PiTE: TCR-epitope Binding Affinity Prediction Pipeline using Transformer-based Sequence Encoder

Pengfei Zhang [1,2], Seojin Bang [2§] and Heewook Lee [1,2 †]

[1] *School of Computing and Augmented Intelligence*
Arizona State University, Tempe, AZ, United States
[†] *E-mail: heewook.lee@asu.edu*

[2] *Biodesign Institute*
Arizona State University, Tempe, AZ, United States

Accurate prediction of TCR binding affinity to a target antigen is important for development of immunotherapy strategies. Recent computational methods were built on various deep neural networks and used the evolutionary-based distance matrix BLOSUM to embed amino acids of TCR and epitope sequences to numeric values. A pre-trained language model of amino acids is an alternative embedding method where each amino acid in a peptide is embedded as a continuous numeric vector. Little attention has yet been given to summarize the amino-acid-wise embedding vectors to sequence-wise representations. In this paper, we propose PiTE, a two-step pipeline for the TCR-epitope binding affinity prediction. First, we use an amino acids embedding model pre-trained on a large number of unlabeled TCR sequences and obtain a real-valued representation from a string representation of amino acid sequences. Second, we train a binding affinity prediction model that consists of two sequence encoders and a stack of linear layers predicting the affinity score of a given TCR and epitope pair. In particular, we explore various types of neural network architectures for the sequence encoders in the two-step binding affinity prediction pipeline. We show that our Transformer-like sequence encoder achieves a state-of-the-art performance and significantly outperforms the others, perhaps due to the model's ability to capture contextual information between amino acids in each sequence. Our work highlights that an advanced sequence encoder on top of pre-trained representation significantly improves performance of the TCR-epitope binding affinity prediction*.

Keywords: TCR; epitope; binding affinity prediction; sequence encoder.

1. Introduction

T cells play fundamental roles in the adaptive immune system. T cell receptor (TCR) is a cell surface protein complex that binds to peptides presented by antigen presenting cells (APCs) via major histocompatibility complex (MHC, pMHC is the peptide-MHC multimers that are presented to T cells).[1] A successful binding and recognition of a foreign antigen triggers an immune response to defend our body from the invaders. The binding is essentially determined

[§] Now at Google.
* Code and models are publicly available at `https://github.com/Lee-CBG/PiTE`

by two short amino acid chains.[2] One is an epitope, a part of antigen peptides bound within pMHC presented by APCs and a TCR is the counterpart. Of a TCR, the complementarity-determining region 3 (CDR3) of TCR β chain is known to be the most important part that interacts with its cognate epitope pairs.[2–4]

Accurate prediction of TCR binding affinity to a target epitope is a critical step to unraveling the underlying binding mechanisms. Especially, the ability to predict computationally is extremely valuable as it can automate screening of cognate TCRs for an epitope of interest. Computational screening of a confident candidate set of TCRs for a target epitope can dramatically reduce the time and the cost of wet lab assays, thereby further enabling rapid development of personalized immunotherapy.[5,6]

Many machine learning models to predict the binding affinity of TCR and epitope sequences have been developed.[7–14] While earlier models such as TCRex[8] and TCRGP[7] utilized random forest and gaussian process respectively, more recent models leveraged a large capacity of deep neural networks. For example, NetTCR[9] and NetTCR2.0[10] were built on multiple convolutional neural network (CNN) layers with different sizes of filters to encode each sequence followed by dense layers to predict the binding affinity scores between the encoded sequences. To accommodate the amino acid sequential data, ERGO[11] utilized a long-short term memory (LSTM)[15] layer followed by a multi-layer perceptron. Similarly, TITAN[12] and ATM-TCR[13] leveraged the attention mechanism.[16]

The first step to process the input for these machine learning models is translating string representation of peptides (both TCR and epitope sequences) into a real-valued numeric vector. Overwhelmingly many models[7–10,12] map each amino acid in a TCR (or epitope) sequence to a predefined vector of numeric values using evolutionary-based distance matrices BLOSUM.[17] However, the models using BLOSUM-based embedding suffer from limited performance, especially when predicting binding affinity for out-of-sample epitopes[13] not present in the training data the models were trained on.

