Determining epitope specificity of T-cell receptors with Transformers

by

Abdul Rehman Khan

in partial fulfillment of the requirements for the degree of

Master of Science
in Computer Science – Data Science and Technology Track

at Delft University of Technology
to be defended publicly on Monday March 21st, 2022 at 10:00.

Student number: 5135028
Project duration: August, 2021 – March, 2022
Thesis committee: Prof. dr. ir. M.J.T. Reinders, TU Delft, supervisor, committee chair
Dr. I. Khatri, Leiden University Medical Center, daily supervisor
Dr. Pradeep Kumar Murukannaiah, TU Delft, external committee member

TU Delft Delft University of Technology

Determining epitope specificity of T-cell receptors with Transformers.

A.R.Khan (5135028)

Supervisors: Dr Indu Khatri[1], Prof.dr.ir. MJT Reinders[2]

[1]Department of Immunology, Leiden University of Medical Center.

[2]Pattern Recognition and Bioinformatics, Delft University of Technology

Abstract

Motivation: Transformers have dominated the field of natural language processing due to their competency in learning complex relationships within a sequence. Reusing a pre-trained transformer for a downstream task is known as Transfer learning. Transfer learning restricts the transformer to a fixed vocabulary; modification in transformer implementation will extend the utility of the transformer. Implementing transformers for complex biological problems can be beneficial in addressing the complexities in the biological sequences. One such biological problem is to capture the specificity of diverse T-cell repertoire to the unique antigens (i.e., immunogenic pathogenic elements). Using transformers to assess the relationship between T-cell receptors and antigen at the sequence level can provide us with better insights into the processes involved in these precise and complex immune responses in humans and murine.

Method: In this work, we determined the specificity of multiple TCR to unique antigens by classifying the CDR3 regions of TCR sequences to a particular antigen. For this problem, we used three pre-trained auto-encoder (ProtBERT, ProtALBERT, ProtELECTRA) and one pre-trained auto-regressive (ProtXLNet) transformer model wherein, to adapt to the challenges of the complex biological problem at hand, we implemented modifications in the transformers chosen here. We used the VDJdb to obtain the biological data for training and testing the selected transformers. After pre-processing data, we predicted the TCR specificity for 25 antigens (classes) in a multi-class setting.

Results: Transformers could predict the specificity of TCRs to an antigen with just the CDR3 sequences from the TCRB chain (weighted F1 score 0.48), the data that was unseen by the transformers. With additional features incorporated, i.e., gene names for TCRs, the weighted F1 improved to 0.55 in the best performing transformer. We demonstrated that different modifications in transformers recognized out-of-vocabulary features with these results. When comparing the AUC from the transformer model to other previously developed methods for the same biological problem such as TCRGP, TCRDist and DeepTCR, we observed that the transformers outperformed the previously available methods. To exemplify, the MCMV epitope family that suffered from restricted performance in TCRGP due to fewer training samples (~100 samples) showed 10% improvement in AUC with transformers under similar training samples.

Conclusion: Transformer's proficiency in learning from fewer data combined with holistic modifications in transformers implementations proves that we can extend its capabilities to explore other biological settings. Further ingenuity in utilizing the full potential of transformers either through attention head visualization or introducing additional features can further extend T-cell research avenues.

Availability: Dataset and scripts will be available on github.com/arkhan19/tcrformer

Contact: a.r.khan@student.tudelft.nl

Supplementary information: Supplementary data is separately available with this document.

1 Introduction

The human's immune system can mount the immune response by generating multiple T-cell receptors (TCR) in response to a pathogenic infection. Principally, this response involves interaction between T-cell receptors (antigen-recognition receptors on T cells) and the epitopes (short peptides from pathogenic proteins) from proteins present in infectious agents (bacteria/viruses). Complementarity determining region 3 (CDR3) on both α and β chains of TCR bind with the epitope, and recognition of a highly diverse range of antigens is thus due to the remarkable specificity of CDR3 to each antigen (Robins, Campregher et al. 2009). Epitope specificity of a TCR is defined by the unique CDR3 sequence specific to a particular T cell and its progeny. The diversity is estimated to be approximately 10^{18} in humans and 10^{15} in mice. The high specificity of the CDR3 region results from the hypervariability of the CDR3 region imparted during V(D)J recombination and junctional diversity (See Appendix 1 for more detail). Due to this reason, not only does TCR have a clonotypic nature, but it is also the focus when determining TCR specificity (Petrova, Ferrante et al. 2012). Determining the specificity of such highly diverse and variable TCR sequences will require extracting patterns from an enormous search space. Learning specificity of these se-

quences will provide a broad range of applications in immunological studies (Attaf, Huseby et al. 2015); however, with a caveat to decipher the immense diversity of CDR3 sequences found in TCRs.

Translating data into information has fascinated linguists and mathematicians alike, which led them to leverage their skills to delegate translation tasks to machines (Hutchins 2007). Linguistic expertise in finding similarities in source and target language was converted to mathematical equations by mathematicians. Neural networks for machine translation (Schwenk 2007) has opened up avenues for extending their support in understanding biological interactions, e.g. identifying binding sites in proteins or assessing protein-protein interactions.

Machine translation (or MT) poses a unique problem for any neural network, where the encoding of the source sequences plays a vital role to reach an accurate translation of a source sequence to a target sequence. The inception of many innovative translation methodologies (Kalchbrenner and Blunsom 2013) (Sutskever, Vinyals et al. 2014) (Cho, Van Merriënboer et al. 2014) revolved around the sphere of providing the best possible representation of input data using either information about placement or information about the surrounding words of the word to encode. Effectively ascertaining relationships of each word to one another led towards conceptualizing the relevancy of a word towards accurate translation (Koehn 2017). A famous quote that goes around in the MT community, "You shall know a word by the company it keeps", best describes the significance of relevant words in encoding a word. The objective of mathematically providing relevancy to neural networks led to the eventual introduction of the alignment model (Bahdanau, Cho et al. 2014); which was used while translating source sequence to any target sequence to account for relevant relationships within source sequence.

An alignment model or *attention* focuses the neural network on learning relevant relationships, reducing heavy translations and improving translation performance. One of the deep neural architectures using attention is Transformers (Vaswani, Shazeer et al. 2017). Unlike long short-term memory (LSTM) models, transformers are not restricted by the input sequence length. Transformers are trained using Transfer Learning, in which transformer models are first pre-trained on task-analogous objectives and finally fine-tuned on task-oriented objectives (See Appendix 2 for more detail on transformers). Through transfer learning, a transformer can learn contextual information from a large dataset during pre-training and then apply the knowledge learnt from pre-training towards a downstream task, which generally has scarcity in data. Since the introduction of transformers, many variations of its architecture have been released. Common in all these transformer architectures is the *self-attention* mechanism.

Attention mimics the psychology of our brain when it is paying attention to only certain parts of the information to reach a conclusion (Lindsay 2020). Self-attention, on the other hand, supplements such complexity by adding *context*. While attention focuses on parts of the input that leads to a better conclusion, self-attention will focus on surrounding words to delineate meaning between similar words. On the other hand, multi-headed self-attention learns multiple representations of each token, each of these are termed as heads. Different transformer models implement different strategies to utilise attention mechanisms more efficiently.

Remarkably, Transformers are not just limited to natural languages; given sufficient corpus, they can be tailored towards any sequence-based inference. ProtTrans are transformer models trained on a large corpus of protein data (Elnaggar, Heinzinger et al. 2020), and they prove that transformers can learn the contextual grammar of amino-acid sequences. Analogous to NLP tasks, the ProtTrans pre-training task involved a vocabulary of twenty amino acids. Therefore, sentences and words are analogous to protein sequences and amino acids. Four of these Transformer models, three auto-encoder (ProtBERT, ProtAlbert, ProtElectra) and an auto-regressive model (ProtXLNet), are publicly available as of November 2021. Attention mechanism's ability to find the most relevant regions in a protein sequence (pattern of amino acids) will accurately emulate knowledge needed to understand the mechanisms behind a protein function. Transformers can potentially learn patterns of a TCR-epitope interaction and understand the process that determines the affinity between a TCR and an epitope.

