TTCR: Accurate TCR-Epitope Binding Affinity Prediction Using Transformers

Manna Dai, Yonghui Wu, Jun Zhou, Zhenzhou Wu, Mengren Man, Linh Le Dinh,
Joyjit Chattoraj, Feng Yang, Dao My Ha, Yong Liu, Rick Siow Mong Goh
Institute of High Performance Computing
Agency for Science, Technology and Research
Singapore 138632, Republic of Singapore
{manna_dai, wu_yonghui, zhou_jun, wu_zhenzhou, Mengren_Man, Linh_Le_Dinh}@ihpc.a-star.edu.sg
{joyjitc, yangf, daomh, liuyong, gohsm}@ihpc.a-star.edu.sg

***Abstract*—Predicting the binding affinity between a T cell receptor (TCR) and an epitope is essential in cancer immunotherapy. The existing predictor epiTCR employs Random Forest to distinguish between binding and non-binding TCR-epitopes. However, the feature representation learning capability of Random Forest is limited, resulting in unsatisfactory prediction outcomes. Therefore, there is a need to build an improved TCR-epitope predictor. In this paper, we build a predictor based on transformers, called TTCR. This model uses two different tokenizers to convert the TCR and epitope sequences into the number sequences that can be understood by the network. We embed the transformers into TTCR encoders, in order to increase the interpretability of the model. The developed models are trained and tested based on five public datasets. The experimental results demonstrate that TTCR is a powerful tool to conduct TCR-epitope binding affinity prediction.**

***Index Terms*—TCR, epitope, binding affinity, transformer, tokenizer**

I. Introduction

The adaptive immune system's core lies in T cells' ability to discern foreign entities from host cells. T cells accomplish this essential function by using their surface protein complex, known as the T cell receptor (TCR), to bind with foreign peptides that are displayed by major histocompatibility complex (MHC) molecules on the surface of host cells [1]. Epitopes, specific regions on peptides, are crucial for TCR binding. Understanding TCR-epitope binding principles is pivotal for advancing novel immunotherapy applications [2].

In cancer immunotherapy, with the current pandemic of SARS-CoV-2, a critical aspect involves evaluating which TCRs will effectively bind to the epitopes present on neoantigens. This assessment holds significant importance for designing immunotherapy treatments. Additionally, amidst the ongoing SARS-CoV-2 pandemic, the urgent need for swiftly screening potential candidate TCRs capable of binding to foreign peptides produced by pathogens has become evident. Identifying these candidate TCRs enables the rapid development of adaptable treatment approaches for diseases that pose a public health threat [3].

Established TCR-epitope bindings are documented, yet the precise biological reasons for their specificity remain elusive. TCRs can bind to multiple epitopes, and conversely, epitopes can also bind to multiple TCRs. This complexity expands the search space significantly, since individual instances could represent a unique binding or merely one of numerous potential interactions within a pair. Due to the many-to-many relationship between TCRs and epitopes, conducting manual tests of candidate TCRs against specific epitopes becomes impractical. The capability to computationally predict the binding affinity of TCR-epitope pairs is essential for swiftly advancing truly personalized immunotherapy.

The emergence of extensive public databases including TCR sequences specific to epitopes, such as VDJdb [4], [5], McPAS-TCR [6], and the Immune Epitope Database (IEDB) [7], has paved the way for computational methodologies to assess the immunogenicity of given epitope sequences. TBAdb [8] consists of numerous interactions sourced from Asian patients, a rarity in other databases, and has not been extensively utilized to train existing tools. Conversely, 10X [9] primarily comprises validated non-binding data, only a limited number of binding interactions within its dataset.

