TCR-BERT: learning the grammar of T-cell receptors for flexible antigen-binding analyses

Kevin Wu[1,2,6], Kathryn E. Yost[2], Bence Daniel[2,3], Julia A. Belk[1,3], Yu Xia[4], Takeshi Egawa[4], Ansuman Satpathy[3], Howard Y. Chang[2,5,*], James Zou[1,6,*]
[1] Department of Computer Science, Stanford University, Stanford, CA 94305.
[2] Center for Personal Dynamic Regulomes, Stanford University, Stanford, CA 94305.
[3] Department of Pathology, Stanford University School of Medicine, Stanford, CA 94305.
[4] Department of Pathology and Immunology, Washington University School of Medicine, St. Louis, MO 63110.
[5] Howard Hughes Medical Institute, Stanford University, Stanford, CA 94305.
[6] Department of Biomedical Data Science, Stanford University School of Medicine, Stanford, CA 94305.
[*] Co-corresponding authors

Abstract

The T-cell receptor (TCR) allows T-cells to recognize and respond to antigens presented by infected and diseased cells. However, due to TCRs' staggering diversity and the complex binding dynamics underlying TCR antigen recognition, it is challenging to predict which antigens a given TCR may bind to. Here, we present TCR-BERT, a deep learning model that applies self-supervised transfer learning to this problem. TCR-BERT leverages unlabeled TCR sequences to learn a general, versatile representation of TCR sequences, enabling numerous downstream applications. TCR-BERT can be used to build state-of-the-art TCR-antigen binding predictors with improved generalizability compared to prior methods. Simultaneously, TCR-BERT's embeddings yield clusters of TCRs likely to share antigen specificities. It also enables computational approaches to challenging, unsolved problems such as designing novel TCR sequences with engineered binding affinities. Importantly, TCR-BERT enables all these advances by focusing on residues with known biological significance.

Introduction

T cells are a central component of the adaptive immune system[1]. Mature T cells continuously monitor their surroundings for signs of foreign invaders or diseased cells and help activate immune defenses upon recognition. These invaders are signified by antigens – short protein fragments presented outside each cell by the major histocompatibility complex (MHC) – that give a "snapshot" of the inner workings of a cell. In the event of a viral infection such as with HIV or SARS-CoV-2, infected cells present viral antigens, which are then recognized by T cells[2]. In cancer, fragments of mutated intracellular proteins are similarly presented by cancerous cells and detected as neoantigens. Healthy cells present antigens identifying themselves as non-threatening, but aberrant recognition of these self-antigens as invaders – a phenomenon known as autoreactivity – can cause autoimmune disorders like multiple sclerosis[3,4] and type 1 diabetes[5,6]. T-cell mediated anti-tumor immunity can be therapeutically harnessed in several ways[7], including immune checkpoint blockade therapies targeting cancers[8]. In each of these settings, antigen(s) recognized by T cells are key to developing effective treatments[9–14].

Despite the importance and therapeutic potential of T cells, it is challenging to predict their antigen recognition behavior, which is mediated by the T-cell receptor (TCR). The TCR is a dimeric protein with two hypervariable chains – typically α and β chains, encoded by the *TRA* and *TRB* genes – that jointly bind and recognize the pMHC-antigen complex[15]. These TRA and TRB sequences are specified through recombination of the variable (V), diversity (D), and junction (J) gene segments, as well as through random insertions and deletions. This stochastic process generates (hundreds of) millions for a healthy human[16–18]. This diversity crucially lends the immune system its ability to recognize a vast array of antigens, but also makes understanding and predicting TCR-antigen specificity difficult. This challenge is compounded by cross-reactivity, where a single TCR often recognizes multiple antigens[19–21], and conversely, an antigen may be recognized by many TCRs with varying affinities[22].

Recently, the growing popularity of sequencing technologies has enabled high-throughput profiling of TCR sequences[23,24], which has in turn empowered computational methods studying TCR-antigen binding.

Conventional methods like GLIPH[25,26], TCRMatch[27], and TCRdist[28,29] rely on sequence motif comparisons and manually tuned heuristics to predict which TCRs likely share antigen binding partners. More recently, researchers have applied various machine learning methods to predicting TCR-antigen binding[30–32]. While specific methodologies differ, these share the same supervised learning strategy training a classifier on TCR sequences with known antigen binding specificities. However, many TCRs do not have such antigen labels, which greatly limits the amount of data that supervised learning can leverage. Furthermore, it is likely that existing labels are incomplete owing to cross-reactivity. Thus, there is a unique opportunity to develop new approaches that leverage unlabeled data in concert with labelled examples to improve these models' robustness and generalizability.

This challenge of using unlabeled data to build models is a burgeoning field of study within the broader machine learning community. A notable advance in this domain is the Bidirectional Encoder Representations from Transformers (BERT) architecture, originally developed for natural language processing tasks[33]. BERT is a highly expressive model trained to understand the grammatical structure of languages using large collections of unlabeled sequences – a process more formally known as pre-training. This general understanding then serves as a robust starting point for targeting more specific tasks. While BERT was originally designed to model human language, BERT and its related architectures have been successfully repurposed for modelling biological sequences like DNA[34] and proteins[35–37].

Inspired by the challenges of TCR analysis and the success of these machine learning approaches, we present TCR-BERT, a modified BERT model trained specifically on TCR CDR3 amino acid sequences. TCR-BERT explicitly leverages unlabeled TCR sequences to achieve state-of-the-art performance on a variety of downstream tasks in TCR analysis, including antigen specificity prediction and exploratory clustering analyses, and even enabling *in silico* design of TCR sequences with specific binding characteristics.

Results

TCR-BERT leverages large, unannotated datasets to learn representations of TCRs

TCR-BERT is built upon a modified BERT architecture (see Methods for details) that takes TCR CDR3 amino acid sequences as input (e.g., CASRPDGRETQYF). We pre-train TCR-BERT to capture the language of TCR CDR3 sequences by optimizing two objectives sequentially. First, we use unlabeled TCR sequences to learn the grammar of the naturally occurring TCR sequence space. We randomly hide 15% of the residues in each sequence and train TCR-BERT to impute these based on surrounding residues. This masked amino acid (MAA) pre-training does not require knowledge of the antigen specificity of each TCR sequence. MAA pre-training is performed using 88,403 predominantly human TRA and TRB sequences drawn from the VDJdb[38] and PIRD[39] datasets (Figure 1).

After MAA pre-training, we leverage the fact that some TCRs are "labelled" with known antigen binding to train TCR-BERT to predict antigen specificity given TCR sequence in the form of a multi-class prediction problem. This classification pre-training task takes the TCR grammar previously learned via MAA and tunes it towards semantics of antigen specificity and is unique to TCR-BERT – existing protein language models are pre-trained only on the MAA task. We consider 4,365 human TRB sequences, each binding to one of 45 antigen labels derived from the PIRD dataset (Figure 1, see Supplementary Methods). Although this antigen classification pre-training step uses relatively few examples, it provides substantial benefits to TCR-BERT (see Supplementary Methods). As a sanity check, we apply TCR-BERT to a set of murine TCR sequences and find evidence of generalization (see Supplementary Methods).

TCR-BERT enables state-of-the-art antigen specificity classifiers

After pre-training, we use TCR-BERT as a basis for building classifiers to predict whether a given TCR sequence can bind to a specific antigen. We use TCR-BERT to generate embedding vectors describing TCR

sequences, which are then used as input features to train a support vector machine (SVM) predicting binding towards a specific antigen (see Supplementary Methods). To evaluate the efficacy of this approach, we perform antigen cross validation. For a subset (n=26) of the antigens used in antigen classification pre-training with at least 20 known binding TCRs, we repeat the antigen classification pre-training step excluding that antigen and associated TRBs. We use the resulting model to embed and classify the held-out TRBs against a negative set of human TRBs of undetermined affinity. This negative set is sampled from the TCRdb database[40], which is not seen in pre-training, at a ratio of 5 negatives for each binding sequence. Under random 70/30 train/test splits, we observe a test set area under the precision-recall curve (AUPRC) of 0.91 averaged across 26 antigens. TCR-BERT outperforms a baseline supervised (without pre-training) convolutional neural network (ConvNet) predicting antigen binding/non-binding given a TRB sequence (Figure 2A, see Supplementary Methods), as well as various models built on *k*-mer featurization schemes (Supplementary Figures 2A, 2B).

We compared TCR-BERT to other pre-trained protein language models by applying the same SVM classification approach to TCR embeddings generated by general-purpose protein language models ESM[35] and TAPE[36]. TCR-BERT's embedding yields improved AUPRC compared to both ESM (Figure 2B) and TAPE (Supplementary Figure 2C). These results are consistent across different classifier heads (Supplementary Figures 2D, 2E) and demonstrate that TCR-specific pre-training data and methodology yields a robust TCR embedding representation even compared to larger, general-purpose models.

As each individual's immune system independently generates TCRs, cross-patient generalization better captures how such antigen classifiers might be applied in a clinical setting. To evaluate this, we study human TRB sequences binding the NP177 influenza A antigen[25], which was not seen during pretraining. These TRBs are primarily derived from a single patient (n=176), whose data we use for training; the remaining (n=38) TRBs are derived from 4 other patients and are used for testing. As this dataset only provides positive examples, we sample non-binding human TRBs from TCRdb at a ratio of 5:1. Among all evaluated models, TCR-BERT provides the best AUPRC on test patients (Figure 2C). Notably, all three highest-performing models were pre-trained on large sets of unlabeled data.