In order to improve generalized prediction performance, several amino acids embedding models have been proposed.[11,14,18,19] These models were trained on a large number of unpaired TCR sequences by considering the input sequence itself as the supervision signal. Among these, especially the embedding models[14,19] whose architectures were inspired by language representation models such as Bert[20] and ELMo[21] have shown to learn more effective contextualized embeddings for TCR and epitope sequences and improved prediction performance. Typically, such models yield a larger size of embedding vectors than those of BLOSUM-based method. Average pooling has been commonly used to reduce the size of the embedding model outputs and enable training a binding affinity prediction model with less computational burden. However, it wipes off position-specific information and degrades prediction performance because it averages vectors over all amino acids.

We propose `PiTE`, a **Pi**peline leveraging **T**ransformer-like **E**ncoders to predict the binding affinity between a pair of TCR and epitope sequences. Our pipeline consists of two parts: (1) amino acids embedding for each TCR and epitope, and (2) binding affinity prediction between the two sequences. First, we use a pre-trained embedding model to map string representations of amino acids sequences (e.g., `GLCTLVAML`) to a sequence of real-valued vectors. It leverages a

large number of unlabeled TCR sequences to train an embedding model, and learn contextual representations of TCRs and epitopes using a bidirectional LSTM architecture. Second, we train a binding affinity prediction model that takes a pair of TCR and epitope embeddings as an input and returns a binding affinity score between those two sequences. `PiTE` encodes TCR and epitope amino acids embeddings using two sequence encoders, respectively, and determines the binding affinity between those two sequences using multiple linear layers. In particular, we explore various different types of neural network architectures to encode each sequence on top of existing embedding models. We highlight the importance of an advanced sequence encoder to boost the performance of the TCR-epitope binding affinity prediction.

2. Data

2.1. *Positive Sample Collection*

To train our models, we sampled TCR-epitope pairs with known binding affinity from three publicly available databases–IEDB,[22] VDJdb,[23] and McPAS.[24] Pairs with MHC class I type epitopes and TCRβ CDR3 sequences were used in our analysis. In this paper, TCR sequence refers to CDR3 unless otherwise stated. Sequences containing wildcard amino acids, such as `*` and `X` were excluded. After removing duplicates from three databases, a total of 150,008 unique TCR-epitope pairs known to bind were obtained.

2.2. *Negative Sample Generation*

While there is real negative binding data,[10] the dataset only covers a limited number of epitopes (19 epitopes), we strictly generated the same number of negative samples so that our data have an 1:1 ratio of positive and negative samples. In detail, we collected TCR sequences from TCR repertoires of healthy controls in ImmunoSEQ[25] portal. We then replaced TCRs of the positive TCR-epitope pairs with TCRs randomly selected from the healthy controls, resulting in 150,008 negative TCR-epitope pairs. Combining our collected positive pairs and generated negative pairs, we had 300,016 unique TCR-epitope pairs in total.

2.3. *Training and Testing Set Split*

The binding characteristic of TCRs and epitopes is many-to-many, which means a TCR can bind to multiple epitopes and an epitope can bind to multiple TCRs. Considering that our dataset has 290,683 unique TCRs and 982 unique epitopes, it is highly likely that an epitope can be found in both training and testing sets if we randomly split the sets. It is less likely that a TCR present in both training and testing sets, but this can still happen. Therefore, the random split of training and testing sets cannot properly measure generalization performance of our model on novel TCRs and epitopes. In order to measure generalization performance on novel TCRs and epitopes, we followed two dataset splitting approaches used in ATM-TCR:[13] the TCR split and the epitope split. In the TCR split, no testing TCRs ever appeared in the training set, allowing us to evaluate the performance of binding affinity prediction models on out-of-sample TCRs. Similarly, in the epitope split, no testing epitopes ever appeared in the training set, allowing us to evaluate the performance on out-of-sample epitopes.

3. Methods

PiTE consists of two parts: amino acid embedding and TCR-epitope binding affinity prediction (see Fig. 1). In the TCR (or epitope) amino acids embedding part, we use a pre-trained embedding model to map a TCR (or epitope) sequence of string representation of amino acids to a sequence of real-valued vectors. In the binding affinity prediction part, we train a variety of different binding affinity prediction models, which composed of two sequence encoders (one for TCR and the other for epitope) and a block of linear classification layers. In particular, we are interested in how different types of encoders would perform in summarization of amino-acid-wise embedding vectors into a sequence-wise representation.