II. Related Works

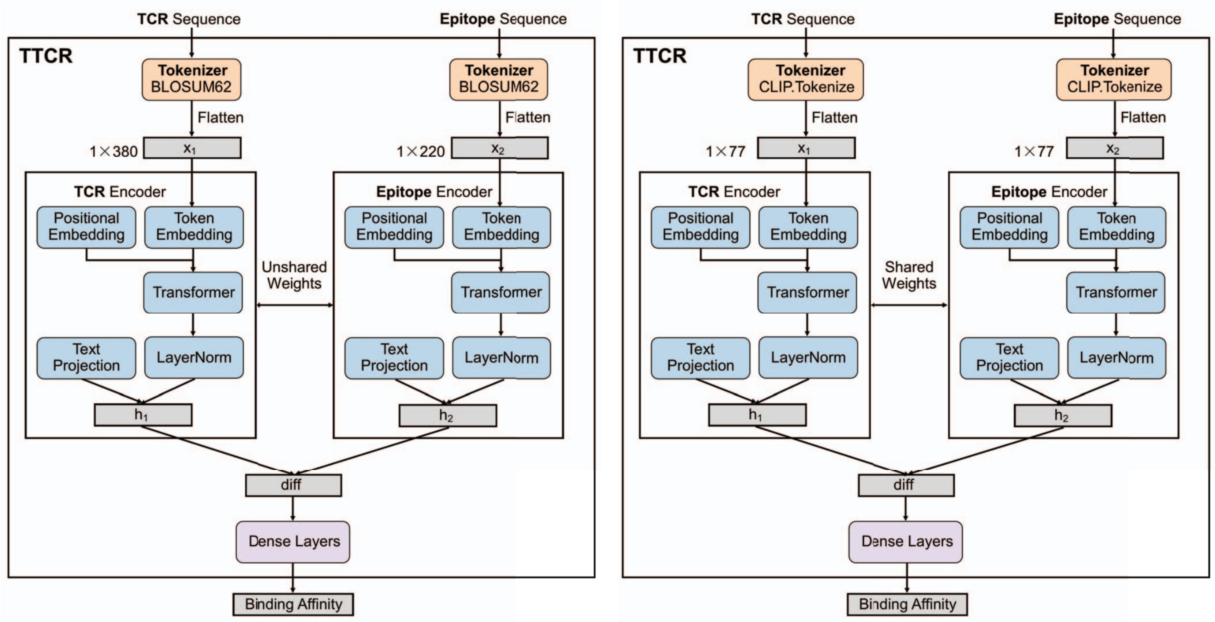
A. *TCR-epitope binding prediction via epiTCR*

The epiTCR method utilizes Random Forest as its foundation to predict TCR-epitope interactions. The epiTCR executes experiments on a combined dataset collected from 5 public datasets (IEDB, VDJdb, TBAdb, McPAS-TCR, and 10X) [10]. The inputs of the epiTCR consist of TCR region-3 beta (CDR3β) [11] sequences and epitope sequences, which are encoded by the flattened BLOSUM62 [12]. After being trained on data collected from 5 public TCR-epitope binding datasets, the epiTCR can predict the binding affinity between the given TCR and epitope sequences. Compared to other existing tools (NetTCR [1], Imrex [13], ATM-TCR [2], and pMTnet [3]), epiTCR achieves superior performance with an area under the curve (AUC) [14] of 0.98 and higher sensitivity (0.94), while maintaining comparable prediction specificity (0.9).

III. Methodology

In this paper, we build a transformer-based model named TTCR for predicting TCR-epitope binding affinity, as illustrated in Fig. 1. This prediction model is structured within a

979-8-3503-5409-6/24/$31.00 ©2024 IEEE
DOI 10.1109/CAI59869.2024.00263



(a) Model architecture of TTCR with the tokenizer BLOSUM62.

(b) Model architecture of TTCR with the tokenizer CLIP.Tokenize.

Fig. 1: A depiction of the TCR-epitope binding affinity prediction model using the transformer.

Siamese network architecture. The TTCR model takes TCR and epitope sequences as inputs, which are tokenized and converted into flattened vectors using specific tokenizers. These tokenized vectors are then passed through individual encoders, generating same-size feature vectors. By subtracting two encoded vectors, we derive a 'diff' vector (referenced in Fig. 1). This 'diff' vector is processed through dense layers to produce the binding affinity, serving as the final output of our TTCR.