Finally, we evaluate classification performance on a murine LCMV GP33 dataset. This dataset uniquely provides antigen binding annotations for many (n=17,702) TRA/TRB pairs, which we split into 70/15/15 train/validation/test partitions. Given these pairs, we embed the TRA and TRB sequences separately, concatenate the embeddings, and train an SVM (see Supplementary Methods). We evaluate TCR-BERT along with three baseline embeddings: ESM, TAPE, and DeepTCR[30]. Of these, TCR-BERT achieves the best test AUPRC of 0.47, compared to 0.43, 0.38 and 0.40 for TAPE, ESM, and DeepTCR, respectively (Figure 2D). The large number of examples available in this LCMV dataset additionally allows us to evaluate the effectiveness of fine-tuning TCR-BERT. We create two copies of TCR-BERT, both initialized using weights from MAA pretraining, to embed TRA and TRB sequences, respectively. These two embeddings are concatenated and passed through a classifier, and the entire "two-armed" network is trained (see Supplementary Methods). We compare this to a ConvNet with two convolutional "arms" corresponding to TRA and TRB, and various models applied to *k*-mer featurization of TRA and TRB sequences. Fine-tuning TCR-BERT achieves best performance with an AUPRC of 0.61 (Figure 2D, Supplementary Figure 4). This strong performance is largely a consequence of TCR-BERT's pre-training: an architecturally identical network trained without pre-training yields a substantially lower test AUPRC of 0.46.

We repeated all antigen binding prediction experiments additionally requiring that no training sequence can be within 1 Levenshtein edit of any test sequence (2 edits for paired TRA/TRBs). Despite this reduced sequence similarity, TCR-BERT nonetheless exhibits similar performance improvements (Supplementary Figure 5), suggesting that it provides a strong foundation for building antigen classifiers for TCR sequences.

TCR-BERT facilitates unsupervised, explorative analyses of TCRs

In many cases, researchers may not know *a priori* which specific antigen(s) to predict TCR affinity for, necessitating more exploratory analyses of TCR sequence data. TCR-BERT facilitates such analyses via visualization and clustering of its TCR sequence embeddings. To illustrate this, we embed (n=2,067) human TRB sequences[25] using TCR-BERT (pre-trained on MAA and classification) and visualize each TRB using UMAP (Figure 3A). While this does not perfectly separate all antigen binding groups (potentially due to cross-reactivity), it nonetheless highlights distinct specificity groups. For example, the group of TRBs corresponding to the BMLF1 Epstein Barr virus (EBV) antigen (GLCTLVAML) in the upper region of the plot corresponds to the illustrated motif (Figure 3A, top). This motif matches one of the most conserved public (i.e., shared between multiple individuals) EBV TRBs[41], which may contribute to its strong separation from other TRBs.

In addition to examining these embeddings visually, these TCRs can be algorithmically clustered using an algorithm like Leiden[42]. We evaluate this approach with NP177-binding TRB sequences mixed with randomly sampled endogenous human TRBs. Ideally, every cluster should consist of only antigen-binding or background sequences; this is measured by percent correctly clustered[25]. Additionally, each TRB should be clustered with at least a few other TRBs, as assigning each sequence its own group would trivially achieve perfect accuracy; this is captured by percent clustered[25]. We examine several clustering resolutions for TCR-BERT's NP177/background embedding (Figure 3B, blue), and use GLIPH[25] to similarly generate TCR groups of varying granularities for these same sequences (Figure 3B, orange). TCR-BERT's embedding yields clusters that are more correct and stable (Supplementary Figures 6A, 6B), while being faster to compute (Supplementary Figure 7). We repeated TCR clustering comparison using a subset (n=2,443) of the murine LCMV GP33 TRBs. This dataset also provides V/J annotations, which allows us to evaluate TCRDist3[29]. Compared to both these methods, clustering TCR-BERT's TRB embeddings again provides a consistently improved tradeoff between percentage clustered and correctness (Figure 3C, Supplementary Figures 6C-F).

TCR-BERT focuses on biologically relevant residues

To better understand how TCR-BERT achieves these advances, we study the amino acid residues highlighted via TCR-BERT's transformer attention mechanism. At a high level, attentions[43] capture how much a given token's representation (i.e., our classification embedding) is influenced by representations of other tokens (i.e., each amino acid in the TCR CDR3). We examine the two-armed TCR-BERT model fine-tuned to predict LCMV GP33 binding from TRA/TRB pairs. We average each TRA/TRB submodule's per-residue attentions across TRA/TRB pairs of equal length (12 and 14 residues for TRA and TRB, spanning 36 binding and 121 non-binding examples, Figure 4A). TCR-BERT's attentions tend to be concentrated towards the central region of both the TRA and TRB. This aligns with prior works describing a functional hot spot around central residues enabling fine discrimination of antigens[44]. These attentions can also be interpreted in the context of the structural interface between TCR and pMHC. Examining three structures profiling a similar LCMV GP33 system (PDB structures 5m00, 5m01, and 5m02, see Supplemental Methods), we find that TCR-BERT's attention is anti-correlated with distance to the antigen (Figures 4B, 4C). Antigens in the lowest quartile of distances relative to the antigen in each chain correspond to significantly higher attentions ($p = 7.14 \times 10^{-16}$ for TRA, $p = 1.06 \times 10^{-8}$ for TRB, two-sided Mann-Whitney test), indicating that TCR-BERT pays the most attention to TCR residues physically proximal to the antigen (Figure 4D, Supplementary Figure 8).

TCR-BERT can facilitate computational TCR design

TCR-BERT enables novel computational approaches to important experimental and clinical challenges involving TCRs. Among these, one exciting domain is TCR engineering, which seeks to redirect T cell

specificity by introducing synthetic TCR sequences into T cells. In principle, this strategy can boost T cells' ability to recognize a given antigen, thus strengthening immune defense against specific pathogenic or malignant cells[45]. We present a proof-of-concept computational approach to engineering new TCR CDR3 sequences by using TCR-BERT to drive an *in silico* directed evolution process. We start with a set of TCR sequences with no observed binding to an antigen of interest and use TCR-BERT to rank these according to their predicted likelihood of binding. We use the sequences with the highest predicted binding likelihood to generate additional, similar sequences via masked amino acid prediction, leveraging TCR-BERT's learned language model to traverse the complex grammar of TCRs. We repeat the ranking and sampling steps using these newly generated sequences, iterating until we arrive at a set of sequences with desirable predicted binding (Figure 5A, see Supplementary Methods). We apply this procedure to generate TRA-TRB pairs targeting GP33 using the version of TCR-BERT fine-tuned to predict LCMV GP33 binding from TRA-TRB pairs. We start with 100 non-binding TRA-TRB pairs apply the directed evolution process described above. The predicted GP33 affinity grows with successive iterations, reaching a minimum 95% predicted probability of binding after 7 iterations (Supplementary Figure 9A). The top 50 generated TCRs share no clear sequence similarity to training examples with observed GP33 binding (Supplementary Figures 9B, 9C).

To show that we are generating reasonable candidates binding GP33, we use BLAST[46] to match both our initial non-binding TRB set and final engineered TRB set against all known murine TRB sequences (we could not find adequate data to similar evaluate TRAs, see Supplementary Methods). TRBs in the final set not only produce more hits to TRBs independently found to bind GP33 in an experiment never seen by TCR-BERT, but these hits are also more specific (Figure 5B). Additionally, our engineered TRBs produce matches to 11 known GP33 binders that were not matched by our starting set (Figure 5C, Supplementary Figure 10), further illustrating that our TCR engineering process generates novel sequences. Rerunning this TCR engineering procedure using a different set of starting sequences shows that we can consistently generate different GP33 binders (Supplementary Figure 11).

Discussion

TCR-BERT is a large language model trained to model and embed T-cell receptor sequences. Compared to prior works using machine learning to predict TCR behavior, TCR-BERT is uniquely designed to leverage unlabeled data with biologically motivated pre-training tasks to learn a general representation of TCRs, before being applied to or fine-tuned on specific downstream tasks. Leveraging transfer learning, TCR-BERT thus enables state-of-the-art prediction of TCR-antigen binding and grouping of TCRs likely to share antigen specificities. TCR-BERT enables these advances by focusing on structurally relevant residues with known biological importance for binding. Finally, we show how TCR-BERT can enable novel applications such as computationally generating new TCR sequences with desirable binding characteristics.

There are several directions for improving and extending TCR-BERT. Most directly, our proof-of-concept TCR engineering work requires follow-up experimental validations. More broadly, TCR-BERT considers only a subset of information regarding the TCR and does not leverage VDJ gene usage information or CDR1/2 sequences in its design. Previous works have found that these additional annotations can improve antigen binding prediction performance compared to using TCR sequences alone[30], particularly the V gene, but the optimal approach for integrating these into a large language model is an open question. It can be difficult to understand exactly how TCR-BERT embeds or classifies a TCR sequence, especially when compared to traditional methods built around more transparent metrics like sequence similarity. Future work in model interpretability could alleviate this. Finally, TCR-BERT would greatly benefit from additional training data, especially as new technologies emerge improving scalability of profiling TCR sequences and their antigen specificities[47]. Such additional data could help alleviate possible biases in HLA representation in TCR-BERT's current pre-training data.

Code and model availability

The TCR-BERT model and model weights are available at https://github.com/wukevin/tcr-bert.

Author contributions

H.Y.C., J.Z., K.E.W., and K.E.Y. defined the original motivations for this research. K.E.W. designed and performed all modelling, benchmarks, and applications. K.E.Y., B.D., J.A.B., Y.X., T.E., and A.S. performed data collection and preprocessing and access to novel data. All authors provided input for interpreting results and contributed to writing the paper.

Disclosure

H.Y.C. is a co-founder of Accent Therapeutics, Boundless Bio, and Cartography Bio. H.Y.C. is an advisor of 10x Genomics, Arsenal Biosciences, and Spring Discovery. A.T.S. is a co-founder of Immunai and Cartography Bio. J.Z. is affiliated with InterVenn Biosciences. K.E.Y. is a consultant for Cartography Bio. The companies listed had no role in the design, execution, interpretation, or funding of this research.

Acknowledgement

H.Y.C. is supported by RM1-HG007735 and Parker Institute for Cancer Immunotherapy. H.Y.C. is an Investigator of the Howard Hughes Medical Institute. T.E. is supported by R01-AI130152, R21-AI161040, and the Leukemia and Lymphoma Society Scholar Award. J.A.B was supported by a Stanford Graduate Fellowship and the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1656518. K.E.Y. was supported by the National Science Foundation Graduate Research Fellowship Program (NSF DGE-1656518), a Stanford Graduate Fellowship, and an NCI Predoctoral to Postdoctoral Fellow Transition Award (NIH F99CA253729).