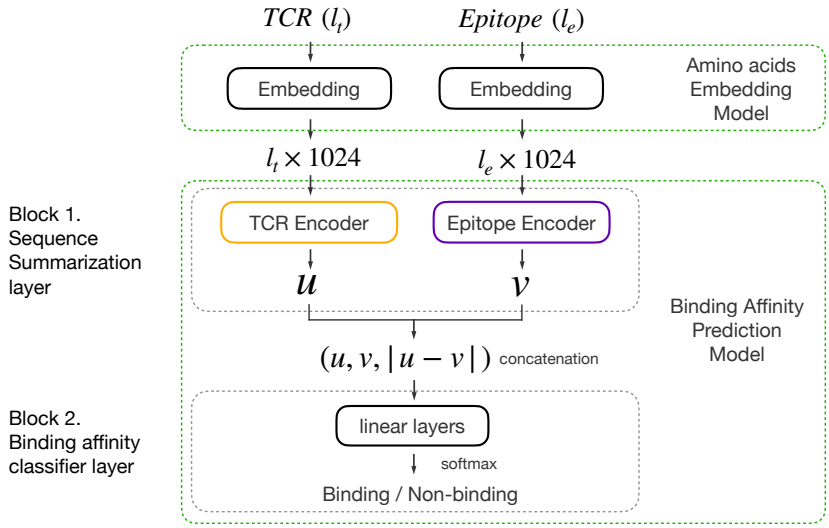

Fig. 1. PiTE pipeline: A TCR sequence with length of l_t is first fed to the amino acids embedding model. Each amino acid is embedded as a 1×1024 vector, hence a TCR sequence is embedded as a $l_t \times 1024$ matrix. Similarly, an epitope sequence with length l_e is embedded as a $l_e \times 1024$. These embeddings are then passed into each sequence encoder to obtain the summarized representation u and v for the TCR and epitope sequence, respectively. Finally, u, v, and their absolute subtraction $|u - v|$ are concatenated, and fed to two linear layers followed by a softmax activation function to predict the binding affinity between the TCR and epitope sequences. Note sequence encoder layers and binding affinity classifier layer are trained together as one binding affinity prediction model.

3.1. *Amino Acids Embedding*

Amino acid embedding is a process to map each amino acid in a TCR (or epitope) sequence to a real-valued vector. Recently, amino acid embedding models[11,14,18,19] leveraging a large number of (unlabeled) TCR sequences have shown great advantages over the BLOSUM-based models. We use a pre-trained amino acids embedding model[19] trained on unlabeled TCR sequences collected from ImmunoSEQ portal. The embedding model adopted the overall architecture from a widely used language representation model, ELMo,[21] with different layer sizes. Note that this paper does not aim to find an optimal architecture for amino acid embedding. We

use this model because it performs the best on our dataset, but it can be replaced by any other state-of-the-art embedding models such as TCR-Bert[14] and DeepTCR.[18]

The embedding model serves as a feature extractor that maps each amino acid in a string representation of TCR (or epitope) sequence to a numeric vector of size of 1×1024. Therefore, a TCR sequence of length l_t is represented by a sequence of embedding vectors (i.e., a matrix of size $l_t \times 1024$). Similarly, an epitope sequence of length l_e is represented by a sequence of embedding vectors (i.e., a matrix of size $l_e \times 1024$). These embeddings will serve as the input of the binding affinity prediction model. Since the binding affinity prediction model requires the input to have the same shape and size, we align TCRs and epitopes using the IMGT approach with a predefined length l. If the length of the TCR sequence (l_t) is longer than l, we remove an embedding vector of the amino acid from the end until it equals l. Otherwise, we append zeros to the end of embedding vectors to ensure the embedding length is l. We predefine l as 22 for both the TCR and epitope sequences. This preprocessing step is applied before feeding the TCR (or epitope) embeddings into our sequence encoder except for the baseline average pooling encoder.

3.2. *TCR-epitope Binding Affinity Prediction*

3.2.1. *Sequence Encoders*

Average pooling (baseline): Average pooling is a pooling technique that projects a high dimensional matrix to a low dimensional one by averaging values with regards to some feature dimension. It has been commonly used for obtaining sequence representations from the output of amino acids embedding models. It helps to reduce the dimension of the amino acid embedding of which the size is generally larger than the BLOSUM embedding. We used an average pooling with regards to the length dimension as the baseline for sequence encoders. In detail, we performed the average pooling on each embedding of TCRs with the size $l_t \times 1024$, and obtained a summarized TCR sequence representation with the size 1×1024. Similarly, we obtained a summarized epitope sequence representation with the size 1×1024. It helps to handle various lengths of TCR (or epitope) sequences by reducing the dimension of their amino acids embedding size.