Tokenizer. In natural language processing (NLP) tasks, we need to use tokenizers to convert natural language from text to numbers that machines can understand [15], [16]. Tokenization is an important step in text pre-processing, converting text into tokens, and using unique tokens to generate vocabularies. The ID of each token in the vocabulary can be represented as a number [17]–[19]. In this paper, the tokenizer is to convert a string into a number sequence, which can be understood and tackled by TTCR. We utilize two different tokenizers (BLOSUM62 [12] and CLIP.Tokenize [20]) and evaluate their contribution to TTCR performance in Section Experiments.

Given a TCR sequence and an epitope sequence, the tokenization process can be expressed as:

$$\boldsymbol{x}_1 = \mathrm{Tokenizer}(\textbf{TCR}), \tag{1}$$

$$\boldsymbol{x}_2 = \mathrm{Tokenizer}(\textbf{Epitope}), \tag{2}$$

where $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ represent the flattened tokens of the input TCR sequence and the epitope sequence. $\mathrm{Tokenizer}(\cdot)$ is the tokenizer that converts the string into a number sequence.

Encoder. We employ the transformer-based text encoder from the contrastive language-image pre-training (CLIP) model [20] as the backbone for our encoders. As the lengths of BLOSUM62 tokenized vectors are different, the TCR encoder and epitope encoder use unshared weights during TTCR training. When using CLIP.Tokenize as tokenizers, the TCR encoder and epitope encoder share the network weights during TTCR training, since the lengths of tokenized vectors are same-size.

Inspired by CLIP, we modified its text encoder as the TCR encoder and the epitope encoder. The encoder takes $\boldsymbol{x}$ as input and generates the feature vector $\boldsymbol{h}$:

$$\boldsymbol{h} = \mathrm{Encoder}(\boldsymbol{x}) \in \mathbb{R}^{C}, \tag{3}$$

where C is the dimension of the feature vector, and it is set to 8.

As depicted in Fig. 1, this encoder consists of 5 parts, including a token embedding, a positional embedding layer, a transformer model, a layer normalization, and a text projection layer. The process of token embedding is defined as follows:

$$\boldsymbol{x}_{\boldsymbol{in}}' = \mathrm{TokenEmbedding}(\boldsymbol{x}) \in \mathbb{R}^{L\times W}, \tag{4}$$

where $\mathrm{TokenEmbedding}(\cdot)$ denotes the token embedding for the flattened TCR or epitope token. L and W denote the length of a TCR/epitope token (see Table II) and transformer width (see Table I), respectively.

To introduce the positional information into the transformer, we also add learnable parameters $\boldsymbol{c} \in \mathbb{R}^{L\times W}$ to represent the positional embedding. The input $\boldsymbol{x_{in}}$ of the transformer model is as follows:

$$\boldsymbol{x_{in}} = \boldsymbol{c} + \boldsymbol{x'_{in}} \in \mathbb{R}^{L\times W}, \tag{5}$$

Then, the transformer can embed the input into a common representation space:

$$\boldsymbol{x_{out}} = \mathrm{Transformer}(\boldsymbol{x_{in}}), \tag{6}$$

A text projection layer consists of learnable parameters $\boldsymbol{cc} \in \mathbb{R}^{W\times D}$, where W and D denote the transformer width and the embed dimension (see Table I), respectively. This layer is responsible for mapping the feature map into a smaller dimension.

We utilize a layer normalization to normalize the embedding vector of a token, which is defined as follows:

$$\boldsymbol{e} = \mathrm{LayerNorm}(\boldsymbol{x_{out}}), \tag{7}$$

The normalized token embedding is combined with the text projection and generates the feature vector $\boldsymbol{h}$ by using the Hadamard Product:

$$\boldsymbol{h} = \boldsymbol{e}@\boldsymbol{cc}, \tag{8}$$

where @ indicates the Hadamard Product.

We use the above method to build the TCR encoder and the epitope encoder. By feeding the flattened TCR token $\boldsymbol{x_1}$ and the flattened epitope token $\boldsymbol{x_2}$, their respective encoders can produce the feature vector $\boldsymbol{h_1}$ and $\boldsymbol{h_2}$. The distance **diff** between two feature vector are calculated via the subtraction operation:

$$\mathbf{diff} = |\boldsymbol{h_1} - \boldsymbol{h_2}|, \tag{9}$$

where $|\cdot|$ denotes the absolute value operation.