This work will demonstrate the transformer's feasibility in determining TCR specificity to the antigens using the CDR3 protein sequences. Additionally, we will confront the inherent challenge of imbalanced classes in the TCR-epitope data. Additionally, transformers are infamous for accepting just input sequence albeit of variable length. Due to inherent complexities in any biological interaction, additional features are indispensable; therefore, it was pertinent to find a way to circumvent this restriction in transformers. As it is known that the gene names of the TCR can impart some level of specificity to recognize an antigen (Hodges, Krishna et al. 2003), we will perform holistic transformer modification to include additional features while preserving pre-trained knowledge. Finally, we will compare our results with the previously available tools implemented using distance metrics or deep learning to address the same problem of TCR specificity.

2 Methods

2.1 Data acquisition and preparation

VDJdb (Shugay, Bagaev et al. 2018) provides a centralised source from where we can retrieve the data for TCR-epitope pairs. For a given TCR sample, the database provides CDR3 sequences of TCRβ and/or TCRα gene; TCR βV and/or TCRαV gene; TCRβJ and/or TCRαJ genes, MHC Class I/II and organism. In addition, the database provides the epitope sequence, parent gene of the epitope, and the antigen species for the epitope. The confidence score associated with each TCR-epitope pair was provided wherein zero scores indicate that the sequences were only obtained by the sequencing technologies without any support from wet-lab experiments; however, higher scores (1, 2, and 3) would represent the pairs that were validated using one or more than one wet-lab based techniques. The database provides information regarding how the TCR-epitope pairs were validated, e.g., assay identification, TCR sequencing or verification procedure.

Data obtained from VDJdb consists of 81,762 entries, which includes data from various sources (including (Dash, Fiore-Gartland et al. 2017), (Sidhom, Larman et al. 2021)). We used the complete dataset (updated on November 2021) and applied our filtering criterion inspired by prior works. We utilised only TCR-β chains of Human and Murine antigens in this work with a confidence score greater than 0. We also removed duplicate CDR3 sequences with the same V-gene, J-gene, MHC A, MHC B, ensuring unique data entries for each label are used for training. Subsequently, we removed certain epitope species resulting in autoimmune disorders or allergies. Table 1 summarises the filtration process applied before providing the data to the transformers.

Unique categories of each V and J gene, rather than allele, were retained. The ordinal encoder encodes genes (63 V genes and 13 J genes) associated with each sequence. Labels of input data are created by concatenating Epitope's species, genes, and sequence, then labels with more than 50 samples were retained. The dataset is divided into 70% training, 15% validation, and 15% testing dataset. Out of 25 labels, as shown in

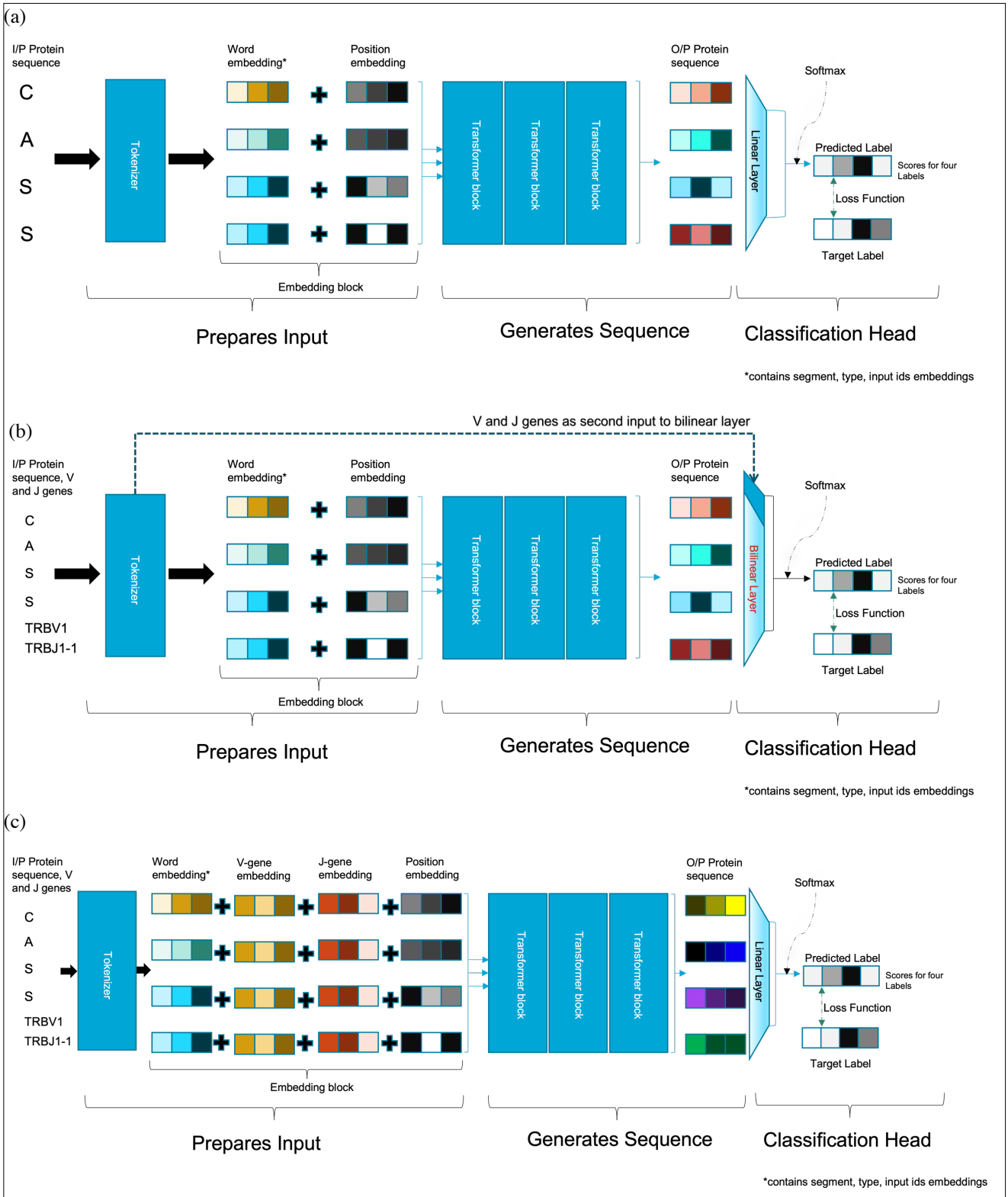

Fig. 1. ProtTrans Overview Data flow common in all Transformers. (a) CDR3 sequence is provided. (b) Sequence provided to transformer block and gene usage information directly to classification head. (c) Sequence along with gene usage information from embedding block is provided to transformer block.

Fig. 2, 10 were more than 100 samples, and 15 were less than 100 samples; the former is considered the easy-to-classify label and later is considered hard-to-classify label for this work.

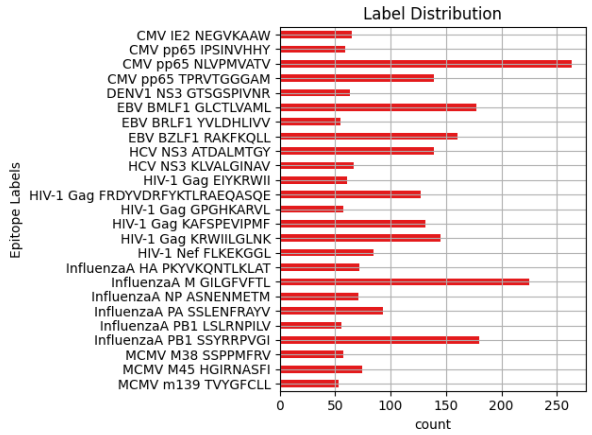

Fig.2. Label distribution. Post-filtering epitope label distribution of data.

Table 1. Data filtering on VDJdb data. The final sample count indicates sample count before splitting of data.

S. no.	Filtration criteria	Sample count
1	Raw data	81762
2	Post removing N/A	80679
3	Selecting only human and murine antigen	78647
4	Selecting TCR-epitope pairs with confidence score > 0	9580
5	Post removing allergen and cancer antigen	8547
6	Post removing duplicated entries*	5680
7	Selecting the TCR-epitope pairs with TRB sequences	4057
8	Selecting the TCR-epitope pairs with >=50 samples	2674

*Duplicates are removed based on the same gene and MHC gene values and CDR3 sequence.

2.2 Multi-class classification

ProtTrans are built using Hugingface framework (Wolf, Debut et al. 2019) and pre-trained on Uniref (Suzek, Wang et al. 2015) and BFD (Steinegger and Söding 2018, Steinegger, Mirdita et al. 2019) datasets. We used models trained on uniref except for ProtElectra as its uniref version was not available. Even though CDR3 sequences are protein sequences, there are challenges in training ProtTrans with just CDR3 sequences. Predominantly, the filtration we applied in the previous section has left us with highly imbalanced data. High imbalance can lead to poor model training, not generalising well on unseen data. Another challenge is that transformers could only ingest data that complies with the vocab on which it was pre-trained, in this case, amino-acid sequences. Additional features will be out of vocabulary but contain pertinent information that can aid in classification.