Transformers: Transformer[16] is a deep learning model using an encoder-decoder structure that leverages multi-head self-attention mechanism to learn contextual representation of texts. Although it was originally designed for machine translation, it and its variants have been achieving revolutionary performances in many other natural language processing tasks such as question answering, text generation, and textual entailment.[20,26]

We use a multi-head self-attention module for sequence encoders, which is similar to Transformer encoder. The attention module allows the model to attend different amino acid residues of a TCR (or epitope) sequence based on their contextual relationship. In detail, the module takes three types of vectors as input: a query vector Q, a key vector K, and a value vector V. Each vector is defined by a linear projection of a TCR (or epitope) embedding, and each element in the projection matrix is considered as a model parameter. Then the scaled dot-product of Q and K determines the strength of contextual relationship between different

amino acid residues. The self-attention layer is then calculated by the following equation:

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V.$$

The multi-head self-attention layer is defined as a concatenation of multiple self-attention layers. Taking an embedded TCR sequence ($l_t \times 1024$) as an example, we first feed it into a multi-head attention layer with two heads followed by a dropout layer[27] with a rate of 0.1 and a layer-wise normalization.[28] The output of which is then served as the input for a feed-forward layer followed by another dropout layer with a rate of 0.1 and a layer-wise normalization. Finally, a SiLU[29] activation function followed by a global max pooling layer is used to produce a 1×1024 summarized representation for the TCR sequence. Similarly, we generate a 1×1024 sized representation for the epitope sequence.

BiLSTMs: LSTM[15] is a type of recurrent neural networks designed for dealing with long-term dependencies in sequential data and have been commonly used to process protein or genomic sequences.[11,30] Evidence has shown that BiLSTMs with max-pooling achieved overall better performance than other recurrent units such as vanilla LSTMs and GRUs[31] for sentence encoding in natural language process.[32] We therefore select a BiLSTM structure as one of our sequence encoders. A BiLSTM layer consists of two LSTM layers in opposite directions: the forward layer and the backward layer. The forward LSTM layer is used to predict the current state given previous ones by feeding the input sequence in order, and the backward LSTM layer is used for producing the current state given the future ones by feeding the input sequence reversely. In this way, a BiLSTM layer can learn features from both directions.

In detail, taking an epitope sequence with length l_e as an example, we first use a biLSTM layer with 32 units to encode the epitope sequence, followed by a time-distributed linear layer with 256 neurons. The output vector size is $l_e \times 256$. We then feed this vector to a SiLU activation function[29] and global max pooling layer as it has been shown the global max-pooling achieves better encoding results in general.[32] The final outputed representation vector is 1×256 for the epitope sequence. Similarly, the representation size for a TCR sequence is also 1×256.

CNNs: CNNs are a type of neural networks using convolution operations to extract high-level features in image processing.[33] CNNs have achieved excellent performances in many computer vision tasks involving videos or images.[34,35] A recent work suggested that CNNs could also perform well even when dealing with sequential data such as protein sequences.[36] Specifically, they trained a ByteNet-based[37] CNN model on protein data and showed that their CNN model achieved comparable performance with Transformers. We thereby design an CNN-based architecture for the sequence encoders using ByteNet.

A ByteNet block consists of 3 one-dimensional CNN layers, each of which is followed by a batch normalization[38] layer and GeLU[39] activation function. The number of filters for these three CNN layers are 256, 512, and 1024, respectively. The first and third CNN layers with both kernel sizes and stride steps being 1 are utilized to process the sequential TCR and epitope sequences. The middle CNN layer is a dilated CNN[40] with a kernel size of 5 and stride

step of 1, and it is used to expand the receptive field of input sequence covered without pooling to learn global context information. The input and output of each block are added together, and serve as the input for the next block. Four blocks are used in total. The dilation rate for the dilated CNN layer in each block increases by a factor of 2, ranging from 2 to 16.

Taking a TCR amino acids embedding ($l_t \times 1024$) as an example, we first feed it into an 1D CNN layer with 256 filters followed by a GeLU activation function and another 1D CNN layer with 512 filters. Batch normalization and a GeLU activation function are then applied. The output of which is then feed into a 1D CNN layer with 1024 filters followed by 4 continuous ByteNet blocks. We use the final output of these ByteNets as the summarized representation for TCR or epitope sequences. The size of summarized representation is 1×1024.