A dense layer is added to match the dimension of the binding affinity. We use a 1×2 matrix to represent this binding affinity, which consists of the probabilities of binding and non-binding. A high probability of binding means that the TCR and epitope have a binding relationship and vice versa.

During training, we minimize a binary cross-entropy loss between the generated prediction P and ground truth y, which is defined as follows:

$$\mathcal{L} = \boldsymbol{y}\log(\boldsymbol{P}) + (1-\boldsymbol{y})\log(1-\boldsymbol{P}), \tag{10}$$

where $\boldsymbol{y}$ and $\boldsymbol{P}$ are the label and predicted probability of binding, respectively.

IV. Experiments

Datasets. To train our TTCR, we used five public datasets from work [10], including TBAdb [8], VDJdb [4], [5], McPAS-TCR [6], IEDB [7], and 10X [9]. Only 10X provides negative paired data (non-binding TCR-epitope pairs), and others are positive data (binding TCR-epitope pairs). As summarized in Fig. 2, the experimental dataset consisted of 5649 TBAdb pairs, 29737 VDJdb pairs, 5869 McPAS-TCR pairs, 64986 IEDB pairs, and 106241 10X pairs. This combined dataset totally comprised 212482 pairs with a 1:1 ratio of positive and negative pairs.

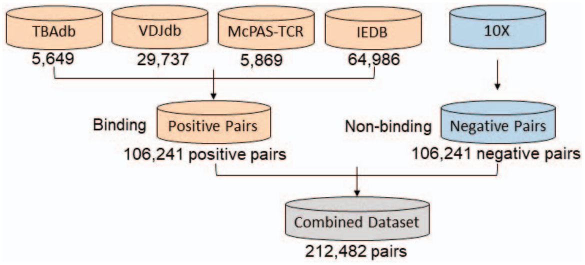
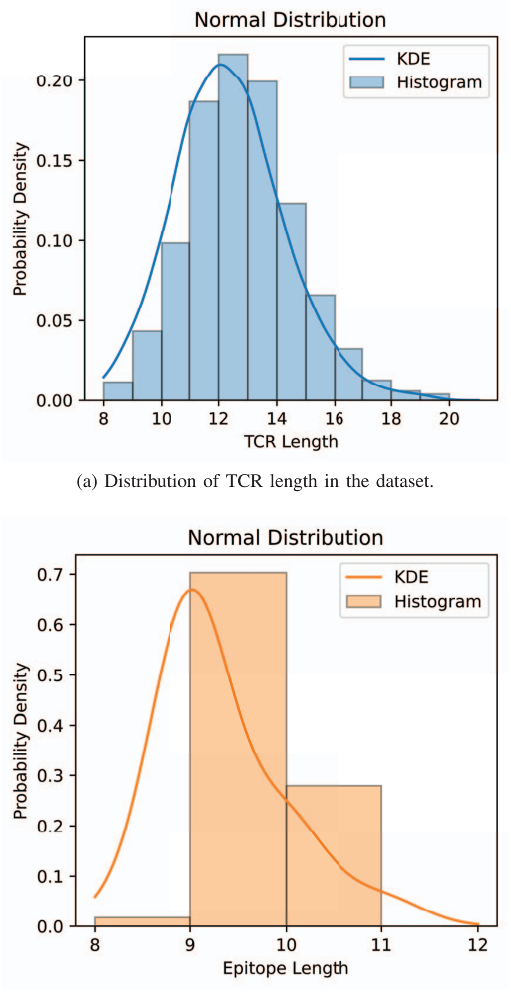


Fig. 2: The training and testing datasets consist of TCR-epitope pairs collected from TBAdb, VDJdb, McPAS-TCR, IEDB, and 10X.



(a) Distribution of TCR length in the dataset.

(b) Distribution of epitope length in the dataset.

Fig. 3: Length frequency analysis for the experimental dataset by kernel density estimators (KDEs) and histograms.