Gene usage has proven to contribute significantly to the epitope specificity of a TCR (Hodges, Krishna et al. 2003). These features are out of vocabulary for our pre-trained models. At the same time, additional features will not help us if we cannot handle an imbalance in our TCR data. Therefore, this leaves us discovering ingenious ways to address issues complicating our task. We addressed these challenges in the following two ways: (1) including a modified loss function (Section 2.2.1) and (2) modifying model implementations to include additional features along with input protein sequences (Section 2.2.2).

2.2.1 Loss Function for Imbalance data

Imbalance in our dataset prompted us to either make compensation in the dataset population or replace the loss function. Compensating datasets through oversampling would defeat the purpose of removing duplicate sequences from the dataset. Therefore, we address the imbalance in the data by overriding our loss function.

A binary loss function created upon cross-entropy is Focal-loss (Lin, Goyal et al. 2017) which adds a modulating factor (or Focusing parameter) to cross-entropy. Modulating factor is adjusted using gamma ($\gamma > 0$), which allows diminishing the contribution of easily classified labels (i.e., samples belonging to the majority class) and enhancing the contribution of hard to classify labels (i.e., samples belonging to minority class) to the loss value. Enhanced loss value of hard-to-classify labels informs the model to accommodate more for these labels, which otherwise would have been overlooked during training. The loss appears normal to the model; the adjusted loss value is the manipulation performed before providing it to the model, and the model can then direct the gradient accordingly.

$$\text{CE} = -\sum_{i=0}^{c} t_i \log\left(f(s)_i\right) \qquad (1)$$

$$\text{FL} = -\sum_{i=0}^{c} t_i \left(1 - f(s)_i\right)^{\gamma} \log\left(f(s)_i\right) \qquad (2)$$

Multi-class focal loss, which is just an extension of multi-class cross-entropy, is required for our problem. Multi-class cross-entropy (Eq. 2) calculates the sum of losses for each class, c, extending it to focal loss would require applying a modulating factor to the sum of losses obtained. A recommendation of $\gamma = 2$ was made by the authors of focal loss; however, they only assumed binary cases. Therefore, we will use γ as hyperparameter in our model optimisation to estimate which value works best for our data.

2.2.2 Gene usage in Transformers

Each transformer has its specific tokeniser, which would prepare input using its pre-trained vocabulary for a transformer to process. A general overview of the transformer model is depicted in Fig 1. (a). The embedding block prepares input for the transformer, which includes adding special tokens to denote special relationships (such as padding), an attention mask to denote if a token is to be considered for attention calculation or segment ID to denote two separate sequences in the same input. The tokeniser in the embedding block also performs encoding of labels and presents sequences to the transformer block. Each tokeniser maintains homogeneity of the sequence encoding and presents each sample (or in batches) to the transformer to learn a representation.

The representation of the input sequence is fed to a classification block, which will perform sequence inference tasks such as classification. This highlights a constraint in transfer learning regarding the vocabulary of a transformer model. If we want to allow the transformer to understand new information, it will take a considerable amount of time and data to pre-train and learn new data (See Appendix 2: Transfer Learning). Fine-tuning with just new information, in our case, gene usage as features can circumvent the issue of training the entire transformer again. Once the transformer block accepts the information, we cannot alter the data transformation, which is based mainly on pre-

trained knowledge. We will ruin the benefits of transfer learning and diminish the performance of the transformers.

The tokenizer can be altered to accommodate additional features along with sequences, leading to two potential sites to introduce gene usage information, either before or after the transformer blocks. Former will require modification in the classification block of the model (Fig 1 b), while later will require modification in the embedding block (Fig 1. c). Both methods rely on the tokeniser of each transformer model to encode gene features along with amino acid sequences. Weaving gene usage into either classification block or embedding block would require associating a sequence to both V and J genes to predict specificity to an epitope.

Gene usage in classification block

In a standard transformer (Fig 1. a), the classification block receives an n-dimensional (n is 768 in the case of ProtBert) representation of each amino acid as input from the final hidden state of the transformer block (Devlin, Chang et al. 2018). The first token of every sequence is a special classification token [CLS] which contains the aggregated representation of the input sequence in the final hidden state. This representation is termed as pooled output. The pooled output is the input to the classification block to perform the sequence inference task.

In classification block modification, the classification block receives gene information directly from the tokenizer, bypassing the transformer block (Fig.1.b). Each CDR3 sequence has V and J genes associated with it. An ordinal encoding for genes is provided to the tokenizer, which it associates with each sample and then given to the classification block

$$y = xA^T + b \quad (3)$$

$$y = x_1^T A x_2 + b \quad (4)$$

The standard classification block consists of a linear layer mapping pooled output vector (x of the size of the hidden layer) from the transformer block as shown in Eq. 3 to class label (y). We replaced the linear layer with a bilinear layer where the pooled output (x_1 of the size of the hidden layer) and gene encoding (x_2 of size 2) are mapped to a class label (y). Bilinear transformation, as shown in Eq. 4, will calculate weight matrix (A) and bias (b) based on the two input vectors (of sequence and genes), which will express the interaction between sequence and genes.

Gene usage in embedding block.

The embedding block of a pre-trained model contains a learnt representation of its vocabulary. The introduction of additional information would then require learning new representations. There are two possible ways to adjust the embedding block towards additional features. The first is to add unique V and J genes as new special tokens to the tokeniser, and the second is to add embedding layers for all unique V and J genes.

Our initial approach was to add genes as tokens (63 for V-genes and 13 for J genes) to the vocabulary, which gets assimilated into the word embedding. These word embeddings are then learnt during fine-tuning. Tokeniser alters the sequence to add V-gene to the beginning and J-gene to the end of the sequence (replicating how it occurs in the TCR chain). There were slight issues with this approach. ProtTrans vocabulary comprises 30 tokens (both unique and amino acid tokens), introducing additional 76 (63+13) tokens, which occurs only twice in a sequence does not provide enough information to learn dependencies for transformers. So, this approach was discarded for a convoluted yet novel approach.

Creating embedding layers for both V and J genes will put less burden on a transformer to learn the new information (Fig 1. c). Two embedding layers, one for all unique V genes and another for all unique J genes, is created. The padding index for these layers is different from the padding for the word embedding layer, so we explicitly passed an alternative padding index. Special tokens are added before and after a protein sequence during tokenization. A separate padding index will associate a gene embedding to just a protein sequence and not to the special tokens. Special tokens are not associated with any genes, so we explicitly passed an additional vector that conveys this relationship (0 indexes of V and J genes embedding layer). Each sequence's word embedding and V and J gene (Fig 1. c) are merged. The new representations are presented to the classification block to perform the classification task.

Padding index, unique genes and special token consideration will result in a total size of V-gene embedding layer to 65 and J gene embedding layer to 15. These new layers are randomly initialised in fine-tuning, so they need to keep up with fine-tuning of the rest of the model. Hence, they are operated to learn at an increased rate than the rest of the model. For the sake of simplicity, we increased the learning rate of new embedding layers by a factor of 10. Our prime concern for not introducing an explicit learning rate for new layers was to avoid overriding weights learned in pre-training for the original layers in the embedding block. Overridden weights will make the transformer oblivious towards amino acid relationships it learnt during pre-training.

2.3 Hyperparameter optimisations and performance evaluation.

Bayesian algorithm (Tree Parzen Estimator) is utilised to optimise hyperparameters in Optuna (Akiba, Sano et al. 2019), which is preferred for cases requiring exploration for many hyperparameters. Experiments are tracked using comet.ml and PyTorch for training each model. Optimization is done for ten major hyperparameters: gradient accumulation, learning rate, weight decay, attention layer dropout (not included in autoregressive or AR model), hidden layer dropout ('dropout' in AR model), classifier layer dropout ('summary last dropout' in AR model), adam_beta1, adam_beta2 warmup ratio and gamma. Seed as a hyperparameter ensures model stability, but performance is not evaluated on seed values.

Table 2. Hyperparameters and their sampling strategy.