3.2.2. *Linear Prediction Layers*

On top of the sequence encoders, we stack two dense layers for determining the TCR-epitope binding affinity score between two sequence representations. The classifier takes a pair of summarized TCR and epitope representation vectors as the input and predicts the probability (0–1) that they are binding to each other. Taking a summarized TCR sequence representation (denoted as u) obtained from the baseline sequence encoder (size of 1×1024) as an example, a summarized epitope sequence representation is also 1×1024 size (denoted as v). We first concatenate u, v, and their absolute subtraction $|u - v|$ together, resulting in a 1×3072 input vector under baseline circumstance. The reason we include $|u - v|$ into the concatenation is that we aim to force the model to not only learn features from TCR and epitope sequences but also pay attention to the difference between them. We then feed this input vector into a linear layer with 1024 neurons, followed by a batch normalization,[38] a 0.3 rate dropout[27] and a SiLU[29] activation function. The output of which is then passed into another linear layer with a single neuron followed by a softmax function to produce a binding affinity score ranging from 0 to 1.

4. Experiments

We compared four different sequence summarizing encoders, including average pooling as baseline, our Transformer-based, BiLSTM-based, and CNN-based sequence encoders. We trained the sequence encoders together with a two-layer neural network that concatenates output representations of the encoders and predicts the binding affinity of TCR and epitope pairs.

4.1. *Implementation Details*

We trained TCR-epitope binding prediction models using adam[41] optimizer and binary cross-entropy loss with a learning rate of 0.001 and a batch size of 32. An early stopping method was used to avoid over-fitting. It stops training if the validation loss has not decreased for the last 30 epochs or the epoch become larger than 200. For each type of the sequence encoder, we listed the size of summarized representations (u for a TCR sequence and v for an epitope sequence showed in Fig. 1), as well as the total number of trainable parameters in the TCR-epitope binding affinity prediction models in Table 1. Note that the summarized representation size of

our BiLSTM-based method is 1×256, which is one fourth of other methods. We intentionally designed in this way to build a lite sequence encoder for comparison purposes. We trained each model for 10 runs and reported mean and standard deviation of AUC, precision, and recall scores. We tuned the number of heads in the multi-head attention layers and the size of binary classification layers, and selected values achieving the highest AUC in epitope split (Supplementary table 1). Each run took less than 1 day to finish on a NVIDIA RTX 2080 GPU with 11 GB memory. All our code was developed upon Tensorflow.[42]

Table 1. Summarized representation size of different sequence encoder and trainable parameters of TCR-epitope binding affinity prediction models. We show number of total trainable parameters in the prediction model and trainable parameters in encoder layers in parentheses.

Sequence Encoder Structure	Representation Size	Trainable Parameters (in encoders)
Average Pooling (Baseline)	1×1024	3,149,825 (0)
Transformer	1×1024	20,082,753 (16,932,928)
BiLSTMs	1×256	1,364,993 (574,464)
CNNs	1×1024	11,430,657 (8,280,832)

4.2. *Results and Discussion*

Our Transformer-based sequence encoder significantly outperforms the rest three methods. To visually compare performances of different sequence encoders, we showcased the ROC curves for both TCR and epitope split in Fig. 2. It was constructed by plotting the true positive rate against the false positive rate. A model is considered to have good performance if its ROC curve is close to the top-left corner. As seen in Fig. 2, we found that our transformer-based model outperformed the other three methods under both TCR and epitope split settings, indicating that it can summarize the TCR and epitope amino acids embedding better. It may be because the multi-head attention mechanism assists to learn contextual information between amino acids. We also compared the AUC, precision, and recall scores of the methods in Fig. 2. The mean values across 10 runs are shown on top of each bar in Fig. 2. The height of error bars represents the standard deviation over 10 runs. A two-sample paired t-test was carried out for statistical significance testing. A p-value less than 0.05 means a significant performance difference, otherwise, we considered it an statistically equivalent. We showed that our Transformer-based model significantly outperformed both the baseline and BiLSTM-based method in TCR and epitope split. In detail, our Transformer-based method achieved a 97.48% AUC score in TCR split, outperforming baseline and BiLSTM-based methods by 3 and 2 points, respectively. Similarly, even bigger performance gains were observed in the epitope split. The Transformer-based method reached a 89.83% AUC score which surpassed the baseline and BiLSTM-methods by around 5 and 4 points, respectively. Our comparison results suggested that Transformer-based sequence encoder can best summarize TCR (or epitope) representations.

