. Out of all LC-MS/MS identified peptides, we filtered out 111 9-mer ligands, 78 for H2-D^b and 33 for H2-K^b. We randomly generated 549 9-mer peptides using the VACV proteomes data and included them as non-binders in the validation dataset.

3 Results

SYFPEITHI is one of the earliest online databases that capture information on MHC binding peptides, MHC binding motifs, and anchor positions et al. [35]. SYFPEITHY motifs were used as a guide for assessing anchor positions in binding peptides. We generated sequence logos using WebLogo [36] based on the 9-mer ligands collected and compared them with binding motifs provided by SYFPEITHI (Fig. 3). Both the H2-D^b binding motifs identify positions 5 and 9 as anchor positions. In addition, the sequence logo showed apparent amino acid selectivity at positions 2 and 3. The H2-K^b binding motifs show high similarity.



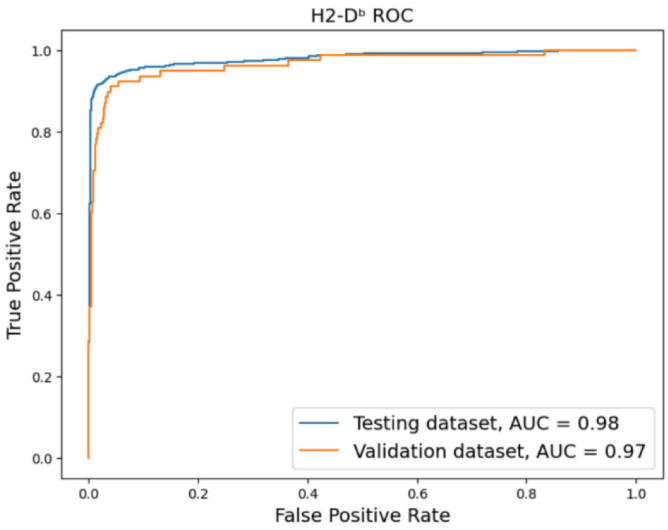
A) H2-D^b 9-mer binding motifs generated by WebLogo (left) and SYFPEITHI (right).



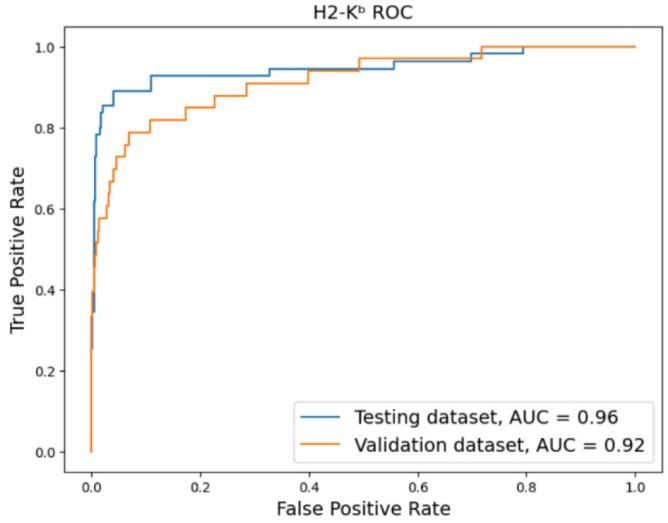
B) H2-K^b 9-mer binding motifs generated by WebLogo (left) and SYFPEITHI (right).

Fig. 3. Binding motifs for A) H2-D^b and B) H2-K^b 9-mer peptides.

Our prediction models were trained and tested using the training and testing datasets described in Table 2. The models were validated using the validation dataset. ROC curves (receiver operating characteristic curve) and AUC (area under the ROC) are used to assess the overall prediction performance. As shown in Fig. 4, both models demonstrated high accuracy. The AUC values of the H2-D^b model were 0.98 using the testing data and 0.97 using the validation data. The AUC values of the H2-K^b model were 0.96 using the testing data and 0.92 using the validation data.