Figures

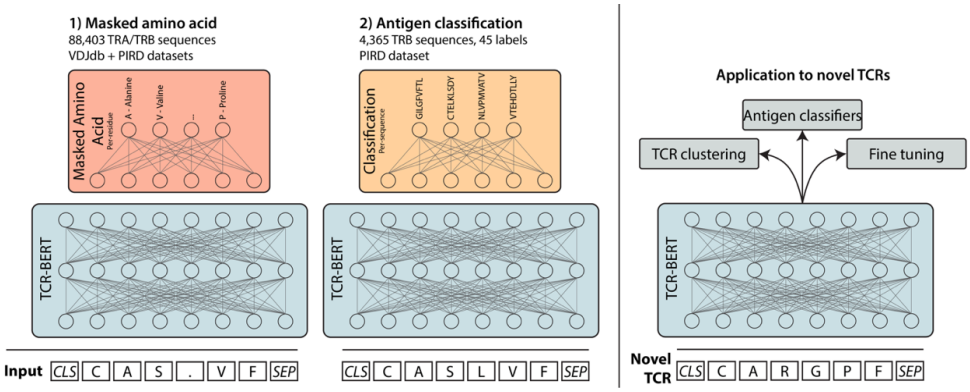


Figure 1: TCR-BERT leverages self-supervised pretraining to model TCRs.

TCR-BERT takes a TCR amino acid sequence (bottom input) and generates an embedding useful for downstream tasks. In pre-training, TCR-BERT learns to predict a masked amino acid (".") based on surrounding residues, thus learning the grammar of natural TCRs. This is done over a large corpus of TRA and TRB sequences with no MHC or HLA restrictions and does not require knowledge of antigen binding (left panel). We subsequently further train TCR-BERT to predict, across a set of 45 antigen labels, the antigen that a given TRB sequence binds to (center panel). We apply TCR-BERT to a variety of TCR analyses, including predicting antigen binding and clustering TCRs (right panel).

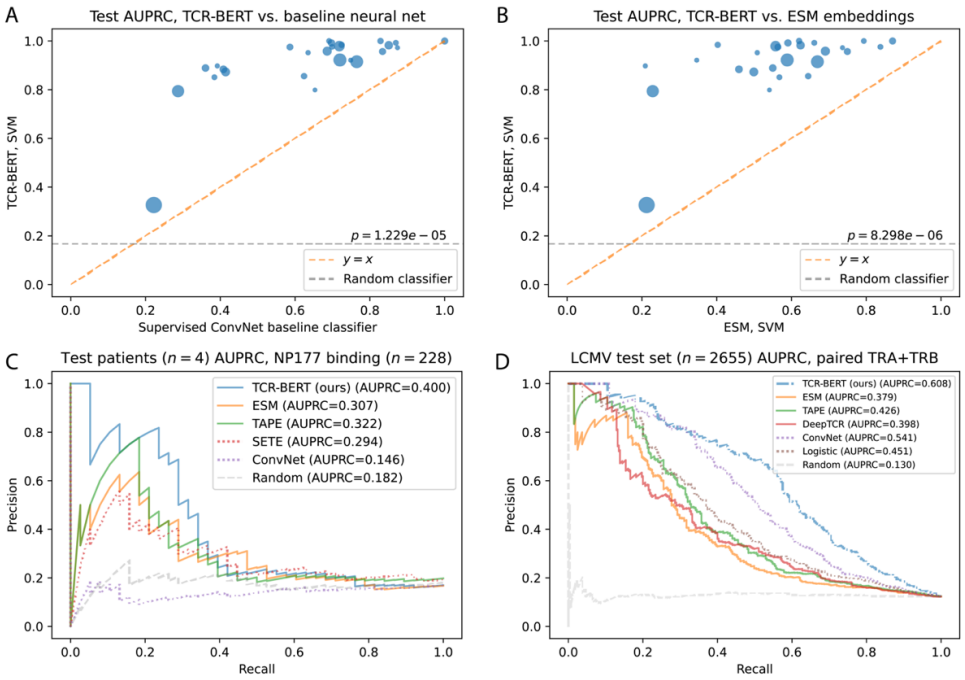


Figure 2: TCR-BERT can be used to build state-of-the-art classifiers predicting antigen binding.

(A) Antigen cross validation comparing an SVM on TCR-BERT embeddings (y-axis) to a supervised ConvNet (x-axis). Each point (n=26) indicates test AUPRC for a single antigen using each of the two methods. Larger points correspond to antigens with more training examples (ranging from 21-231 binding TCRs, with an outlier of 4115). TCR-BERT delivers significantly improved performance (i.e., above orange line indicating equal performance). P-value is computed using a two-sided Wilcoxon test. (B) Antigen cross-validation also shows that TCR-BERT outperforms ESM, a larger, general protein language model. Indicated p-value is computed using a two-sided Wilcoxon test. (C) Evaluates model performance on NP177 antigen where training and test data is split by patient identity. Pre-trained models are shown in solid lines whereas supervised models are shown in dotted lines. TCR-BERT provides the best performance; pre-trained models generally outperform supervised models. (D) Using a murine dataset profiling GP33 binding for TRA/TRB pairs, we evaluate various models' antigen binding predictions on a random test set. Prior to fine-tuning, TCR-BERT's embedding achieves test AUPRC (0.466, curve not shown for clarity). After fine-tuning, TCR-BERT achieves class-leading performance (dot-dash line, see Supplementary Figure 4) compared to both supervised (dotted lines) and embedding methods (solid lines).

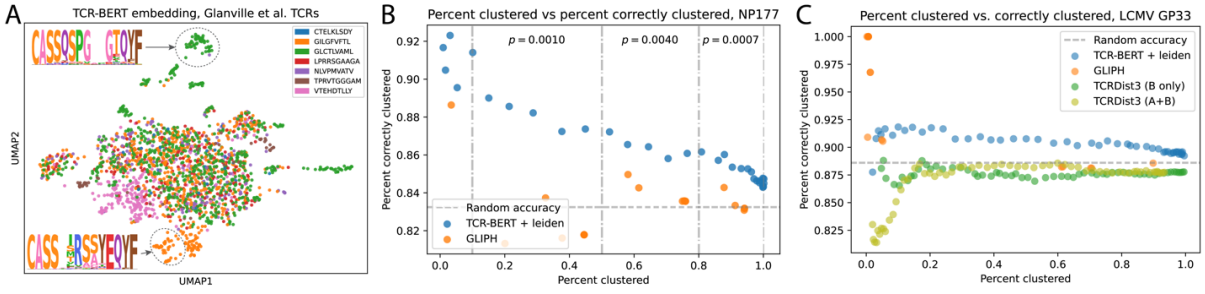


Figure 3: TCR-BERT's embedding enables clustering analyses of patient TCR sequences.

(A) We use TCR-BERT to embed and visualize 2,067 human TRB sequences binding various antigens. Each point represents a TRB sequence, colored by its known antigen binding partner. An unsupervised clustering algorithm like Leiden can be used to identify clusters of TCRs; two examples of identified clusters are circled, along with motifs summarizing constituent TRBs. (B) To systematically quantify the utility of TCR-BERT's embedding for clustering, we study (n=217) TRB sequences binding the NP177 antigen mixed in a negative background of randomly sampled

human TRBs (n=1070). We evaluate several clustering resolutions for TCR-BERT (blue) and a GLIPH (orange), plotting each method's trade-off between percent clustered (x-axis) and percent correctly clustered (y-axis). We define 3 ranges of percent clustered (delineated by vertical lines); within each, TCR-BERT significantly improves clustering accuracy (indicated p-values, two-sided Mann-Whitney test, Supplementary Figures 6A, 6B). Random accuracy indicates the trivial solution with all sequences in a single cluster. This improvement is consistent when evaluating a different LCMV GP33 murine dataset, subsampled to 2,443 TCRs (C). TCR-BERT (blue) provides more improved clustering performance compared to GLIPH (orange) and TCRDist3 (green/olive); see Supplementary Figures 6C-F.

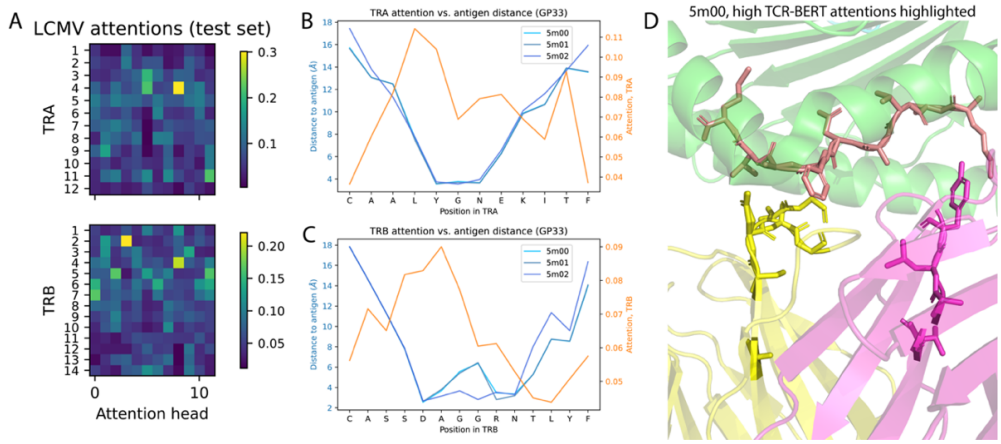


Figure 4: TCR-BERT's attentions reveal biologically meaningful learned patterns.

(A) Heatmaps visualizing the attentions that TCR-BERT has learned for predicting LCMV GP33 binding, averaged across test set TRA and TRB sequences of fixed length. The vertical axis indicates positions in the TRA (top) and TRB (bottom) sequence, and the horizontal axis illustrates each of the 12 attention heads within TCR-BERT. Attentions tend to be concentrated to the center of the TCRs. (B) We relate these averaged attentions to biophysical structures of TCR-antigen binding using three empirical PDB structures (5m00, 5m01, 5m02). Blue lines indicate, for each residue in the TRA, the minimum distance to the antigen in each experimental structure. Orange line indicates TCR-BERT attention for those residues. TCR-BERT pays the most attention to residues closest to the antigen, and the same is true for the TRB sequence (C). (D) We illustrate this with a 3D structure showing the MHC (green), modified GP33 antigen (salmon), and TRA (pink) and TRB (yellow). Side chains are shown and highlighted for the antigen and the TCR residues receiving the top 33rd percentile of attentions. See Supplementary Figure 8 for additional views.