TABLE I: Hyper-parameters of our TTCR that cooperates with existing tokenizers in experiments.

Model	Tokenizer	Embed Dim	Transformer Width	Transformer Heads	Transformer Layers
TTCR+BLOSUM62	BLOSUM62 [12]	8	8	4	1
TTCR+CLIP.Tokenize	CLIP.Tokenize [20]	128	128	8	3

TABLE II: Hyper-parameters of tokenization methods in experiments.

Tokenizer	Length of TCR Tokens	Length of Epitope Tokens	Vocabulary Size
BLOSUM62 [12]	380	220	16
CLIP.Tokenize [20]	77	77	49408

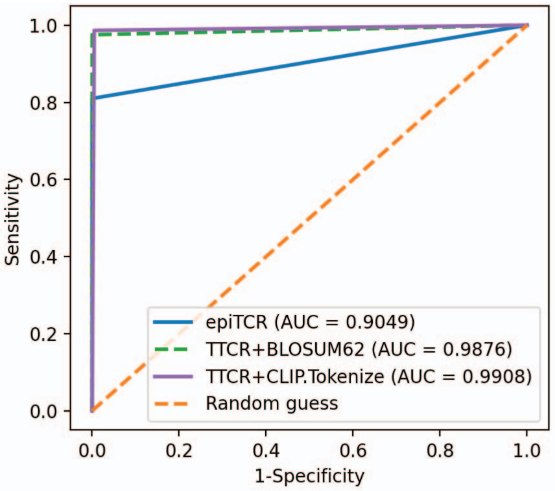
TABLE III: Performance comparison of different models using 10-fold cross-validation.

Model	Mean Accuracy	Mean Precision	Mean Sensitivity	Mean Specificity	Mean F1-Score	Mean AUC
epiTCR	0.9027	1.0	0.8053	1.0	0.8922	0.9027
TTCR+BLOSUM62	0.9205	0.9172	0.9688	0.8714	0.9347	0.9201
TTCR+CLIP.Tokenize	0.9892	0.9952	0.9831	0.9953	0.9891	0.9892

Fig. 3a displays the length distribution of TCR sequences, ranging from 8 to 19. Kernel density estimators (KDEs) [21] portray the probability density of TCR lengths, showing a concentration around the length of 12. Additionally, Fig. 3b shows that the lengths of epitope sequences range between 8 and 11, with the highest probability density observed at length 9. These findings highlight the length diversity of TCR and epitope sequences in the combined dataset.

Evaluation Metric. To evaluate the prediction performance of TTCR, we use the standard metrics in the work [10]: accuracy [22], precision [23], sensitivity [24], specificity [25], F1-Score [26], and AUC [14]. Accuracy measures the overall correctness of predictions made by the model. It refers to the ratio of correctly predicted samples (both true positives and true negatives) to the total number of samples. Precision quantifies the accuracy of positive predictions made by the model. It calculates the ratio of correctly predicted positive samples to the total predicted positive samples (both true positives and false positives). Sensitivity, also known as recall, measures the model's ability to correctly identify positive (or true) instances among all actual positives. It's calculated as the ratio of true positives to the sum of true positives and false negatives. Sensitivity assesses how well a model avoids false negatives. Specificity gauges the model's ability to correctly identify negative samples among all actual negatives. It's computed as the ratio of true negatives to the sum of true negatives and false positives. Specificity measures how well a model avoids false positives. The scale ranges from 0 to 1, where higher values signify superior model performance in both precision and recall aspects.

Implementation Details. In our training of TTCR, we employ two distinct tokenization methods, namely BLOSUM62 and CLIP.Tokenize. The specific parameters used for TTCR and the tokenization methods in our experiments are outlined in Tabel I and Tabel II, respectively. To assess its performance, we select epiTCR as the comparative method. To ensure fairness in comparison, we choose ten-fold cross-validation for experimental validation.



(a) ROC plot of prediction performance for epiTCR.

Fig. 4: The performance of epiTCR, TTCR+blosum62, and TTCR+CLIP.Tokenize for predicting TCR-epitope binding affinity on a subdataset.