A) H2-Dᵇ ROC curves.



B) H2-Kᵇ ROC curves.

Fig. 4. ROC curves show the performance of the prediction models.

4 Conclusion

Immune-based therapies are revolutionizing cancer care and vaccine development. Most of these therapies are first developed and tested in mice due to their shared mammalian features with humans. Understanding the epitope presentation to the T cells in mice is crucial for analyzing and interpreting these immune therapies. We propose developing a bioinformatics pipeline based on deep learning for *in silico* prediction of T-cell epitopes

in C57BL mice. This computational modeling approach integrates peptide binding and antigen processing in one deep learning system. It addresses critical contributing factors to the antigen processing and recognition pathways, including proteasomal cleavage, gene expression, and antigenicity. Here we reported preliminary results on building a prediction tool for peptide binding to H2-D^b and H2-K^b alleles that represent the immunological profile of C57BL/6 mice, a common research model. We trained and rigorously validated dense neural network models using naturally eluted MHC ligands. The prediction models are highly accurate, evidenced by the high AUC values in the testing and validation. We plan to build on this model, incorporate additional tools to develop a highly accurate bioinformatic pipeline for *in silico* prediction of T-cell epitopes in C57BL mice and other mouse models, and extend it to human haplotypes.

References

1. Pollard, A.J., Bijker, E.M.: A guide to vaccinology: from basic principles to new developments. Nat. Rev. Immunol. **21**(2), 83–100 (2021)
2. Osamor, V.C., Ikeakanam, E., Bishung, J., Abiodun, T., Ekpo, R.H.: COVID-19 vaccines: computational tools and development. Inform. Med. Unlocked 101164 (2023)
3. Grødeland, G., Fossum, E., Bogen, B.: Polarizing T and B cell responses by APC-targeted subunit vaccines. Front. Immunol. **6**, 367 (2015)
4. Bonilla, F.A., Oettgen, H.C.: Adaptive immunity. J. Aller. Clin. Immunol. **125**(2), S33–S40 (2010)
5. Shastri, N., Cardinaud, S., Schwab, S.R., Serwold, T., Kunisawa, J.: All the peptides that fit: the beginning, the middle, and the end of the MHC class I antigen-processing pathway. Immunol. Rev. **207**(1), 31–41 (2005)
6. Zhang, G.L., Keskin, B.D., Chitkushev, L.: Extraction of Immune Epitope Information. In: Ranganathan, S., Nakai, K., Schönbach, C., Gribskov, M. (eds.) Encyclopedia of Bioinformatics and Computational Biology, vol. 3, pp. 39–46. Elsevier, Oxford (2019)
7. Brusic, V., Petrovsky, N., Gendel, S.M., Millot, M., Gigonzac, O., Stelman, S.J.: Computational tools for the study of allergens. Allergy **58**(11), 1083–1092 (2003)
8. Tregoning, J.S., Flight, K.E., Higham, S.L., Wang, Z., Pierce, B.F.: Progress of the COVID-19 vaccine effort: viruses, vaccines and variants versus efficacy, effectiveness and escape. Nat. Rev. Immunol. **21**(10), 626–636 (2021)
9. Abbasi, B.A., et al.: Identification of vaccine targets & design of vaccine against SARS-CoV-2 coronavirus using computational and deep learning-based approaches. PeerJ **10**, e13380 (2022)
10. Keshavarzi Arshadi, A., et al.: Artificial intelligence for COVID-19 drug discovery and vaccine development. Front. Artif. Intell. **65** (2020)
11. Bagabir, S.A., Ibrahim, N.K., Bagabir, H.A., Ateeq, R.H.: Covid-19 and artificial intelligence: genome sequencing, drug development and vaccine discovery. J. Infect. Public Health **15**(2), 289–296 (2022)
12. Schuster, H., et al.: A tissue-based draft map of the murine MHC class I immunopeptidome. Sci. Data **5**(1), 1–11 (2018)
13. Banchereau, J., Palucka, K.: Cancer vaccines on the move. Nat. Rev. Clin. Oncol. **15**(1), 9–10 (2018)
14. Sahin, U., Türeci, Ö.: Personalized vaccines for cancer immunotherapy. Science **359**(6382), 1355–1360 (2018)
15. Ott, P.A., et al.: An immunogenic personal neoantigen vaccine for patients with melanoma. Nature **547**(7662), 217–221 (2017)