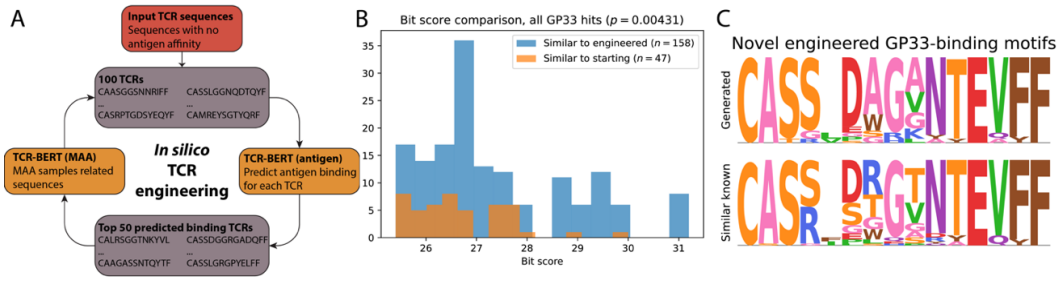


Figure 5: TCR-BERT enables in-silico engineering of novel TCR sequences

(A) To engineer sequences with affinity for GP33, we take endogenous TRA and TRB pairs with no measured binding to GP33, use TCR-BERT to select sequences most likely to bind, and use TCR-BERT's masked amino acid predictions to sample variants of these TCRs. This directed evolution process is repeated to iteratively enrich for desirable binding properties (Supplementary Figure 9). (B) We use BLAST to match starting (orange) and final (blue) engineered TRB sequences against previously characterized murine TRBs. Engineered sequences produce more hits to previously characterized GP33-binding TRBs than the starting set. These hits are also more specific with significantly elevated bit scores, which correspond to the log-size of the database needed to produce a hit by random chance (displayed p-value, two-sided Mann-Whitney test). (C) Our *in silico* engineered sequences match several known GP33-binding sequences not matched by starting sequences, one such match is illustrated here. (see Supplementary Figure 10).

References

1. Kumar, B. V., Connors, T. J. & Farber, D. L. Human T Cell Development, Localization, and Function throughout Life. *Immunity* **48**, 202-213 (2018).

2. Walker, B. & McMichael, A. The T-cell response to HIV. *Cold Spring Harb Perspect Med* **2**, a007054 (2012).

3. Stinissen, P., Raus, J. & Zhang, J. Autoimmune pathogenesis of multiple sclerosis: role of autoreactive T lymphocytes and new immunotherapeutic strategies. *Crit Rev Immunol* **17**, 33-75 (1997).

4. Olsson, T. et al. Autoreactive T lymphocytes in multiple sclerosis determined by antigen-induced secretion of interferon-gamma. *J Clin Invest* **86**, 981-985 (1990).

5. Roep, B. O. The role of T-cells in the pathogenesis of Type 1 diabetes: from cause to cure. *Diabetologia* **46**, 305-321 (2003).

6. Pugliese, A. Autoreactive T cells in type 1 diabetes. *J Clin Invest* **127**, 2881-2891 (2017).

7. Strønen, E. et al. Targeting of cancer neoantigens with donor-derived T cell receptor repertoires. *Science* **352**, 1337-1341 (2016).

8. Schumacher, T. N. & Schreiber, R. D. Neoantigens in cancer immunotherapy. *Science* **348**, 69-74 (2015).

9. Xue, S. et al. Exploiting T cell receptor genes for cancer immunotherapy. *Clin Exp Immunol* **139**, 167-172 (2005).

10. Chandran, S. S. & Klebanoff, C. A. T cell receptor-based cancer immunotherapy: Emerging efficacy and pathways of resistance. *Immunol Rev* **290**, 127-147 (2019).

11. Houot, R., Schultz, L. M., Marabelle, A. & Kohrt, H. T-cell-based Immunotherapy: Adoptive Cell Transfer and Checkpoint Inhibition. *Cancer Immunol Res* **3**, 1115-1122 (2015).

12. Arellano, B., Graber, D. J. & Sentman, C. L. Regulatory T cell-based therapies for autoimmunity. *Discov Med*

Discovery medicine **22**, 73-80 (2016).

13. Bertoletti, A. & Tan, A. T. Challenges of CAR- and TCR-T cell-based therapy for chronic infections. *J Exp Med* **217**, e20191663 (2020).

14. Ketelhuth, D. F., Gisterå, A., Johansson, D. K. & Hansson, G. K. T cell-based therapies for atherosclerosis. *Curr Pharm Des* **19**, 5850-5858 (2013).

15. Singh, N. K. et al. Emerging Concepts in TCR Specificity: Rationalizing and (Maybe) Predicting Outcomes. *J Immunol* **199**, 2203-2213 (2017).

16. Warren, R. L. et al. Exhaustive T-cell repertoire sequencing of human peripheral blood samples reveals signatures of antigen selection and a directly measured repertoire size of at least 1 million clonotypes. *Genome Res* **21**, 790-797 (2011).

17. Robins, H. S. et al. Comprehensive assessment of T-cell receptor beta-chain diversity in alphabeta T cells. *Blood* **114**, 4099-4107 (2009).

18. Qi, Q. et al. Diversity and clonal selection in the human T-cell repertoire. *Proc Natl Acad Sci U S A* **111**, 13139-13144 (2014).

19. Yin, Y. & Mariuzza, R. A. The multiple mechanisms of T cell receptor cross-reactivity. *Immunity* **31**, 849-851 (2009).

20. Wooldridge, L. et al. A single autoimmune T cell receptor recognizes more than a million different peptides. *J Biol Chem* **287**, 1168-1177 (2012).

21. Petrova, G., Ferrante, A. & Gorski, J. Cross-reactivity of T cells and its role in the immune system. *Crit Rev Immunol* **32**, 349-372 (2012).

22. Sewell, A. K. Why must T cells be cross-reactive. *Nat Rev Immunol* **12**, 669-677 (2012).

23. De Simone, M., Rossetti, G. & Pagani, M. Single Cell T Cell Receptor Sequencing: Techniques and Future Challenges. *Frontiers in Immunology* **9**, 1638 (2018).

24. Pai, J. A. & Satpathy, A. T. High-throughput and single-cell T cell receptor sequencing technologies. *Nat Methods* **18**, 881-892 (2021).

25. Glanville, J. et al. Identifying specificity groups in the T cell receptor repertoire. *Nature* **547**, 94-98 (2017).

26. Huang, H., Wang, C., Rubelt, F., Scriba, T. J. & Davis, M. M. Analyzing the Mycobacterium tuberculosis immune response by T-cell receptor clustering with GLIPH2 and genome-wide antigen screening. *Nat Biotechnol* **38**, 1194-1202 (2020).

27. Chronister, W. D. et al. TCRMatch: Predicting T-Cell Receptor Specificity Based on Sequence Similarity to Previously Characterized Receptors. *Front Immunol* **12**, 640725 (2021).

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

29. Mayer-Blackwell, K. et al. TCR meta-clonotypes for biomarker discovery with tcrdist3 enabled identification of public, HLA-restricted clusters of SARS-CoV-2 TCRs. *Elife* **10**, e68605 (2021).

30. Sidhom, J. W., Larman, H. B., Pardoll, D. M. & Baras, A. S. DeepTCR is a deep learning framework for revealing sequence concepts within T-cell repertoires. *Nat Commun* **12**, 1605 (2021).

31. Tong, Y. et al. SETE: Sequence-based Ensemble learning approach for TCR Epitope binding prediction. *Comput Biol Chem* **87**, 107281 (2020).

32. Fischer, D. S., Wu, Y., Schubert, B. & Theis, F. J. Predicting antigen specificity of single T cells based on TCR CDR3 regions. *Mol Syst Biol* **16**, e9416 (2020).

33. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). *Proceedings of the 2019 Conference of the North* 4171-4186 (2019).

34. Ji, Y., Zhou, Z., Liu, H. & Davuluri, R. V. DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. *Bioinformatics* btab083 (2021).

35. Rives, A. et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc Natl Acad Sci U S A* **118**, e2016239118 (2021).

36. Rao, R. et al. Evaluating Protein Transfer Learning with TAPE. *Adv Neural Inf Process Syst* **32**, 9689-9701 (2019).

37. Grechishnikova, D. Transformer neural network for protein-specific de novo drug generation as a machine translation problem. *Scientific Reports* **11**, 321 (2021).

38. Bagaev, D. V. et al. VDJdb in 2019: database extension, new analysis infrastructure and a T-cell receptor motif compendium. *Nucleic Acids Res* **48**, D1057-D1062 (2020).

39. Zhang, W. et al. PIRD: Pan Immune Repertoire Database. *Bioinformatics* **36**, 897-903 (2020).

40. Chen, S. Y., Yue, T., Lei, Q. & Guo, A. Y. TCRdb: a comprehensive database for T-cell receptor sequences with powerful search function. *Nucleic Acids Res* **49**, D468-D474 (2021).

41. DeWitt, W. S. et al. Human T cell receptor occurrence patterns encode immune history, genetic background, and receptor specificity. *Elife* **7**, e38358 (2018).

42. Traag, V. A., Waltman, L. & van Eck, N. J. From Louvain to Leiden: guaranteeing well-connected communities. *Sci Rep* **9**, 5233 (2019).

43. Bahdanau, D., Cho, K. & Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

44. Degano, M. et al. A functional hot spot for antigen recognition in a superagonist TCR/MHC complex. *Immunity* **12**, 251-261 (2000).

45. Rapoport, A. P. et al. NY-ESO-1-specific TCR-engineered T cells mediate sustained antigen-specific antitumor effects in myeloma. *Nat Med* **21**, 914-921 (2015).