Hyperparameters	Sample count
Gamma	(0, 10) with step of 0.5
Adam beta 1	[0.5, 0.9] with step of 0.01
Adam beta 2	[0.5, 0.99] step of 0.001
Learning rate	[10^{-5}, 10^{-2}] from log domain
Weight decay	0, 0.01, 0.001, 0.0001, 0.00001, 0.000001
Gradient accumulation steps	[1, 128]
Classifier / Summary layer dropout	[0.5,0.9] with step of 0.1
Hidden layer / dropout	[0.5, 0.9] with step of 0.1
Attention layer dropout	[0.5, 0.9] with step of 0.1
Warmup ratio	0.10, 0.20
Seed*	[1, 100]

*For model stability

Optimisations with plenty of hyperparameters helped us gain insight into the behaviour of transformers when trained with TCR data. Optuna provides sampling strategies depending on choices of hyper-parameter. Table 2 presents the sampling strategy for each hyper-parameter and its range. Since training steps are directly dependent on training batch size and we have small training data of 1871 samples, we used training batch size and evaluation batch size of 1 and 8, respectively. Fixing training and evaluation batch size will prevent training steps from dropping too low, which will occupy hyperparameter search in unproductive search space. In preliminary experiments, we experienced high fluctuations in evaluation loss, which made transformer training susceptible to premature early stop. Therefore, each experiment is trained with early-stopping for 50 training epochs giving sufficient training time to adjust fluctuations caused by focal loss.

While the standard methodology used the exact data for testing and validation due to fewer TCR samples, many works would use in-silico CDR3 sequences to test their model's performance on unseen data. Our filtration step has removed duplicate sequences, and hence our test dataset contains unseen sequences, which conforms with standard practice to measure performance on unseen data.

2.3.1 Evaluation metrics

Metrics to assess the predictability of the transformer models includes a weighted f1-score that is used as an objective value for hyperparameter optimization. Additional metrics included are balanced accuracy score, weighted precision, and weighted recall to provide an overview of performance by a model. Parallel coordinate plots and parameter importance are plotted to evaluate the contribution of parameters towards objective value.

After optimization, an overview of performance is provided through confusion metrics and ROC plots for each implementation. ROC plots are generated after binarizing labels, followed by calculating false positive rate and true positive rate using logit scores for the binarized labels. Multiple ROC curves are plotted on the same ROC plot using this technique. Furthermore, the confusion matrix will describe classification performance on the test dataset.

2.3.2 Evaluation of different transformer implementations

In our multi-class setting, assessment of transformers on the problem of TCR is done through the epitope-specific perspective and model-specific perspective. The area under the ROC curve provides us with an epitope-specific perspective where performance comparison for TCR specificity can be made. The weighted f1-score of the model provides us with a holistic measure for comparing different models (across all methods) in a multi-class setting and hence a model-specific perspective.

For the sake of clarification, the transformer model with numbers indicates the implementation (or modification applied). For example, the standard ProtXLNet model with no modification is denoted by ProtXLNet-0, with classification block modification is denoted by ProtXLNet-1, and with embedding layer modification is denoted by ProtXLNet-2.

In order to evaluate the performance of all 12 transformers implementation, we utilised ROC plots produced from all implementations. The area under ROC (AUROC) curve for all 25 labels is then used by the paired Wilcoxon test to compare performance across different implementations of the same transformer model. We assume each model implementation with no modification as a baseline since it consists of no other modification apart from focal loss. Out of 25 labels (Fig. 2), there are 15 labels with less than 100 samples, and we term them as hard to classify labels in our comparison.

2.3.3 Evaluation of transformer with publicly available tools

We compared TCRGP, TCRdist and DeepTCR, using CDR3 sequences and gene usage as input; this is done to ensure that similar input features are employed to reach the results. As prior works focus on epitope-specific models while this work specifies a multi-class model, a common ground for comparison is an epitope-specific metric, i.e., area under ROC curve for each epitope. However, the number of samples between different tools differs since we used a newer version of VDJdb data.

TCRGP and TCRDist both have 24 common epitopes, while DeepTCR has only 20 common epitopes with our approach. In the case of DeepTCR, not all murine antigens were present in our filtered data. Out of the nine murine antigens used in their work, we had only three (m139, m38, m45) murine antigens present. We utilised their script to get AUC values for these three antigens. We then used the values provided in their work to compare with ours. Mean AUC over 24 epitopes obtained from TCRGP, TCRDist is compared with transformers and mean AUC over 20 epitopes obtained from DeepTCR is compared with the transformer models developed in this study.

3 Results

Herein, we tuned the hyperparameters of transformer models to provide us with model-specific performance. Each model differs either in pre-training methodology or the architecture of transformer blocks. Optimised transformer models predict on a test dataset (unseen by the model while training) to test the performance of the models. The following section will highlight the key takeaways after optimizations, their performance on TCR data and compare them with prior works.

3.1 Assessment of different implementations of transformers from optimisation.

Auto-encoder optimisation had eleven hyperparameters (including seed value), while the auto-regressive model had ten (including seed value); understanding their behaviour during optimisation will help us assess how transformers react to TCR data. We ran optimizations for at least 100 runs for each implementation and generated parameter importance plots and parallel coordinate plots for all 12 implementations of four transformer models (Supplementary 5, 6, 7, 8). The area under ROC for each transformer is provided in Supplementary section 4 Table 4-7.

Optimization of all three ProtBert implementations displayed higher importance towards hidden-layer dropout probability (Supplementary 4). A similar tendency was also seen in ProtElectra. Regarding ProtAlbert, the contribution was more spread out with 30% to hidden-layer dropout probability and 21% to the learning rate. Auto-encoders are encoder blocks stacked one after another, each block improving representation sent by the previous encoder block. In the case of ProtBert and ProtElectra, there is no parameter sharing among the encoder blocks, which results in large size and higher dependence on representation. This is improved in ProtAlbert by enabling parameter sharing among encoder blocks resulting in more diminutive size (number of weights/parameters). Therefore, ProtAlbert optimization displays less affinity towards hidden-layer dropout when encoding CDR3 sequences. Since ProtAlbert was not solely dependent on encoding representation but also on learning rate, this high importance to learning rate also implies learning new relationships for TCR data.

The two-stream self-attention mechanism in ProtXLNet would exhibit different behaviour than auto-encoders. While auto-encoders suffered from having a high burden on transformer block to better sequence representation, ProtXLNet did not suffer from the same. The learning rate in ProtXLNet-0 and ProtXLNet-2 displayed 30% and 36% contribution, while in ProtXLNet-1, it was reduced to 6%. This drastic reduction could be attributed to the newly introduced gene usage in the classification block, where ProtXLNet-1 is trying to generate better representation from the transformer block to make sense of the gene usage in the classification block. ProtXLNet-1 reacted similarly to auto-encoders during optimization with a 62% contribution of dropout probability, while ProtXLNet-0 and ProXLNet-2 were able to utilise optimizations well with contributions spread out across all parameters.

Table 3. Performance metrics on the test dataset.

Model	Duration (hours)	Weighted f1-score	Balanced Accuracy	Mean AUC
ProtBert-0	4.49	0.39	0.33	0.79
ProtBert-1	1.99	0.41	0.37	0.80
ProtBert-2	2.21	0.41	0.35	0.79
ProtAlbert-0	3.16	0.42	0.37	0.79
ProtAlbert-1	0.57	0.38	0.33	0.78
ProtAlbert-2	3.21	0.46	0.44	0.81
ProtElectra-0	1.29	0.46	0.41	0.86
ProtElectra-1	0.90	0.44	0.41	0.80
ProtElectra-2	0.99	0.47	0.40	0.84
ProtXLNet-0	1.57	0.48	0.44	0.81
ProtXLNet-1	1.21	0.46	0.38	0.80
ProtXLNet-2	1.23	0.55	0.50	0.88

Deduction from a parallel coordinate plot can help us gain further insight into the behaviour of models towards TCR data during optimization. Models with high importance (across all methods) towards hidden layer dropout displayed a tendency towards 0% probability. After providing gene information in embedding block, transformers required more regularisation (adam-beta1 and adam-beta2) and preferred 0% dropout probability of hidden layers. We can conclude from these findings that in transformers with embedding layer modifications, the pretrained knowledge was still playing an essential role in achieving objective value even with new and fewer TCR data. Additionally, High regularisation in embedding block modification indicates overfitting on new information and keeps most of the load on the transformer block to generate a relevant input representation (from new embedding layers) to reach objective value.