16. Keskin, D.B., et al.: Neoantigen vaccine generates intratumoral T cell responses in phase Ib glioblastoma trial. Nature **565**(7738), 234–239 (2019)
17. Rappuoli, R., Bottomley, M.J., D'Oro, U., Finco, O., De Gregorio, E.: Reverse vaccinology 2.0: human immunology instructs vaccine antigen design. J. Exp. Med. **213**(4), 469–481 (2016)
18. Zhang, G.L., Sun, J., Chitkushev, L., Brusic, V.: Big data analytics in immunology: a knowledge-based approach. BioMed Res. Int. **2014**, 1–9 (2014)
19. Paul, S., et al.: Benchmarking predictions of MHC class I restricted T cell epitopes in a comprehensively studied model system. PLoS Comput. Biol. **16**(5), e1007757 (2020)
20. Sarkizova, S., et al.: A large peptidome dataset improves HLA class I epitope prediction across most of the human population. Nat. Biotechnol. **38**(2), 199–209 (2020)
21. Abelin, J.G., et al.: Mass spectrometry profiling of HLA-associated peptidomes in mono-allelic cells enables more accurate epitope prediction. Immunity **46**(2), 315–326 (2017)
22. Truex, N.L., et al.: Automated flow synthesis of tumor neoantigen peptides for personalized immunotherapy. Sci. Rep. **10**(1), 723 (2020)
23. Reynisson, B., Alvarez, B., Paul, S., Peters, B., Nielsen, M.: NetMHCpan-4.1 and NetMHCIIpan-4.0: improved predictions of MHC antigen presentation by concurrent motif deconvolution and integration of MS MHC eluted ligand data. Nucleic Acids Res. **48**(W1), W449-W454 (2020)
24. Vita, R., et al.: The immune epitope database (IEDB): 2018 update. Nucleic Acids Res. **47**(D1), D339–D343 (2019)
25. Géron, A.: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd edn.. O'Reilly Media, (2019)
26. Chollet, F., et al.: (2015). https://github.com/keras-team/keras. Accessed 7 June 2023
27. ElAbd, H., Bromberg, Y., Hoarfrost, A., Lenz, T., Franke, A., Wendorff, M.: Amino acid encoding for deep learning applications. BMC Bioinform. **21**, 1–14 (2020)
28. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. Proc. Icml **30**(1), 3 (2013)
29. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
30. Topsøe, F.: Bounds for entropy and divergence for distributions over a two-element set. JIPAM. J. Inequal. Pure Appl. Math. **2**(2) (2001)
31. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456 (2015)
32. Yao, Y., Rosasco, L., Caponnetto, A.: On early stopping in gradient descent learning. Constr. Approx. **26**(2), 289–315 (2007)
33. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980, (2014)
34. Croft, N.P., et al.: Most viral peptides displayed by class I MHC on infected cells are immunogenic. Proc. Natl. Acad. Sci. **116**(8), 3112–3117 (2019)
35. Schuler, M.M., Nastke, M.D., Stevanović, S.: SYFPEITHI: database for searching and T-cell epitope prediction. Immunoinform.: Predict. Immunogenicity Silico 75–93 (2007)
36. Crooks, G.E., Hon, G., Chandonia, J.M., Brenner, S.E.: WebLogo: a sequence logo generator. Genome Res. **14**(6), 1188–1190 (2004)