46. Altschul, S. F., Gish, W., Miller, W., Myers, E. W. & Lipman, D. J. Basic local alignment search tool. *Journal of Molecular Biology* **215**, 403-410 (1990).

47. Bentzen, A. K. et al. Large-scale detection of antigen-specific T cells using peptide-MHC-I multimers labeled with DNA barcodes. *Nat Biotechnol* **34**, 1037-1045 (2016).

48. Paszke, A. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *arXiv* 1912.01703v1 (2019).

49. Wolf, T. A. et al. Transformers: State-of-the-Art Natural Language Processing Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. 38-45 (2020).

50. Vaswani, A. et al. Attention is all you need. **Advances in neural information processing systems**, 5998-6008 (2017).

51. Loshchilov, I. & Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

52. Daniel, B. et al. Divergent clonal differentiation trajectories of T cell exhaustion. *bioRxiv* 2021.12.16.472900 (2021).

53. McInnes, L., Healy, J. & Melville, J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv* 1802.03426v3 (2018).

54. Joglekar, A. V. & Li, G. T cell antigen discovery. *Nat Methods* **18**, 873-880 (2021).

55. Kelley, D. R., Snoek, J. & Rinn, J. L. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Res* **26**, 990-999 (2016).

56. Movva, R. et al. Deciphering regulatory DNA sequences and noncoding genetic variants using neural network models of massively parallel reporter assays. *PLOS ONE* **14**, e0218073 (2019).

57. Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization. *CoRR* **abs/1412.6980**, (2015).

58. Vig, J. A Multiscale Visualization of Attention in the Transformer Model. *ArXiv* **abs/1906.05714**, (2019).

59. Berman, H. M. et al. The Protein Data Bank. *Nucleic Acids Res* **28**, 235-242 (2000).

60. Edgar, R. C. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics* **5**, 113 (2004).

61. Tareen, A. & Kinney, J. B. Logomaker: beautiful sequence logos in Python. *Bioinformatics* **36**, 2272-2274 (2020).

62. Hunter, J. D. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering* **9**, 90-95 (2007).

63. Virtanen, P. et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat Methods* **17**, 261-272 (2020).

64. Fabian, P. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **12**, 2825-2830 (2011).

65. Harris, C. R. et al. Array programming with NumPy. *Nature* **585**, 357-362 (2020).

66. Wolf, F. A., Angerer, P. & Theis, F. J. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biol* **19**, 15 (2018).

67. Henikoff, S. & Henikoff, J. G. Amino acid substitution matrices from protein blocks. *Proc Natl Acad Sci U S A* **89**, 10915-10919 (1992).

Supplementary Materials & Methods

Datasets & preprocessing

From the pan immune repertoire database (PIRD) [39], we use the TCR-AB database containing 47,040 TRB sequences and 4,607 TRA sequences. Among these, 605 examples are explicitly paired TRA and TRB, and 8,429 examples have annotated antigen specificities. These annotated antigen specificities span 73 unique antigens. We use the PIRD dataset for masked amino acid modelling pre-training, and its labels for antigen classification pre-training and antigen cross-validation. For antigen cross-validation, we include only antigens with 20 or more identified binding sequences.

VDJdb is a curated dataset of T-cell receptor sequences with known antigen specificities [38]. This dataset consists of 58,795 human TCRs and 3,353 mouse TCRs. More than half of the examples are TRBs (n= 36,462) with the remainder being TRAs (n= 25,686). Although these sequences are all labelled with a known antigen binding partner, we only used this dataset during the masked amino acid pre-training step. We empirically found that including this dataset's labels in classification pre-training did not improve the usefulness of TCR-BERT's embedding for downstream classification or embedding tasks (data not shown).

The TCRdb dataset [40] consists of 139,00,913 TRB sequences of unknown antigen binding affinity. While we attempted to leverage TCRdb in our first MAA pre-training step, we empirically found that doing so greatly increased training runtime without yielding improved downstream results (e.g., MAA prediction performance on a held-out set of sequences and antigen classifier accuracy, data not shown). Rather, we use TCRdb as a pool of unseen, naturally occurring human sequences of unknown antigen binding affinity. We sample from these sequences to create a "negative" set of TCR sequences. This is useful for building classifiers from datasets that only describe TCR sequences with known binding affinity (i.e., a positive label), but does not describe TCR sequences with no known binding affinity (i.e., a negative label). These negative sets are sampled at a ratio of 5 TCRdb negatives to each known positive example. This ratio is a round value that approximates the proportion of positive examples in the relatively exhaustive LCMV dataset, discussed below.

The murine LCMV GP33 dataset consists of T cells from the lung, liver, and spleens of mice infected with either LCMV Armstrong or LCMV Clone 13. Following tissue dissociation, single cell suspensions were stained with class I tetramer H-2Db LCMV GP33-41 (KAVYNFATC) (PE) and then sorted via flow cytometry as tetramer high, mid, or negative. TCR sequencing was performed using the 10X 5' Single Cell Immune Profiling Solution Kit (v1.1 Chemistry) using the 10x Chromium Single Cell V(D)J Enrichment Kit for mouse T cells, according to the manufacturer's instructions. Single-cell TCR-seq libraries were sequenced on an Illumina NovaSeq S4 sequencer using the following read configuration 26bp Read1, 8bp i7 Index, 91bp Read2. TCR reads were aligned to the mm10 reference genome and consensus TCR annotation was performed using cellranger vdj (10x Genomics, version 3.1.0). We keep only unique TRA/TRB clones where 80% of annotated cells assigned the same label, in which the clone is assigned the majority label. Overall, this results in (n=17,702) unique TRA/TRB pairs with consistent labels that we use for model training and evaluation. Among these, 13% (n=2306) are observed to have mid or high binding – we consider these "positive" examples of TRA/TRBs binding GP33. For training and evaluation purposes, this dataset randomly divided into a 70/15/15 train/validation/test split. This dataset is also being used for a separate, unrelated manuscript under preparation for submission; this manuscript contains more details surrounding exact experimental conditions for gathering this data as well as details on data access.

For all TCR sequence records, we exclude any sequences that include residues outside the set of 20 standard amino acids, e.g., wildcard residues that indicate variability. All data splits are created such that there are no sequences that appear in both training, validation, and/or test. For some evaluations, we further constrain the training set to not include any sequences that are within 1 edit distance to any test

sequence. Edit distance is calculated using the Levenshtein metric, and training sequences that are not at least 2 edits away from all test sequences are excluded from training. When considering TRA/TRB sequence pairs instead of singular sequences, we required a total edit distance between each training TRA/TRB pair and all test pairs of at least 3 edits (summed across TRA and TRB). This procedure reduces the size of the training set but preserves examples in the test set, which makes results more comparable to models where this constraint is not active.

Modelling, pre-training, and layer selection

TCR-BERT is implemented in Python primarily using the PyTorch [48] and Transformers libraries [49]. TCR-BERT uses a lightly modified version of the BERT language modelling architecture. We provide a brief description of BERT and general transformer models here; for full details, please refer to the original BERT manuscript [33]. Transformer models [50] use a series of blocks that apply attention and feed-forward layers. Attention attempts to model pairwise interactions in an input sequence by learning how strongly the embedding representation of each token should be influenced by the embedding of other tokens (including itself). Applying several layers of such blocks allows models like BERT to learn increasingly complex interactions between tokens that ultimately capture higher-order concepts like grammatical structure. These transformer networks have been shown to outperform more conventional recurrent and convolutional models in various natural language problem settings.

TCR-BERT's input is TCR sequence of length *M*, formatted as a series of *M* tokens spanning the set of 20 amino acids. These input tokens are then padded with special tokens: a classification token *C* as a prefix, and a separator token *S* as a suffix. The padded input tokens are then passed through a trained embedding layer that maps each token (amino acid) to a continuous representation of 768 dimensions. As this token embedding does not capture positional information, our TCR-BERT model follows the BERT model in adding a positional encoding to the amino acid embedding. This summed sequence embedding is then fed through a series of 12 transformer blocks to arrive at the overall sequence embedding. This sequence embedding represents each input amino acid chain as a $(M + 2) \times 768$ matrix. The sequence embedding can then be fed into various "heads" that perform pretraining and downstream tasks such as masked amino acid prediction or sequence classification.

We pre-train TCR-BERT using two objectives optimized sequentially. First, we pretrain using a masked amino acid (MAA) modelling objective, where we randomly hide, or "mask" 15% of the amino acids in each TCR amino acid sequence in the training set, and train TCR-BERT to predict these masked amino acids. Architecturally, this is done by appending a MAA "head" network to the previously described transformer network. This MAA head is a simple, fully connected layer that maps TCR-BERT's per-residue hidden representation to logits (20 dimensions, corresponding to each amino acid), followed by a softmax activation. MAA pretraining is done using both TRA and TRB sequences from the VDJdb and PIRD datasets and aims to leverage the large amount of TCR sequences with (potentially) unknown antigen specificity to learn the "grammar" of a valid TCR sequence. TRA and TRB sequences are given to the model without features or flags distinguishing the two. We use a random 85/15 train/test split for MAA pre-training and tune hyperparameters based on test set loss (a validation set is not used as we do not directly benchmark MAA predictions). We perform grid search across the following hyperparameters with final chosen values in **bold**. Several of these hyperparameters describe architectural configurations (e.g., dimension of the hidden representation) whose values apply to downstream training/tasks as well. Default values for the BERT architecture are indicated. Hyperparameters and architectural configurations not indicated are left at their default values.

- Hidden representation dimensionality: [144, 384, **768**] (BERT default: 768)
- Intermediate representation dimensionality: [**1536**, 3072] (BERT default: 3072)

- Number of attention heads: [6, 8, **12**] (BERT default: 12)
- Number of transformer layers: [8, **12**] (BERT default: 12)
- Batch size: [128, **256**]
- Learning rate warmup (number of training steps to linearly "ramp up" learning rate to specified value): [**0**, 0.1]
- Training epochs: [10, 15, 25, **50**, 100]
- Learning rate: [2e-5, **5e-5**]

We also reduce the maximum positional embedding length to be 64 (as opposed to BERT's default maximum of 512) to reflect the relatively short TCR sequence lengths compared to sentence lengths in natural language. We use linear learning rate decay to 0 over training epochs, coupled with the AdamW [51] optimizer and negative log likelihood loss.