Comparisons. Table III shows the performance comparison of different models that are evaluated by the ten-fold cross-validation method. Our TTCR method outperforms the epiTCR, especially in terms of mean accuracy, mean sensitivity, F1-score, and AUC. Moreover, when comparing the contribution of two different tokenizers, TTCR with CLIP.Tokenize performance better than TTCR with BLOSUM62. The TTCR with a high mean accuracy, so that the TTCR outperforms the epiTCR in terms of identifying binding or non-binding relationships. The mean precision of the epiTCR is high, but its sensitivity is lower than our proposed TTCR. These results demonstrate that the epiTCR is prone to predict the relationship between the given TCR and the epitope as binding. However, in practical applications, there is a greater need for a predictor that can identify the non-binding relationships between TCRs and epitopes. The sensitivity and specificity of our proposed TTCR are higher than the epiTCR, so the TTCR can successfully identify a binding TCR-epitope as a positive pair, and identify a non-binding TCR-epitope as a negative pair.

Fig. 4 depicts the performance of TCR-epitope binding prediction using epiTCR, TTCR with BLOSUM6, and TTCR with CLIP.Tokenize on a testing subdataset. Notably, all three models exhibit superior performance compared to random guessing. Based on the comparison results, the proposed TTCR model shows good prediction with a mean AUC>0.98, whereas epiTCR achieves a mean AUC=0.9049. We further evaluate our TTCR with two distinct tokenizers, including BLOSUM62 and CLIP.Tokenize, and TTCR with CLIP.Tokenize performs better than BLOSUM62 in our testing dataset. This comparison emphasizes that TTCR with CLIP.Tokenize demonstrates the most promising outcomes, displaying the fewest occurrences of False Positives and False Negatives. That means CLIP.Tokenize is also a potential tool for converting strings into protein language which can be understood by the deep learning method.

V. Conclusion

This paper introduces the TTCR model, designed specifically for predicting the binding affinity between TCRs and epitopes. Leveraging the transformer framework derived from CLIP as its foundational encoder, this model encapsulates biological information from input sequences into uniform-sized feature vectors. Tokenizers are employed to convert TCR and epitope sequences from textual strings into numerical sequences, facilitating comprehension by the model. Through empirical experimentation and evaluation, the findings substantiate the efficacy of TTCR as a dependable tool for predicting the binding affinity between TCRs and epitopes. Additionally, devising strategies to handle the long-tail distribution of epitope-associated TCRs remains an open problem, which is an interesting future work.

Acknowledgment

This work was financially supported by the A*STAR AME Programmatic Fund (Grant No. Grant A20H5b0142) and the A*STAR Central Research Fund “A Secure and Privacy Preserving AI Platform for Digital Health”.

References

[1] 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*, vol. 4, no. 1, p. 1060, 2021.

[2] 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*, vol. 13, p. 893247, 2022.

[3] T. Lu, Z. Zhang, J. Zhu, Y. Wang, P. Jiang, X. Xiao, C. Bernatchez, J. V. Heymach, D. L. Gibbons, J. Wang *et al.*, “Deep learning-based prediction of the t cell receptor–antigen binding specificity,” *Nature machine intelligence*, vol. 3, no. 10, pp. 864–875, 2021.

[4] 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*, vol. 46, no. D1, pp. D419–D427, 2018.

[5] D. V. Bagaev, R. M. Vroomans, J. Samir, U. Stervbo, C. Rius, G. Dolton, A. Greenshields-Watson, M. Attaf, E. S. Egorov, I. V. Zvyagin *et al.*, “Vdjdb in 2019: database extension, new analysis infrastructure and a t-cell receptor motif compendium,” *Nucleic Acids Research*, vol. 48, no. D1, pp. D1057–D1062, 2020.

[6] 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*, vol. 33, no. 18, pp. 2924–2929, 2017.

[7] 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*, vol. 47, no. D1, pp. D339–D343, 2019.