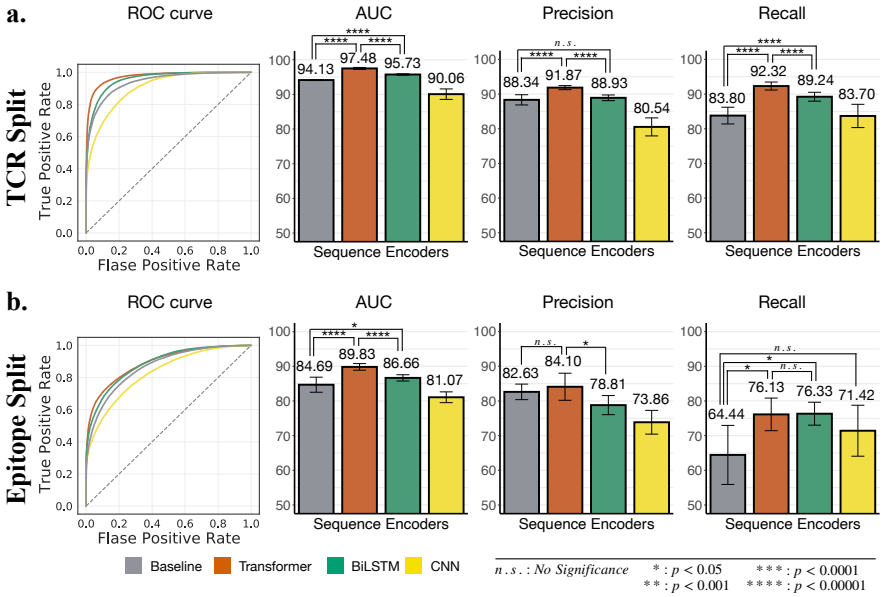

Fig. 2. Performance of the TCR-epitope binding affinity prediction models using variety of different sequence encoders in **a.** TCR split and **b.** Epitope split.

The choice of model architecture can be more important than the number of model parameters. Our BiLSTM-based method significantly outperformed the baseline method in both TCR split and epitope splits as well. As seen in Fig. 2, it achieved a 95.73% AUC score in the TCR split, which is 1 point higher than the baseline of which the number of trainable parameters are three times larger (Table. 1). We also observed that the current size of BiLSTM-based method performed similar with a larger size BiLSTM model (representation size 1×1024). The large BiLSTM model achieved an AUC of 95.52% in the TCR split, and of 87.13% in the epitope split, showing that increasing the number of parameters in BiLSTM is not a significant factor for improving the prediction performance. Moreover, we also observed that CNN may not be an optimal structure for summarizing TCR or epitope sequences. It performed significantly worse than baseline in both TCR split and epitope split. The AUC score dropped around 4 points to 90.06% and 81.07% compared to baseline in both TCR and epitope split, respectively. While the CNN-based model contains three more times parameters than the baseline method, it failed to summarize better embeddings for sequences. It may be because the the CNN-based model focused on leaning local contextual information but not on global contextual information. All those results showed that carefully selecting the neural network structure can make great improvement for TCR-epitope binding affinity prediction than simply increasing model capacities.

The Transformer-based method performs best on most individual out-of-sample epitopes. To take a closer look at our models' performance on individual unseen epitopes, we further compared AUC scores of each epitopes having the top 20 frequency in the epitope split (Table. 2). For each epitope, we highlighted the highest AUC score across four models in bold. We found that our Transformer-based method achieved the highest AUC scores in 17 out

of 20 epitopes. Apart from the first two epitopes, the Transformer-based and BiLSTM-based model surpassed the baseline for the other 18 epitopes. The CNN-based model, on the other hand, generally performed worse than baseline. Overall, the comparison results of individual epitopes was consistent with our observation in Fig 2.

Table 2. AUC scores for Top 20 frequent epitopes in testing set

Epitopes	Number of TCRs	Baseline	Transformers	BiLSTM	CNNs
MIELSLIDFYLCFLAFLLFLVLIML	23146	**74.37**	60.05	68.81	64.94
GILGFVFTL	10802	**85.93**	80.75	83	78.82
LLWNGPMAV	4716	79.75	**89.51**	87.41	75.75
LSPRWYFYYL	3502	71.19	**93.69**	80.62	78.67
VQELYSPIFLIV	2126	77.99	**92.86**	89.56	80.67
GMEVTPSGTWLTY	1990	74.88	**93.17**	86.19	76.99
ELAGIGILTV	1970	86.86	**90**	88.84	82.29
YEDFLEYHDVRVVL	1752	81.06	**96.58**	92.76	75
FLPRVFSAV	1734	78.78	**89.16**	84.49	75.38
MPASWVMRI	1558	75.61	**89.74**	81.26	75.23
FPPTSFGPL	1362	79.01	**93.18**	86.79	80.92
YEQYIKWPWYI	1074	67.88	**95.63**	87.25	77.1
VLHSYFTSDYYQLY	970	79.18	**86.5**	86.09	79.39
KTAYSHLSTSK	952	59.14	**80.68**	78.79	70.59
CRVLCCYVL	870	71.04	80.35	**80.92**	75.64
ILGLPTQTV	472	78.39	**95.34**	93.65	75.43
FIAGLIAIV	406	77.1	**93.35**	82.52	66.26
SMWSFNPETNIL	398	80.66	**92.72**	89.45	81.41
ILHCANFNV	398	80.16	**95.98**	90.46	85.18
FTISVTTEIL	396	76.27	**94.45**	88.39	80.31