The second pre-training objective is a multi-class classification task where we train TCR-BERT to classify each input TRB sequence as binding to a one of a set of profiled known antigens. We focus on TRB sequences exclusively, as very few TRA sequences have annotated antigen specificities (and even fewer paired TRA-TRB examples have such annotations). Consequently, TCR-BERT is only suitable for analyzing TRB when using this pre-training step. To train this objective, we subset the antigens represented in the PIRD dataset to include only antigen sequences with at least 6 positive examples; antigens with fewer positive examples are aggregated into a single "other" label. This results in 44 antigen labels and 1 "other" antigen label for a total of 45 labels, distributed among with 6,235 TRB sequences. These are randomly split into training, validation, and test sets using a 70/15/15 split. The antigen classification "head" consists of a (somewhat misleadingly named) "pooling" layer (a feedforward network projecting the 768-dimensional classification token embedding into 768 dimensions with Tanh activation) and a "classifier" layer (a feedforward network projecting the 768-dimensional output of the pooling layer into the number of labels). As this pre-training objective predicts a single antigen for each TRB, we use a softmax activation coupled with a negative log likelihood loss. For this pre-training objective, we perform grid search over the following hyperparameters optimizing for validation set AUPRC, with selected values in bold:

- Learning rate: [2e-5, **5e-5**]
- Batch size: [**128**, 256]
- Learning rate warmup: [0, **0.1**]
- Training epochs: [10, 15, **25**, 50]

We additionally perform early stopping after no improvement in validation set AUPRC after 5 epochs.

Pre-training ablations

Antigen cross-validation also allows us to investigate the impact of each of our two pre-training steps on the quality of TCR-BERT's embeddings. We evaluated the performance of an SVM on TCR-BERT's embedding, compared to an embedding trained using only antigen classification (holding out antigens as necessary) and an embedding trained using only MAA. For both pre-training ablations, AUPRC classifying unseen antigens is significantly reduced compared to the full TCR-BERT model (Supplementary Figures 3A, 3B), demonstrating the importance of both of TCR-BERT's pre-training steps in learning an embedding that generalizes to unseen antigens and their associated TRBs. We also compared TCR-BERT's pre-training ablations against ESM and TAPE embeddings and found that TCR-BERT trained using only MAA exhibits a small performance improvement, whereas using only classification pre-training exhibits a larger performance advantage (Supplementary Figures 3C-F). This suggests that while training on specifically TCR sequences (as opposed to all protein sequences) is helpful, TCR-BERT's performance improvements stem primarily from its unique antigen classification pre-training task. More generally, our observation that TCR-BERT outperforms ESM and TAPE, which are both pre-trained on much larger datasets, suggests that

leveraging datasets and objectives that are biologically related to downstream applications can have a larger impact than sheer dataset size.

Cross-species validation of pre-trained TCR-BERT model

We performed several evaluations to check the quality and generalizability of TCR-BERT's two-stage pre-training using a new dataset of (n=17,702) TRA/TRB sequence pairs from mice infected with lymphocytic choriomeningitis virus (LCMV) clone 13 antigen GP33, a model of chronic viral infection (see Methods for additional details) [52]. This dataset is not used in training and is murine whereas 97% of TCR-BERT's pre-training data comes from humans. Nonetheless, TCR-BERT accurately predicts masked amino acids for both TRA and TRB sequences (Supplementary Figure 1A), suggesting that the model has learned a generalizable TCR grammar. After the second antigen classification pre-training step, we visualize TCR-BERT's amino acid embedding vectors and observe some separation by biochemical properties (Supplementary Figure 1B). We additionally use TCR-BERT to generate an embedding for each TRB sequence in this GP33 dataset, and visualize the embeddings using UMAP [53] (Supplementary Figure 1C). We observe that TCR-BERT's sequence embedding reflects sequence similarity (Supplementary Figure 1C, 1D), which in turn correlates with shared antigen binding affinities [25,54]. This suggests that TCR-BERT's sequence embeddings capture meaningful structures and relationships within the TCR sequence space.

Downstream classifiers and layer selection

After pre-training, we can use TCR-BERT to obtain a sequence embedding by averaging across each input amino acid's embedding. For this embedding, we select a representation layer from TCR-BERT that is most conducive to downstream tasks like clustering or building classifiers. This is necessary as empirical works have found that the last layer of large language models like BERT is sometimes too specialized towards pre-training tasks to be generally useful for embedding new inputs. To evaluate the quality of each layer, we measure the held-out performance of a simple classifier trained upon it – however, since it is also unclear which classifier is ideal, we jointly evaluate several classifiers. Specifically, we consider the last six layers of TCR-BERT, and for each layer's embedding, we train four models: a logistic regression (LR) consuming the embedding, a support vector machine (SVM) classifier with Gaussian radial basis function kernel consuming the embedding, a logistic regression on the top N principal components spanning at least 90% of the variance (PCA-LR), and a SVM on the same principal component input (PCA-SVM). LR and SVM represent linear and nonlinear classifiers. Building a classifier on top of PCs rather than the raw features may help the classifier better focus on major sources of variation in the data.

We perform this combinatorial evaluation using the LCMV GP33 dataset, measuring validation AUPRC given TRB sequences as input. For training, we randomly subset the LCMV training set to different sizes to mimic different sizes of datasets commonly encountered in real-world TCR analyses, sampling to 50% of the training examples (n=5304 unique sequences), 20%, 10%, 5%, and 2% (n=212 sequences). The size of the validation set does not change. Evaluating these varying levels of available training data, we find that SVM applied to the final layer embeddings (without PCA) most consistently provides the best validation set performance. Both forms of logistic regression perform comparably poorly. PCA-SVM provides similar performance, but its improvements lack the statistical significance and consistency to justify its additional complexity. Thus, all downstream tasks consuming TCR-BERT's embedding (i.e., embedding without fine-tuning and clustering) use the final embedding. All downstream classifiers that do not involve additional fine-tuning use this final embedding with an SVM. For applications embedding both TRA and TRB sequences, we use the version of TCR-BERT pre-trained only on MAA to embed the TRA and the version of TCR-BERT with full pre-training to embed the TRB. This is done due to the second pre-training step using only TRB sequences. Note that this layer and classifier selection process is only performed once using a murine dataset and is not re-tuned for different datasets (including human sequences), for antigen cross-validation, or for different pre-training schemes.

TCR-BERT can also be fine-tuned to perform classification. Rather than treating TCR-BERT as a fixed black box for generating continuous embeddings from discrete amino acid sequences, fine-tuning modifies TCR-BERT using the pre-trained parameters as a starting point. In our work, we fine-tune TCR-BERT to perform antigen binding prediction given a TRA and TRB sequence pair. Architecturally, this paired prediction model is comprised of two separate TCR-BERT transformers individually responsible for embedding the TRA or TRB sequence, respectively. Both are initialized using weights from the MAA pretraining step, as MAA pre-training is agnostic of TRA/TRB identity. We extract the initial "classification" token embedding from both the TRA and TRB, concatenate the two embeddings, and apply a fully connected layer mapping to two outputs with softmax activation to generate binding predictions. Training this overall model tunes each encoder towards patterns specific to either the TRA or TRB. Hyperparameters for this model are selected maximizing validation set AUPRC using grid search over the following values (final values are in bold):

- Learning rate: [5e-5, **3e-5**, 2e-5]
- Training epochs: [10, **25**, 50, 100]
- Dropout (in final fully connected layer, does not affect TCR-BERT itself): [0.1, **0.2**]

We train using a batch size of 128, linear learning rate decay over training epochs, and the AdamW optimizer. While we focus on fine-tuning targeting both TRA and TRB pairs, TCR-BERT could also be fine-tuned to target only TRB sequences using a similar approach without concatenating multiple embeddings.

In several of the experiments described in our work, we aim to build a classifier distinguishing human antigen binding TRB sequences from a random background, sampled at 5 background sequences per binding sequence. To create this random background, we randomly select TRB sequences from the TCRdb dataset [40], which contains human TRB sequences of undetermined antigen affinity (see above). This process attempts to mimic a natural sample of TCRs, where there are many "background" TCRs and a subset of TCRs with binding to a specific antigen. This ratio is similar to what is observed in tissues from severely diseased mice, though this ratio is likely lower in humans.

To evaluate classifier performance, we primarily focus on area under the precision recall curve (AUPRC). AUPRC is a much more informative metric in cases of class imbalance towards negative cases (i.e., when there are very few positive labels). Such class imbalance is commonly observed in the context of TCR specificity, where most TCRs will not bind to a given antigen. We also present area under the receiver operating characteristic (AUROC) in some instances for completeness. The expected AUPRC for a random classifier is the proportion of positive examples the dataset contains, and the expected AUROC is 0.5.

Clustering TCR embeddings

Our method for using TCR-BERT to embed TCR sequences for clustering analysis shares parallels with our method for building a SVM on top of TCR-BERT's embeddings as it uses the previously identified optimal transformer layer (i.e., the last layer). We visualize the resulting embedding using Uniform Manifold Approximation and Projection (UMAP) [53] and cluster sequences using the Leiden clustering algorithm [42]. To control the number and granularity of output clusters, we vary the resolution parameter for Leiden.

To evaluate clustering performance, we use two metrics originally developed to quantify the performance of the GLIPH algorithm [25]: percent clustered and percent correctly clustered. Each group of TCR sequences with 3 or more TCR sequences is considered "clustered." The number of such clustered TCRs divided by the total number of TCRs gives the percent clustered. To calculate percent correctly clustered, we iterate over each cluster and for each TCR in that cluster, we retrieve its associated label (i.e., the antigen that that TCR binds to). If there is a dominant (≥50% occurrence) label within the cluster, the entire cluster is assigned that dominant label. If there is no dominant label, the entire cluster is considered incorrectly

clustered. Percentage correctly clustered is the average accuracy of each clustered TCR evaluated against this majority label. As a toy example, consider a single cluster with sequences $[a, b, c, d, e]$ and corresponding antigens labels $[x, x, x, x, y]$. The percent correctly clustered is 80%, as this would be the accuracy if we had assigned the x label to all points in the cluster. Assuming these are the only five sequences in our experiment, the percent clustered would be 100%.