A trend in the behaviour of classifier dropout (or summary dropout in the case of ProtXLNet) would indicate the behaviour of the classification block to ease optimization. Similarly, assessment of embedding layer modification on optimization could be provided through learning rate. ProtXLNet-1 preferred higher dropout, whereas ProtXLNet-2 implementations showed varied preferences towards higher dropout. ProtBert showed similar behaviour as ProtXLNet, unlike ProtAlbert and ProtElectra. ProtAlbert-1 preferred lower dropout than ProtAlbert-2 implementations, similarly ProtElectra-1 preferred lower dropout than ProtElectra-2 implementation. All learning rates were concentrated towards lower values, and hence no differences were observed. It was concluded that there was no trend seen to support the claim that classification or embedding modification would ease optimization. Although based on performance metrics in Table 3, models with classification modification were economical as they had the least training time when compared with other implementations.

Gamma is a hyper-parameter directly linked to focal loss, which will help us deduce improvement over cross-entropy. None of the models preferred cross-entropy (gamma = 0) loss. Another trend related to focal loss is seen in parallel coordinate plots. Lower modulating factors are preferred for methods 0 and 2, but intermediate values (around 5) are preferred for method 1. Less diversity in gamma values for the bilinear layer implies fine-tuning was concentrated towards the classification block for method-1. We can conclude that focal loss contributed to optimisation and classification performance.

3.2 Assessment of different implementations of transformers on TCR data.

Assessment of different implementations is done among transformers with confusion matrix (Supplementary 1), ROC plots (Supplementary 2) and paired Wilcoxon test (Supplementary 3) on AUROCs from individual transformers. Misclassification is compared based on confusion matrix, and epitope specificity is evaluated using ROC plots, and AUROC for each epitope is provided in Supplementary 4. Implementational significance for each Transformer is compared using the paired wilcoxon test results.

Misclassification in ProtBert-0 was scattered mostly for the CMV epitope family, with even many unclassified labels such as CMV IE2 or HCV NS3. ProtBert-1 improved the results of ProtBert-0 by reducing the number of unclassified labels from seven in ProtBert-0 to one in ProtBert-1, while ProtBert-2 managed to keep unclassified labels to five. Comparison between the different implementations of ProtBert by Wilcoxon (Supplementary Fig 26) test showed all implementations in ProtBert worked similarly.

Comparing ProtBert-0 with ProtAlbert-0 showed clear improvement over misclassification and the number of unclassified labels. A more significant number of hard to classify examples are correctly classified in ProtAlbert-0 compared to ProtBert-0. ProtAlbert-0 had three unclassified labels, which is more than ProtBert-1 but comparing ProtBert-0 and ProtAlbert-0, ProtAlbert-0 showed improvements in classifying hard-to-classify labels. The most striking improvement over ProtBert-1 is how ProtAlbert-2 improved on all its misclassification. Misclassification with HIV-1 Gag, which was highly confused with CMV and EBV epitopes in ProtBert-2, is reduced in ProtAlbert-2. ProtAlbert seemed to perform

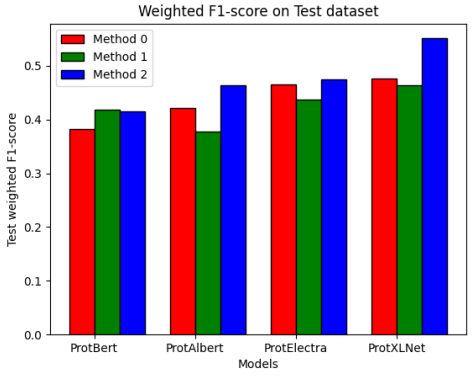

Fig. 4 F1-score for all models. Weighted f1-score on test dataset for all methods and across all models. Performance of ProtBert-1, ProtAlbert-2, ProtElectra-2 and ProtXLNet-2 on test dataset are best among same transformer model.

well with embedding layer modification based on the Wilcoxon test among ProtAlbert implementation (Supplementary Fig 27). In addition to that, in ProtAlbert-2, hard-to-classify classes have drastically improved, for example, the MCMV epitope family. Moreover, unclassified labels came down from five in ProtAlbert-0 to one in ProtAlbert-2.

ProtElectra had a very different response to modification than others models. The unclassified labels increased from two in ProtElectra-0 and ProtElectra-1 to four in ProtElectra-2. Many samples were misclassified in ProtElectra-1 when comparing confusion matrices of all ProtElectra models, implying it performed poorly among all the implementations. Wilcoxon test showed a higher mean AUC for ProtElectra-0 than all other implementations (Supplementary Fig 28). Even though this could be attributed to lower AUC for HIV-1 Gag epitope (0.77 in ProtElectra-0 and 0.53 in ProtElectra-2) in ProtElectra-2, which caused the mean AUC to drop from 0.856 (ProtElectra-0) to 0.836 (ProtElectra-2) but weighted f1-score for ProtElectra-2 was still higher among all the ProtElectra implementations. Among all the auto-encoders, ProtElectra2 showed the best f1-score (Fig 4).

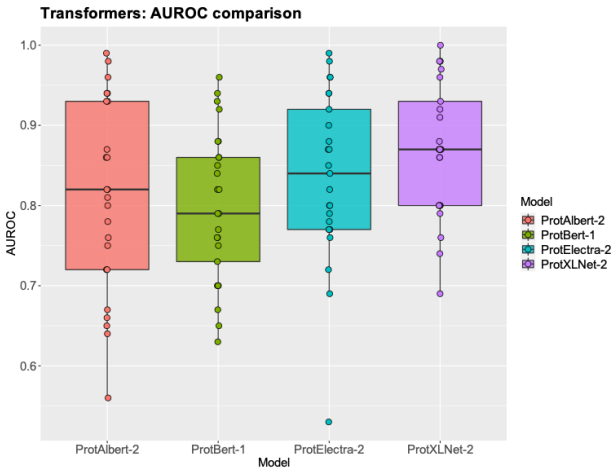

Fig. 5 AUROC comparison for Transformers. Performance of ProtBert-1, ProtAlbert-2, ProtElectra-2 and ProtXLNet-2 on TCR test dataset provide us an overview of how transformers perform on unseen TCR data.

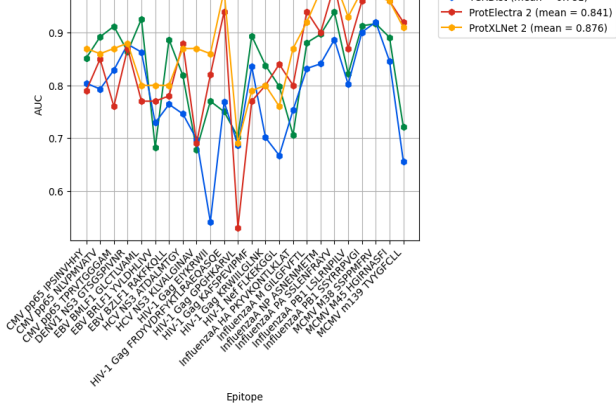

Fig. 6 Comparison with TCRGP and TCRdist. Area under ROC for twenty-four epitopes common with TCRGP and TCRdist are presented. Mean of each model over given epitopes are also mentioned.

We only used one auto-regressive model for TCR data, but it showed the most receptiveness towards all modifications. Wilcoxon test showed ProtXLNet-2 performing better among all implementations of ProtXLNet where an improvement in mean AUC from 0.812 in ProtXLNet-0 from 0.870 in ProtXLNet-2 was observed. Observing ROCs of ProtXLNet-0 showed improved results over ProtElectra-2. For the HCV family, ProtElectra-2 produced higher AUC for ease to classify (HCV NS3 ATDALMTGY: 0.88) and lower AUC for hard to classify (HCV NS3 KLVALGINAV: 0.69). ProtXLNet-2, on the other hand, was able to learn patterns within the HCV family to produce higher AUC for both classes (0.87). A similar improvement for the CMV epitope family was observed when comparing the AUC of hard-to-classify labels among ProtXLNet-2 and ProtElectra-2. In all, ProtXLNet-2 performed well by reaching the highest F1-score of 0.55 (Fig.4) among all models, with better classification performance for hard to classify labels.

3.3 Transformers in comparison with publicly available tools.

VDJdb comprises the TCR-epitope pairs from multiple independent studies, and the database keeps on adding the sequenced or validated TCR-epitope pairs. It allowed previously built tools to use similar datasets (with substantial overlap in epitopes) and compare their outcomes to benchmark the different available tools. In our approach, we reported the ProtElectra-2 and ProtXLNet-2 to be best among various implementations based on weighted f1-score. To estimate the performance of the best models from our approach, we compared our best-performing methods with TCRGP, TCRdist and DeepTCR. We first compare our best performing models with TCRGP and TCRdist, where we have the most (24 out of 25) common epitope classes. An additional comparison is made with another Deep Learning implementation, DeepTCR, with which we have 20 epitopes in common.