5. Conclusions

This paper proposed PiTE, a pipeline that achieved a state-of-the-art performance for the TCR-epitope binding affinity prediction problem. In particular, we explored various types of neural network architectures for the sequence encoders that can be used on top of the existing embedding models. We showed that the Transformer-based method achieved the best performance. Our experimental evidence showed that the performance can be further boosted with more advanced structure of sequence encoders.

References

1. C. Szeto, C. A. Lobos, A. T. Nguyen and S. Gras, TCR recognition of peptide–MHC-I: Rule makers and breakers, *International Journal of Molecular Sciences* **22**, p. 68 (2020).
2. M. Krogsgaard and M. M. Davis, How T cells 'see' antigen, *Nature Immunology* **6**, 239 (2005).
3. M. M. Davis and P. J. Bjorkman, T-cell antigen receptor genes and T-cell recognition, *Nature* **334**, 395 (1988).

4. J. L. Xu and M. M. Davis, Diversity in the CDR3 region of VH is sufficient for most antibody specificities, *Immunity* **13**, 37 (2000).
5. C. Graham, R. Hewitson, A. Pagliuca and R. Benjamin, Cancer immunotherapy with CAR-T cells–behold the future, *Clinical Medicine* **18**, p. 324 (2018).
6. L. Zhao and Y. J. Cao, Engineered T cell therapy for cancer in the clinic, *Frontiers in Immunology* **10**, p. 2250 (2019).
7. E. Jokinen, J. Huuhtanen, S. Mustjoki, M. Heinonen and H. Lähdesmäki, Predicting recognition between T cell receptors and epitopes with TCRGP, *PLoS Computational Biology* **17**, p. e1008814 (2021).
8. S. Gielis, P. Moris, W. Bittremieux, N. De Neuter, B. Ogunjimi, K. Laukens and P. Meysman, Detection of enriched T cell epitope specificity in full T cell receptor sequence repertoires, *Frontiers in Immunology* **10**, p. 2820 (2019).
9. V. I. Jurtz, L. E. Jessen, A. K. Bentzen, M. C. Jespersen, S. Mahajan, R. Vita, K. K. Jensen, P. Marcatili, S. R. Hadrup, B. Peters *et al.*, NetTCR: sequence-based prediction of TCR binding to peptide-MHC complexes using convolutional neural networks, *BioRxiv* , p. 433706 (2018).
10. A. Montemurro, V. Schuster, H. R. Povlsen, A. K. Bentzen, V. Jurtz, W. D. Chronister, A. Crinklaw, S. R. Hadrup, O. Winther, B. Peters *et al.*, NetTCR-2.0 enables accurate prediction of TCR-peptide binding by using paired TCRα and β sequence data, *Communications Biology* **4**, 1 (2021).
11. I. Springer, H. Besser, N. Tickotsky-Moskovitz, S. Dvorkin and Y. Louzoun, Prediction of specific TCR-peptide binding from large dictionaries of TCR-peptide pairs, *Frontiers in Immunology* , p. 1803 (2020).
12. A. Weber, J. Born and M. Rodriguez Martínez, TITAN: T-cell receptor specificity prediction with bimodal attention networks, *Bioinformatics* **37**, i237 (2021).
13. M. Cai, S. Bang, P. Zhang and H. Lee, Atm-tcr: Tcr-epitope binding affinity prediction using a multi-head self-attention model, *Frontiers in Immunology* **13** (2022).
14. K. Wu, K. E. Yost, B. Daniel, J. A. Belk, Y. Xia, T. Egawa, A. Satpathy, H. Y. Chang and J. Zou, TCR-BERT: learning the grammar of T-cell receptors for flexible antigen-binding analyses, *bioRxiv* (2021).
15. S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Computation* **9**, 1735 (1997).
16. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, Attention is all you need, *Advances in Neural Information Processing Systems* **30** (2017).
17. S. Henikoff and J. G. Henikoff, Amino acid substitution matrices from protein blocks, *Proceedings of the National Academy of Sciences* **89**, 10915 (1992).
18. J.-W. Sidhom, H. B. Larman, D. M. Pardoll and A. S. Baras, DeepTCR is a deep learning framework for revealing sequence concepts within T-cell repertoires, *Nature Communications* **12**, 1 (2021).
19. P. Zhang, S. Bang, M. Cai and H. Lee, Cracking TCR-epitope interactions using language model representations, *Under review* .
20. J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (Association for Computational Linguistics, Minneapolis, Minnesota, June 2019).
21. M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee and L. Zettlemoyer, Deep contextualized word representations (2018), NAACL.
22. R. Vita, S. Mahajan, J. A. Overton, S. K. Dhanda, S. Martini, J. R. Cantrell, D. K. Wheeler, A. Sette and B. Peters, The immune epitope database (IEDB): 2018 update, *Nucleic Acids Research* **47**, D339 (2019).
23. M. Shugay, D. V. Bagaev, I. V. Zvyagin, R. M. Vroomans, J. C. Crawford, G. Dolton, E. A.