To evaluate clustering runtime, we use the UNIX "time" command with TCR-BERT and GLIPH methods. Timing includes the entire process from reading a new input of TRB sequences, calculating clusters, and writing relevant output files. Runtime benchmarking is done with different subsets of murine LCMV GP33 TRBs. All benchmarks are run using the same machine with an Intel i9-9960X processor, 128GB of RAM, and an Nvidia GeForce RTX 2080Ti GPU. No other foreground processes were run during benchmarking.

Evaluated external methods

We use various tools to contextualize TCR-BERT's ability to perform classification and clustering. GLIPH was downloaded from the authors' GitHub repository: https://github.com/immunoengineer/gliph. To cluster our TRB sequences, we run GLIPH using the "group discovery" script using varying values for the global convergence cutoff. We attempted to evaluate the updated GLIPH2 algorithm [26], but this tool does not appear to provide a clear mapping of which input sequences constitute which output clusters, so cannot be easily evaluated according to our framework.

We downloaded TCRDist3[29] version 0.2.2 via Python pip, according to the authors instructions at https://tcrdist3.readthedocs.io/en/latest/index.html#. We then use TCRDist3 to generate a matrix of pairwise similarities given a list of TRB sequences and corresponding V/J gene usage, or a list of TRA/TRB sequence pairs and corresponding V/J gene usage, using default parameters in both cases. The pairwise similarity matrix for TRA/TRB pairs is the sum of the TRA matrix and the TRB matrix, as the authors suggest. As the authors suggest using hierarchical clustering to identify neighborhoods, we apply agglomerative clustering (with average linkage) to this similarity matrix, varying the number of identified clusters to obtain clusters of various resolutions.

To evaluate against the Evolutionary Scale Model (ESM), we downloaded the ESM-1b model from the PyTorch model hub. To embed TCR sequences, we use the default mode of extracting the final transformer layer's embedding and average the embedding for each amino acid to obtain the embedding for the overall sequence. This follows the recommendations given in the original authors' GitHub repository https://github.com/facebookresearch/esm. These embeddings are then used to train an SVM (ESM performs similarly when using logistic regression and SVM).

The TAPE model [36] was downloaded from the authors' GitHub: https://github.com/songlab-cal/tape. We use the UNIREP version of their model to generate averaged embeddings using the default configuration. These embeddings are then used to train an SVM (TAPE performs better with an SVM than logistic regression).

The SETE model [31] does not provide a simple code interface to train and evaluate on a dataset; we instead used the authors' manuscript and reference code from their GitHub repository: https://github.com/wonanut/SETE to re-implement their algorithm.

The DeepTCR model [30] version 2.0.10 was installed via Python pip. To evaluate DeepTCR's performance on our GP33 dataset, we first train DeepTCR on the example murine antigen data using TRA/TRB featurization (i.e., excluding VDJ annotations for comparability). We then freeze DeepTCR's parameters and use it to embed our LCMV GP33 TRA/TRB sequence pairs. These embeddings are then used as input to an SVM. We also evaluated other classifiers such as logistic regression on top of DeepTCR's embeddings, but SVM yielded the best performance.

We developed an in-house convolutional architecture (ConvNet) as an additional baseline for predicting binary antigen binding given TCR sequences. This architectural choice is motivated by the fact that many researchers have broadly demonstrated strong results using convolutional networks to perform classification and motif discovery within biological sequences [30 55 56]. ConvNet maps amino acids to a 16-dimensional embedding, followed by convolutional layers mapping to 32, 32, and 16 channels with kernel sizes of 5, 5, and 3 respectively. The output of the final convolutional layer is then passed through a fully connected layer to a 2-dimensional output with softmax activation to predict antigen binding probability. ConvNet is trained using the Adam optimizer [57] with cross entropy loss, a learning rate of 0.001, and a batch size of 512 along with early stopping after 25 epochs of no improvement to validation AUPRC. When training on paired TRA/TRB sequences, we modify ConvNet to learn a separate embedding and convolutional portion for each chain. The final convolution embeddings are then concatenated and input to a single fully connected layer with softmax activation for output probabilities.

Interpreting and contextualizing model attentions

To obtain TCR-BERT's attention across TRA and TRB chains, we use the version of TCR-BERT fine-tuned to perform LCMV GP33 antigen prediction. Recall that this two-armed model contains two fine-tuned variants of the TCR-BERT model embedding TRA and TRB sequences, respectively. Thus, for each chain within the TCR, we examine the respective fine-tuned TCR-BERT arm and extract the model attentions from the first classifier token at the last transformer layer for each of the 12 attention heads. We then trim these attentions to the length of the input L (excluding padding tokens). This results in a matrix of shape $12 \times L$ for each chain in each example. This approach is heavily inspired by the bertviz library [58].

We average these attention matrices across test set examples to approximate typical TCR-BERT attentions. For simplicity, we restrict to test examples with the same length TRA and TRB (n=157, 12 and 14 residues for TRA and TRB, respectively). Test set examples are used as they are not seen for training or hyperparameter tuning. When contextualizing TCR-BERT's attentions, we average across the 12 attention heads to obtain a vector of length L.

We contextualize TCR-BERT's per-residue attentions using the antigen distance of each residue, computed from the 3D structure of the MHC-antigen-TCR complex. We define antigen distance as the minimum Euclidean distance between a given TCR residue and any residue in the antigen peptide. Each residue's 3D coordinates are summarized as an average of the atoms comprising the residue to alleviate computationally intensive pairwise atom calculations. We apply this to Protein Data Bank (PDB, https://pdb101.rcsb.org/) [59] structures 5m00, 5m01, and 5m02. These three structures exhibit minor differences from our dataset. They study a slightly modified LCMV GP33 antigen (KAVANFATM versus our antigen sequence KAVYNFATC) interacting with the TRA/TRB pair CAALYGNEKITF/CASSDAGGRNTLYF (also of lengths 12 and 14 residues, respectively). This specific TRA/TRB pair is not predicted to bind to our GP33 antigen, but it is unclear whether this is driven by the differences in the specific antigen or by model error. We believe that our results should be robust despite these minor differences, as the overall structure of these interactions should be similar.

TCR engineering

Our *in silico* TCR engineering process begins with a set of starting sequences with no binding affinity for the antigen in consideration. In our case, this consists of 100 non-binding TRA/TRB pairs randomly selected from the LCMV GP33 test set. Selecting from the test set ensures that the model has not been trained or tuned on these specific sequences, as would be the case for real-world usage. We then give these TRA/TRB pairs to the version of TCR-BERT fine-tuned to predict LCMV GP33 binding form TRA/TRB pairs, ranking them by their predicted GP33 binding. We take the top half of these sequences (n=50) and use them as seeds to generate a set of new sequences (n=100).

To sample a new TRA/TRB pair, we start by randomly selecting one of the given seed sequence pairs. We then use TCR-BERT to mutate both the TRA and TRB, introducing two amino acid mutations to each chain. Mutations are introduced incrementally by choosing a single random position within the chain, masking that position, and giving the masked input to TCR-BERT (pretrained on MAA only) to predict the masked amino acid. This yields a probability distribution describing the most likely amino acids given the rest of the sequence. We rely on this prediction to explore the "grammar" of valid TCRs, such that we do not generate sequences that are untenable (e.g., a sequence entirely consisting of a single repeated residue). We randomly sample from the top 5 amino acids in this distribution.

The (n=100) newly generated TCR chains are given to TCR-BERT to re-rank, and the process of using the top sequences to re-generate new TCR chains with (hopefully further) enhanced binding is repeated. This cycle is iterated until the predicted binding converges to a satisfactorily high value (in our example, minimum of 95% predicted binding probability).

We use BLAST [46] to check the resulting TRB sequences against known murine TRB sequences. We construct a custom BLAST database using all protein sequences from RefSeq protein matching the query string "t cell receptor beta chain[All Fields] AND "Mus musculus"[porgn]" (n=2467). We then use BLAST version 2.5.0 to match sequences against this database using an E-value cutoff of 0.001. For a baseline comparison, we ran the top 50 predicted GP33 binding starting sequences through BLAST as well against this same database. We compared the number of resulting GP33-related hits versus other, non-related hits for the starting and final BLAST matches using Fisher's exact test. We additionally compare the GP33-related bit values (proportional to the log-size of the database required to produce such a hit by chance) corresponding to matches to the starting set, and matches to the engineered set, using a Mann-Whitney test.

Miscellaneous external libraries

Baseline neural networks and fine-tuned TCR-BERT models were developed using version 0.10.0 of the 20corch library. 3D protein structures are visualized using PyMOL (The PyMOL Molecular Graphics System, Version 2.5 Schrödinger, LLC.). Motif logos are generated by using MUSCLE (version 3.8.1551) [60] to generate a multiple sequence alignment that is visualized using the Logomaker Python package [61]. All other plots were generated using the matplotlib [62] and seaborn libraries. All metrics were computed using the scipy [63], scikit-learn [64], and numpy [65] libraries. We additionally use scanpy (version 1.7.1) and anndata (version 0.7.5) to simplify implementation of select clustering analyses [66].