[8] W. Zhang, L. Wang, K. Liu, X. Wei, K. Yang, W. Du, S. Wang, N. Guo, C. Ma, L. Luo *et al.*, “Pird: pan immune repertoire database,” *Bioinformatics*, vol. 36, no. 3, pp. 897–903, 2020.

[9] 10x Genomics, “A new way of exploring immunity–linking highly multiplexed antigen recognition to immune repertoire and phenotype,” *Tech. rep*, 2019.

[10] M.-D. N. Pham, T.-N. Nguyen, L. S. Tran, Q.-T. B. Nguyen, T.-P. H. Nguyen, T. M. Q. Pham, H.-N. Nguyen, H. Giang, M.-D. Phan, and V. Nguyen, “epitcr: a highly sensitive predictor for tcr–peptide binding,” *Bioinformatics*, vol. 39, no. 5, p. btad284, 2023.

[11] A. Weber, J. Born, and M. Rodriguez Martínez, “Titan: T-cell receptor specificity prediction with bimodal attention networks,” *Bioinformatics*, vol. 37, no. Supplement_1, pp. i237–i244, 2021.

[12] S. Henikoff and J. G. Henikoff, “Amino acid substitution matrices from protein blocks.” *Proceedings of the National Academy of Sciences*, vol. 89, no. 22, pp. 10 915–10 919, 1992.

[13] P. Moris, J. De Pauw, A. Postovskaya, S. Gielis, N. De Neuter, W. Bittremieux, B. Ogunjimi, K. Laukens, and P. Meysman, “Current challenges for unseen-epitope tcr interaction prediction and a new perspective derived from image classification,” *Briefings in Bioinformatics*, vol. 22, no. 4, p. bbaa318, 2021.

[14] A. M. Carrington, D. G. Manuel, P. W. Fieguth, T. Ramsay, V. Osmani, B. Wernly, C. Bennett, S. Hawken, O. Magwood, Y. Sheikh *et al.*, “Deep roc analysis and auc as balanced average accuracy, for improved classifier selection, audit and explanation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 329–341, 2022.

[15] S. Qian, Y. Zhu, W. Li, M. Li, and J. Jia, “What makes for good tokenizers in vision transformer?” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[16] M. Tschannen, B. Mustafa, and N. Houlsby, “Clippo: Image-and-language understanding from pixels only,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 11 006–11 017.

[17] P. Wang, C. Da, and C. Yao, “Multi-granularity prediction for scene text recognition,” in *European Conference on Computer Vision*. Springer, 2022, pp. 339–355.

[18] J. D. Havtorn, A. Royer, T. Blankevoort, and B. E. Bejnordi, “Msvit: Dynamic mixed-scale tokenization for vision transformers,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 838–848.

[19] X. Li, Y. Ge, K. Yi, Z. Hu, Y. Shan, and L.-Y. Duan, “mc-beit: Multi-choice discretization for image bert pre-training,” in *European Conference on Computer Vision*. Springer, 2022, pp. 231–246.

[20] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, “Learning transferable visual models from natural language supervision,” in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[21] Y. Zheng, Y. Ou, A. Lex, and J. M. Phillips, “Visualization of big spatial data using coresets for kernel density estimates,” *IEEE transactions on big data*, vol. 7, no. 3, pp. 524–534, 2019.

[22] B. Juba and H. S. Le, “Precision-recall versus accuracy and the role of large data sets,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 4039–4048.

[23] F. Thabtah, S. Hammoud, F. Kamalov, and A. Gonsalves, “Data imbalance in classification: Experimental evaluation,” *Information Sciences*, vol. 513, pp. 429–441, 2020.

[24] A. Das, S. K. Mohapatra, and M. N. Mohanty, “Design of deep ensemble classifier with fuzzy decision method for biomedical image classification,” *Applied Soft Computing*, vol. 115, p. 108178, 2022.

[25] A. Pal and V. Kumar, “Agridet: Plant leaf disease severity classification using agriculture detection framework,” *Engineering Applications of Artificial Intelligence*, vol. 119, p. 105754, 2023.

[26] D. Chicco and G. Jurman, “The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation,” *BMC genomics*, vol. 21, no. 1, pp. 1–13, 2020.