Komech, A. L. Sycheva, A. E. Koneva, E. S. Egorov *et al.*, VDJdb: a curated database of T-cell receptor sequences with known antigen specificity, *Nucleic Acids Research* **46**, D419 (2018).
24. N. Tickotsky, T. Sagiv, J. Prilusky, E. Shifrut and N. Friedman, McPAS-TCR: a manually curated catalogue of pathology-associated T cell receptor sequences, *Bioinformatics* **33**, 2924 (2017).
25. S. Nolan, M. Vignali, M. Klinger, J. N. Dines, I. M. Kaplan, E. Svejnoha, T. Craft, K. Boland, M. Pesesky, R. M. Gittelman *et al.*, A large-scale database of T-cell receptor beta (TCRβ) sequences and binding associations from natural and synthetic exposure to SARS-CoV-2, *Research Square* (2020).
26. T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, Language models are few-shot learners, *Advances in Neural Information Processing Systems* **33**, 1877 (2020).
27. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, *The Journal of Machine Learning Research* **15**, 1929 (2014).
28. J. L. Ba, J. R. Kiros and G. E. Hinton, Layer normalization, *arXiv:1607.06450* (2016).
29. S. Elfwing, E. Uchibe and K. Doya, Sigmoid-weighted linear units for neural network function approximation in reinforcement learning, *Neural Networks* **107**, 3 (2018).
30. M. Heinzinger, A. Elnaggar, Y. Wang, C. Dallago, D. Nechaev, F. Matthes and B. Rost, Modeling aspects of the language of life through transfer-learning protein sequences, *BMC Bioinformatics* **20**, 1 (2019).
31. K. Cho, B. Van Merriënboer, D. Bahdanau and Y. Bengio, On the properties of neural machine translation: Encoder-decoder approaches, *arXiv:1409.1259* (2014).
32. A. Conneau, D. Kiela, H. Schwenk, L. Barrault and A. Bordes, Supervised learning of universal sentence representations from natural language inference data, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, September 2017.
33. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel, Backpropagation applied to handwritten zip code recognition, *Neural Computation* **1**, 541 (1989).
34. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, ImageNet: A large-scale hierarchical image database, in *2009 IEEE conference on computer vision and pattern recognition*, 2009.
35. A. Krizhevsky, I. Sutskever and G. E. Hinton, ImageNet classification with deep convolutional neural networks, *Advances in Neural Information Processing Systems* **25** (2012).
36. K. K. Yang, A. X. Lu and N. K. Fusi, Convolutions are competitive with transformers for protein sequence pretraining, *bioRxiv* (2022).
37. N. Kalchbrenner, L. Espeholt, K. Simonyan, A. v. d. Oord, A. Graves and K. Kavukcuoglu, Neural machine translation in linear time, *arXiv:1610.10099* (2016).
38. S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in *International Conference on Machine Learning*, 2015.
39. D. Hendrycks and K. Gimpel, Gaussian error linear units (GELUs), *arXiv:1606.08415* (2016).
40. F. Yu and V. Koltun, Multi-scale context aggregation by dilated convolutions, in *International Conference on Learning Representations (ICLR)*, May 2016.
41. D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, in *ICLR (Poster)*, 2015.
42. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu and X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems (2015).