The mean AUC for TCRGP and TCRdist is 0.831 and 0.781 calculated over 24 epitopes. ProtElectra-2 showed improvements of just 1% (0.841) over TCRGP but ProtXLNet-2 was able to show an improvement of almost 5% (0.876) (Fig 6). TCRGP had utilised epitope-specific models in a binary setting. TCRGP and TCRdist had a mean AUC of 0.843 and 0.807, respectively, for the MCMV family with three hard-to-classify classes. ProtXLNet-2 and ProtElectra-2 had mean AUC of 0.946 and 0.953, respectively for the MCMV family. An improvement of ≥10% by both auto-encoder and auto-regressive models proves the superiority of transformer models over TCRGP and TCRdist.

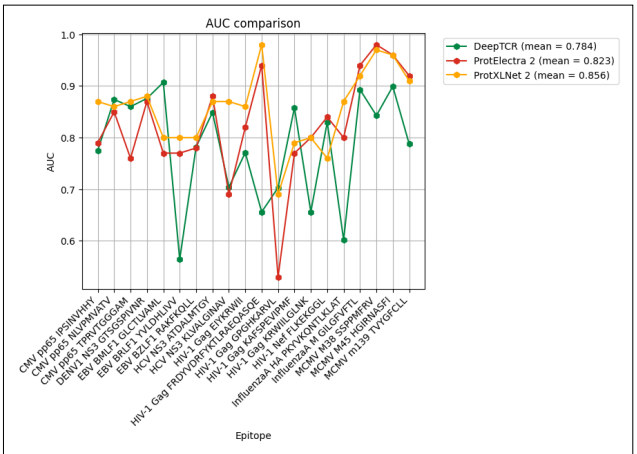

Fig. 7 Comparison with DeepTCR. Area under ROC for twenty epitopes common with DeepTCR are presented. Mean of each model over given epitopes are also mentioned.

DeepTCR performance is compared on common epitope classes with our work. AUC for twenty epitopes is compared (Fig. 7) between DeepTCR and Transformers. ProtXLNet-2, with a mean AUC of 0.856, showed improvement over DeepTCR with a mean AUC of 0.784. Most lack of performance in DeepTCR was due to hard to classify classes, while transformers show improved performance with such labels. For example, the MCMV family, DeepTCR produced results with a mean AUC of 0.843 and while ProtElectra-2 and Prot-XLNet-2 produced results with 0.953 and 0.946 mean AUC.

4 Discussion and Limitations

In this work, we addressed the biological problem of TCR specificity through Transformers. Due to their self-attention mechanism, transformers in NLP have outperformed on all translation and sequence inference tasks. We utilised the self-attention mechanism of multiple transformer architectures to learn the antigen specificity of TCRs. Each architecture of transformer has improved shortcoming of previous architecture, and we demonstrated which transformer performs best on TCR data. As pre-trained transformers are attuned to a specific vocabulary, providing additional features to the transformers was never explored. The transformer's capabilities can be extended by adding additional features along with sequences to improve sequence inference. We demonstrated potential modification in pre-trained Transformers for improved classification and compared our results with other works. We introduced measures in the form of multi-class focal loss, gene usage in classification block and gene usage in embedding block to aid our specificity classification. We presented the possibility that methodologies on biological problems can be streamlined to utilise a transformer for sequence inference accompanied by non-sequence-based features.

Previous works tackled TCR specificity either through sequence similarity (Pogorelyy, Minervina et al. 2019), k-mer sequence features (Tong, Wang et al. 2020), convolutional models (Jurtz, Jessen et al. 2018) , utilising amino acid physicochemical properties (Gielis, Moris et al. 2018), learning a non-parametric function (Jokinen, Huuhtanen et al. 2021) or enhancing sequence feature into a high-dimensional space (Sidhom, Larman et al. 2021), even utilising molecular structures (Weber, Born et al. 2021) or just recently published, pre-training BERT model on TCR data (Wu, Yost et al. 2021). A common theme in all these works is to improve encoding of input data to exemplify TCR specificity for an antigen. Usually, a binary classification model is trained to exhibit antigen specificity among a background of antigens. However, DeepTCR, along with binary setting, included a multi-class setting; they did not employ it for the entire TCR data, just for the GAG TW10 epitope family (with ten variants). To the best of our knowledge, the multi-class approach has never been utilised in such a fashion because of the scarcity of TCR data or lacking computational resources. This work provided an alternative approach towards TCR classification for similar data sizes and reduced training time by utilising pre-trained transformers.

We only used CDR3 sequences and gene usage of the beta chain of TCR, which is a simplified approach. A more widely utilised approach is utilising both alpha and beta chain sequences (along with gene usage for both chains). TCR-BERT used two different BERT models (one for beta sequences other for alpha sequences) and combined their embeddings for the classifier block. A similar approach can modify all ProtTrans models to accommodate alpha sequences, providing more performance in determining TCR specificity.

A tendency we observed during optimisation was the early stopping of potentially good runs. As early stopping was performed based on evaluation loss values, the potential origin for this issue was loss function. Depending on modulating factor, the focal loss causes many spikes in loss values; in turn, these fluctuations are misinterpreted by early stopping as loss value worsens. We were increasing the tolerance threshold for early stopping but was led to more stale runs and immense waste of training time. Exploring how to best tackle this problem could increase the productivity of optimisations.

Comparing transformers with other methods proved that they outperformed in determining TCR specificity. A recent work utilising transformers using a similar configuration of data is TCR-BERT. Although we outperformed all previous methods in epitope-specific AUCs, it would be insightful to compare our methodology with TCR-BERT. Despite not making a head-to-head comparison with TCR-BERT, their approach resulted in an AUC of 0.837 for CMV-pp65-NLVPMVATV (using both TRA and TRB sequences), whereas ProtXLNet-2 classified CMV-pp65-NLVPMVATV (using only TRB sequences) with 0.86. Their work utilises the entire 81K entries from VDJdb to pre-train a BERT model as opposed to ours, where we utilised data with a confidence score of more than 0. TCR-BERT fine-tuned TCR specificity for binding to a positive (antigen) class. Our preliminary approach included all confidence scores, where we fine-tune our transformers with 78647 samples on 48 labels. The drawback of this approach was that we are fine-tuning with entries that are not adequately validated, so we would not get comparable results. Nevertheless, comparing two approaches could provide greater insight into which approach (fine-tuning or pre-training) is better suited for TCR data.

Although our approach demonstrates the potential of transformers by fine-tuning on CDR3 sequences and how additional features can be juxtaposed cohesively, some limitations remain. Problematic events such as hallucination and catastrophic forgetting (Sun, Qiu et al. 2019) were not evaluated. These problems are prevalent in deep neural networks and contribute significantly to misclassification. While hallucination occurs in generated sequence at the last transformer block, leading to unlikely content. Catastrophic forgetting is prevalent in transformers where the weights learned during pre-training gets overridden during fine-tuning.

Predominantly, hallucination (Kolouri, Ketz et al. 2019) (Ji, Lee et al. 2022) often occurs when generating natural languages. Hallucinating amino acid is less likely due to our models' small vocab size (amino acids). Vulnerability to hallucination was a prime reason for not adding gene usage as additional tokens to the vocabulary but instead adding them through embedding layers. If we had added 76 new tokens upon 30 tokens in the vocab, which occurs only twice with the sequence (V and J gene associated with the sequence), either there would be hardly anything to learn, or there would be hallucinations of genes in place of amino acids or vice-versa.

For catastrophic forgetting, we had employed early stopping. We lowered the learning rate (which, as discussed in results, was preferred by all Transformers) to prevent pre-trained knowledge from being erased during fine-tuning. Although we used a factor of 10 to train new embedding layers, comparison with different factors could help us evaluate whether different transformers perform better at a different rate. Appropriate assessment of the limitations, as mentioned earlier, can help us improve on the results of transformers even with such highly imbalanced data.

Modification in embedding block gave us a new perspective towards the attention masking procedure. As protein binding leads to a 3D conformation in protein structures, information about bindings in these structures is learned by the attention mechanism for a given sequence. An as epitope binding involves very few amino acids of a sequence, explicitly indicating non-relevant amino acids (known as junctional

regions) puts less load on the attention mechanism. Information about which regions in CDR3 are junctional regions is provided in the VDJdb meta description. Piecing this knowledge, we can instruct the tokeniser to calculate attention for specific regions in CDR3 sequences. The results can further be evaluated using attention head visualisation (Vig 2019), where the attention heads playing the most vital role towards correct classification gets highlighted.

We have demonstrated the feasibility of transformers in determining antigen specificity of a TCR from CDR3 protein sequences. Diversity conveyed in CDR3 sequences has long been a challenge in T-cell research; we aimed to diminish the challenge with this work. We conceptualised modifications in transformers to convey additional features that improved classification performance. Results of these modifications were compared and shown to outperform prior tools that addressed a similar problem. Additional opportunities in examining implications of the approach adopted in this work can be helpful for many other fields utilising transformers for either its self-attention mechanism or transfer learning.

Acknowledgements

I would like to thank Dr. Khatri for sharing her experience and expertise to tackle challenges in this arduous journey. I also want to thank Prof. Reinders for providing direction and focus to this work. This work would not have been possible without their feedback, guidance and unwavering support. I would also like to express my gratitude towards T-cell and NLP research community for their extensive documentation of their work, which kept me inspired and motivated for the entirety of this work. Lastly, to the people dearest to me, thank you for your support.

Appendix

1 Biology behind antigen recognition by T-Cell

Understanding T cell receptors (TCRs) relates directly to the understanding of mechanisms involved in the adaptive immune system. While adaptive immunity involves both B and T cells, we will focus on T cells which will provide us insight into cell-mediated adaptive immunity.

Antigen Presentation and MHC Restriction

Presenting peptides to the TCRs is done through a class of cells known as antigen-presenting cells (APC) and is presented through Major Histocompatibility complexes (MHC); this process is termed antigen presentation. Degradation of antigen is done inside APC; this degradation is undertaken through two different pathways depending on if MHCI or MHCII is expressed on the APC. The affinity of degraded peptide (or epitope) to the MHC dictates which peptide will be presented by the MHC and also to which T-Cell (CD4+ or CD8+) (Zareie, Farenc et al. 2020).

Antigen Recognition

TCR expressed on either CD8+ or CD4+ T cells binds to MHC for antigen recognition by their respective T cells. T cells' clonal nature dictates the unique binding site on its TCRs and hence its specificity. T-Cell consists of Variable (V) and Constant (C) region; it is on V region where the sequence diversity is the most concentrated and is present on both Alpha and Beta chain of TCRs. TCR genes undergo V, D and J gene rearrangement providing TCR with high diversity.

Gene rearrangement

Although Gene rearrangement provides TCR diversity expected at 10^{18} in humans and 10^{15} in mice, it is not random at all, and thus regulation is governed by multiple factors (Attaf, Huseby et al. 2015). TCR beta locus comprises 46 V gene segments, followed by two groups of D, J and C gene segments (D1, J1, C1 or D2, J2, C2). D to J recombination first occurs between a D gene and one of J gene of first group or D and J gene of the second group; followed by V to the newly rearranged D and J gene. V(D)J recombination of TCR genes plays a vital role in governing the diversity of a repertoire (all unique TCR within an individual's immune system). Additionally, random insertion and deletion of nucleotides at junctions of V, D and J give rise to hypervariable regions, known as complementarity determining regions.

Complementarity-determining region on TCR

Complementarity-determining regions are present at the junction of V and D segment of TCRs Fig.8. This region of TCR interacts the most with both MHC and epitope for recognition, which is due to its hypervariability. Two types of TCR, i.e., TCRAB and rarely TCRGD are mounted by the adaptive immune system for an immune response. These receptors (at protein level) are generated from two different chains, i.e., TCRAB from TCRA and TCRB and TCRGD from TCRG and TCRD. Each of these four receptors is generated from the recombination of Variable (V), diversity (D) and Joining (J) genes from their individual loci. Several V, D and J genes for TCRB and TCRD and V and J genes for TCRA and TCRG are distributed over a long stretch of chromosomes 7 and 14 in the human genome. Any of these V, D and J genes can be selected for each locus and any of the chains, i.e., TCRA/TCRB can be randomly selected to generate diverse receptors to raise an efficient immune response against infections. Overall, 10^{16} receptors can alone be generated by the V(D)J recombination events. Apart from V(D)J recombination, another factor that adds to the diversity of these receptors is the addition of nucleotides on both sides of D gene during V(D)J recombination. This region is known as complementarity determining region 3 (CDR3), which is fundamental in interacting with and recognising the antigen. Determining binding specificity to an antigen thus helps us to assess the immune system's ability to engage with pathogens and also to evaluate the response undertaken by it.

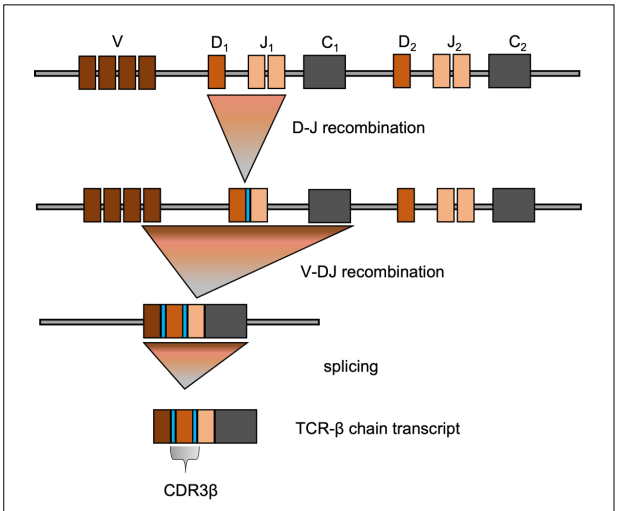

Fig. 8. TCRβ gene rearrangement and structure. Overview of V(D)J recombination. Region in blue indicates junctional sites which are assembled by random additional and deletion. CDR3β region shown in grey on TCR protein.

2 Transformers

Transformers are a type of Sequence to Sequence (or Seq2Seq) model that transforms a source sequence's representation to a representation of a target sequence.

Machine translation (also known as sequence transduction) which were widely performed using LSTM (long-short-term-memory) models or RNN (recurrent neural networks) models, had limitations. Limitations of fixed sequence length in traditional methods (and even the bottleneck between encoder and decoder architectures) were then overcome with the introduction of Transformers. Transformers retained the encoder-decoder architecture and introduced a new form of attention mechanism, self-attention, which outperformed standard practices.

Encoder-Decoder Architecture

Given a source sequence to be translated to a target sequence, the encoder would generate vector representation for each word in the source sequence, and the decoder would then read these vectors and generate words in the target sequence. Certain limitations were discovered; translating word to word doesn't account for languages written from left to right or with different grammar compositions. Subsequently, to reduce computational resources instead of encoding the entire source sequence, a need to focus on relevant words arise.

Recurrent Neural Network (Bahdanau, Cho et al. 2014) would solve the former issue by including a weighted sum of preceding and succeeding words in a sequence whilst encoding a word. It employed RNN to encode each word, and a decoder would then decode from this representation. With the introduction of LSTMs (Sutskever, Vinyals et al. 2014) long-range dependencies for a word were also accounted for.

Ditching recurrent and convolutional layers was Transformers, replacing it with an attention mechanism. Transformers introduced encoding through self-attention, which would amplify the contribution of relevant words and diminish the contributions of irrelevant words while encoding a vector representation for a word. All this whilst retaining encoder-decoder architecture (there are encoder-only architectures like BERT).

Attention, Self-attention and multi-head attention

Origins of attention lie in the field of psychology, where the observations in behavioural patterns were attributed to where the brain was paying attention. The brain preserves computation resources by paying attention to only crucial details to reach an answer (Lindsay 2020), which can be mimicked by utilising a weighted sum of relevant words while encoding a word. Mathematically formulating this "flexibility" in a neural network is known as the attention mechanism. Global and Local attention are some of the early attention mechanisms which were applied in computer vision (Luong, Pham et al. 2015, Xu, Ba et al. 2015). With the introduction of transformers, self-attention came into the limelight.

While traditional attention mechanisms would include the contribution of surrounding words blindingly, self-attention would also include the position of the word to introduce the sense of context in encoding a word. In summary, similar words in different positions in a sequence would get different encoding.

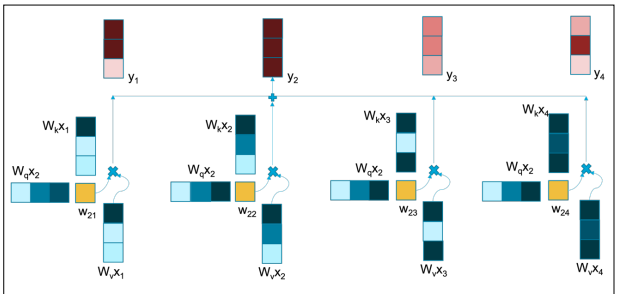

Fig. 9. Self-Attention in Transformers. Overview of self-attention computation for encoding y_2. Two representations (Query and Key) of a word are used to compute scores (shown in yellow) which enhances or reduces effect of a word (multiplying by Value). Resultant effect is summed across all words to encode a single word (y_2)

Fig. 9 shows encoding the second word in the output sequence. Three different representation of input is used to calculate self-attention. Learning the weights for every three representations is equivalent to learning self-attention. Query and Key are used to compute scores, which undergoes some processing before taking a dot product with the Value representation. This dot product will enhance values in the Value matrix, which corresponds to higher relevance and hence augment the effect of those words into the resulting representation of the word. This can be done multiple times in parallel for a single word, termed as multi-headed attention. Multi-headed attentions allow transformers to accommodate relevance from multiple positions in a sequence.

Transfer Learning and Transformers architectures

Training a transformer involves the concept of transfer learning; in transfer learning, we divide training a model into two different parts: pre-training and fine-tuning. Pre-training involves two tasks mask language modelling and Next Sentence prediction. In the case of ProtTrans, only mask language modelling was utilised. The weights learned in pre-training are fine-tuned on downstream tasks, consequently saving time and resources in training a transformer from the beginning and prime advantage of transfer learning. Additionally, different Transfer models employ different approaches for pre-training tasks, which is motivated by their methodology.

There are two architectures utilised in this work, Auto-encoder and Auto-regressive. BERT, Albert, Electra are auto-encoder models with only encoders and no decoders. XLNet is an autoregressive language model. While BERT learns bidirectional language modelling, ALBERT (A lite BERT) is a more efficient version of BERT with parameter sharing among different encoder layers. Both of them use the same mask language modelling. On the other hand, ELECTRA utilises Generator-Discriminator based approach; the discriminator then detects corrupted tokens generated by the generator during masked language modelling. The discriminator trained is then used for fine-tuning on downstream tasks.

While auto-encoder models reconstruct original data during mask language modelling. They lack obvious information; masked token (it is a type of special token [MASK]) will not be seen in downstream tasks, dependency learnt for the masked token (and the original token) is then not transferred (or would never be needed); termed as the pretrain-finetune discrepancy. The pretrain-finetune discrepancy is addressed in XLNet. XLNet is an auto-regressive model which implements two-stream self-attention as a means to address both forward and backward dependencies as well as to address pretrain-finetune discrepancy. Providing different factorisation orders during permutation language modelling enables the model to gather positional information from a possible position for a given token.

On the other hand, two-Stream self-attention uses additional self-attention to isolate the content while the model learns positional information (or context) during pre-training. Any difference between XLNet and BERT is due to this very difference in pre-training objective, which helps it retain a more significant number of dependencies than BERT. In summary, XLNet is a bidirectional transformer similar to BERT but utilises permutation language modelling.

References

Akiba, T., et al. (2019). Optuna: A next-generation hyperparameter optimization framework. Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining.

Attaf, M., et al. (2015). "αβ T cell receptors as predictors of health and disease." Cellular & molecular immunology **12**(4): 391-399.

Bahdanau, D., et al. (2014). "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473.

Cho, K., et al. (2014). "On the properties of neural machine translation: Encoder-decoder approaches." arXiv preprint arXiv:1409.1259.

Dash, P., et al. (2017). "Quantifiable predictive features define epitope-specific T cell receptor repertoires." Nature **547**(7661): 89-93.

Devlin, J., et al. (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

Elnaggar, A., et al. (2020). "ProtTrans: towards cracking the language of Life's code through self-supervised deep learning and high performance computing." arXiv preprint arXiv:2007.06225.

Gielis, S., et al. (2018). "TCRex: a webtool for the prediction of T-cell receptor sequence epitope specificity." BioRxiv: 373472.

Hodges, E., et al. (2003). "Diagnostic role of tests for T cell receptor (TCR) genes." Journal of clinical pathology **56**(1): 1-11.

Hutchins, J. (2007). "Machine translation: A concise history." Computer aided translation: Theory and practice **13**(29-70): 11.

Ji, Z., et al. (2022). "Survey of Hallucination in Natural Language Generation." arXiv preprint arXiv:2202.03629.

Jokinen, E., et al. (2021). "Predicting recognition between T cell receptors and epitopes with TCRGP." PLoS computational biology **17**(3): e1008814.

Jurtz, V. I., et al. (2018). "NetTCR: sequence-based prediction of TCR binding to peptide-MHC complexes using convolutional neural networks." BioRxiv: 433706.

Kalchbrenner, N. and P. Blunsom (2013). Recurrent continuous translation models. Proceedings of the 2013 conference on empirical methods in natural language processing.

Koehn, P. (2017). "Neural machine translation." arXiv preprint arXiv:1709.07809.

Kolouri, S., et al. (2019). "Attention-based structural-plasticity." arXiv preprint arXiv:1903.06070.

Lin, T.-Y., et al. (2017). Focal loss for dense object detection. Proceedings of the IEEE international conference on computer vision.

Lindsay, G. W. (2020). "Attention in psychology, neuroscience, and machine learning." Frontiers in computational neuroscience: 29.

Luong, M.-T., et al. (2015). "Effective approaches to attention-based neural machine translation." arXiv preprint arXiv:1508.04025.

Petrova, G., et al. (2012). "Cross-reactivity of T cells and its role in the immune system." Critical Reviews™ in Immunology **32**(4).

Pogorelyy, M. V., et al. (2019). "Detecting T cell receptors involved in immune responses from single repertoire snapshots." PLoS Biology **17**(6): e3000314.

Robins, H. S., et al. (2009). "Comprehensive assessment of T-cell receptor β-chain diversity in αβ T cells." Blood, The Journal of the American Society of Hematology **114**(19): 4099-4107.

Schwenk, H. (2007). "Continuous space language models." Computer Speech & Language **21**(3): 492-518.

Shugay, M., et al. (2018). "VDJdb: a curated database of T-cell receptor sequences with known antigen specificity." Nucleic acids research **46**(D1): D419-D427.

Sidhom, J.-W., et al. (2021). "DeepTCR is a deep learning framework for revealing sequence concepts within T-cell repertoires." Nature communications **12**(1): 1-12.

Steinegger, M., et al. (2019). "Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold." Nature methods **16**(7): 603-606.

Steinegger, M. and J. Söding (2018). "Clustering huge protein sequence sets in linear time." Nature communications **9**(1): 1-8.

Sun, C., et al. (2019). How to fine-tune bert for text classification? China national conference on Chinese computational linguistics, Springer.

Sutskever, I., et al. (2014). Sequence to sequence learning with neural networks. Advances in neural information processing systems.

Sutskever, I., et al. (2014). "Sequence to sequence learning with neural networks." Advances in neural information processing systems **27**.

Suzek, B. E., et al. (2015). "UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches." Bioinformatics **31**(6): 926-932.

Tong, Y., et al. (2020). "Sete: Sequence-based ensemble learning approach for tcr epitope binding prediction." Computational Biology and Chemistry **87**: 107281.

Vaswani, A., et al. (2017). Attention is all you need. Advances in neural information processing systems.

Vig, J. (2019). BertViz: A tool for visualizing multihead self-attention in the BERT model. ICLR Workshop: Debugging Machine Learning Models.

Weber, A., et al. (2021). "TITAN: T-cell receptor specificity prediction with bimodal attention networks." Bioinformatics **37**(Supplement_1): i237-i244.

Wolf, T., et al. (2019). "Huggingface's transformers: State-of-the-art natural language processing." arXiv preprint arXiv:1910.03771.

Wu, K., et al. (2021). "TCR-BERT: learning the grammar of T-cell receptors for flexible antigen-xbinding analyses." BioRxiv.

Xu, K., et al. (2015). Show, attend and tell: Neural image caption generation with visual attention. International conference on machine learning, PMLR.

Zareie, P., et al. (2020). "MHC restriction: Where are we now?" Viral Immunology **33**(3): 179-187.