Data availability

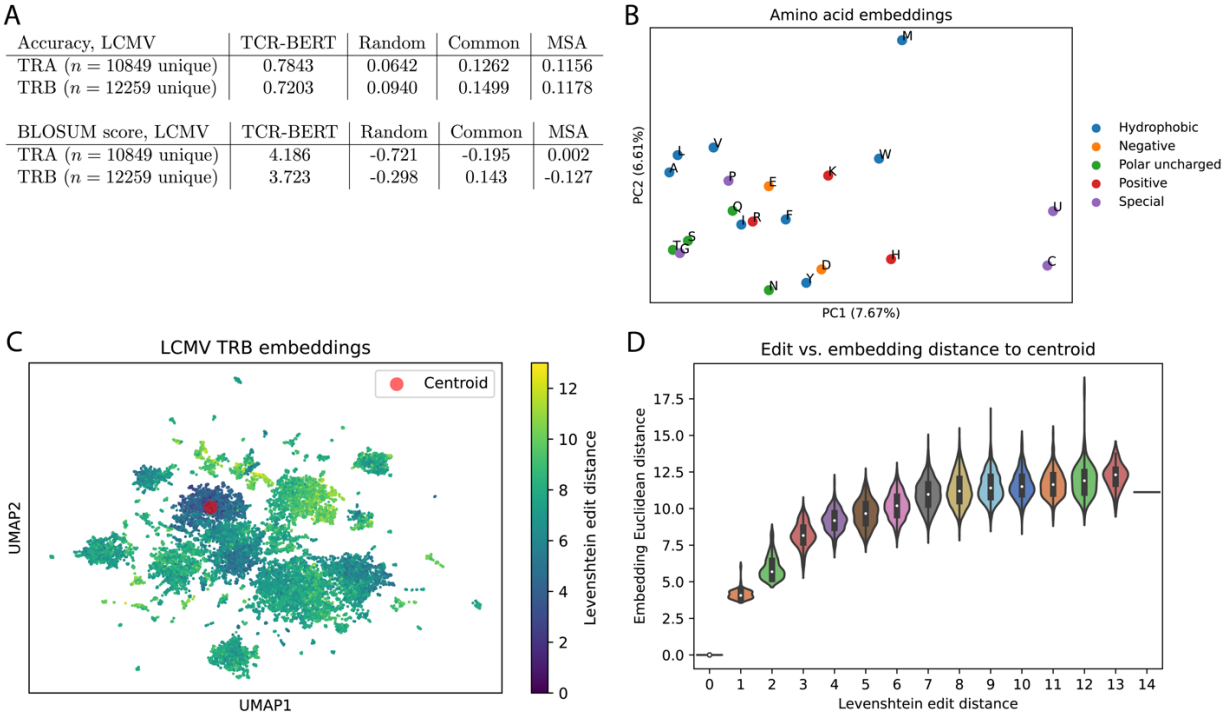
All data used for training TCR-BERT can be found from their respective public databases: PIRD at https://db.cngb.org/pird/, VDJdb at https://vdjdb.cdr3.net, and TCRdb at http://bioinfo.life.hust.edu.cn/TCRdb/#/. TCR sequences from Glanville et al. [25] are available from the original authors' publication at https://doi.org/10.1038/nature22976. Data used for fine-tuning on LCMV GP33 is currently being used to prepare an unrelated, separate experimental manuscript; the LCMV data will be made publicly available when that work is completed and submitted for review. PDB structures for contextualizing TCR-BERT model attentions publicly available at the following accessions: 5m00, 5m01, and 5m02. The murine TCR database used to evaluate our synthetically generated TCRs is available under a subfolder within our GitHub repository: https://github.com/wukevin/tcr-bert/tree/main/data/mus_musculus_trb_blastdb. Local copies of all datasets used in the work are also available under our GitHub repository https://github.com/wukevin/tcr-bert should any of the original sources become unavailable.

Supplementary Figures

A

Accuracy, LCMV	TCR-BERT	Random	Common	MSA
TRA ($n = 10849$ unique)	0.7843	0.0642	0.1262	0.1156
TRB ($n = 12259$ unique)	0.7203	0.0940	0.1499	0.1178

BLOSUM score, LCMV	TCR-BERT	Random	Common	MSA
TRA ($n = 10849$ unique)	4.186	-0.721	-0.195	0.002
TRB ($n = 12259$ unique)	3.723	-0.298	0.143	-0.127



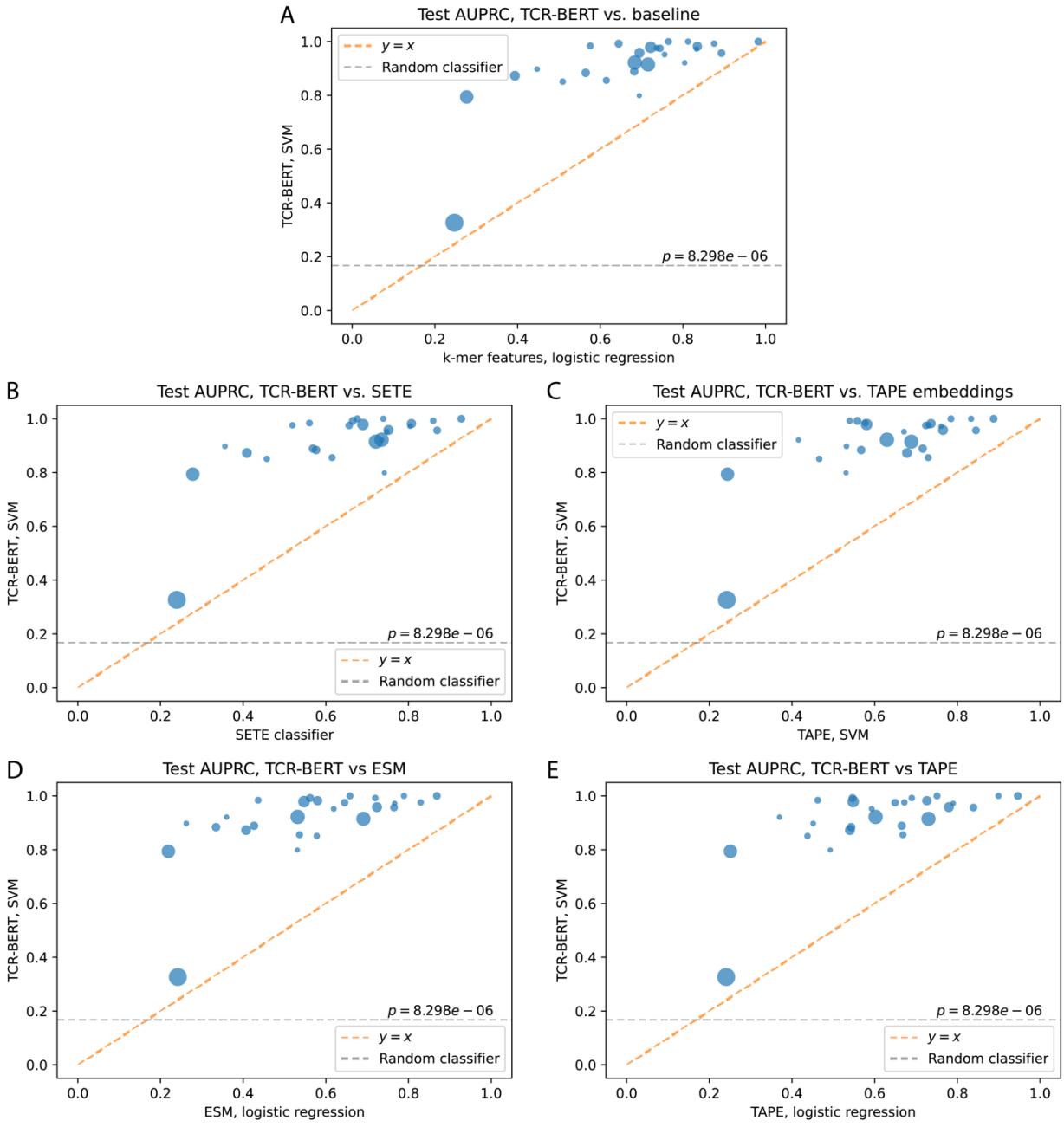
Supplementary Figure 1: Validating TCR-BERT pretraining

(A) TCR-BERT's performance on masked amino acid prediction on an LCMV GP33 murine dataset, which is not used for pre-training. This evaluation is done after performing the first masked amino acid pre-training task, which exposes TCR-BERT to both TRA and TRB sequences (without indicator distinguishing the two). One amino acid at a random position is masked from each input sequence, and TCR-BERT's top prediction is evaluated against the masked amino acid using accuracy (i.e., exact match, top table) or the BLOSUM62 [67] scoring matrix (biochemically-motivated "fuzzy" matching, bottom table). In both cases, higher values indicate more correct predictions. We compare TCR-BERT's performance to three baselines: predicting a random amino acid (random), predicting the most common amino acid in this LCMV dataset regardless of position in the sequence (common), and predicting the most common amino acids at each position after performing a multiple sequence alignment across all LCMV sequences of the given chain (MSA). TCR-BERT outperforms all baselines for predicting masked amino acids, which suggests that it captures complex, nontrivial patterns in the grammar of TCR sequences. Examining TCR-BERT's errors, we see that for TRA sequences, TCR-BERT most commonly predicts a valine in place of a leucine, both of which have small hydrophobic side chains. For TRB sequences, TCR-BERT most commonly predicts a serine in place of a glycine.

(B) TCR-BERT's learned embeddings for each of the 20 amino acids, visualized using PCA. Points are colored by biochemical properties and labelled using standard one-letter abbreviations. We observe some limited separation according to biochemical properties of these amino acids, such as by hydrophobicity along the x-axis. Biochemically similar residues also appear to have similar embeddings, e.g., alanine (A), leucine (L), and valine (V). However, given that the top two PCs jointly describe less than 15% of the total variance, these interpretations are inherently limited.

(C) UMAP visualization of TCR-BERT's embedding of (n=17,702) TRB sequences in the LCMV dataset. Each point represents one TRB sequence and is colored by Levenshtein edit distance to the randomly chosen "centroid" sequence (red). TCR-BERT embeds similar sequences in similar space; distant sequences are more dissimilar with larger edit distances. Since sequence similarity is a known heuristic for predicting TCR binding, this suggests that TCR-BERT's embedding is conducive to downstream TCR sequence analyses.

(D) Violin plot comparing Levenshtein edit distance between TRBs relative to the centroid TRB shown in (C, red) and their Euclidean distance in TCR-BERT's embedding space. X-axis denotes the (discrete) edit distance to the centroid, and y-axis denotes the distribution of (continuous) Euclidean distances for all TRBs with that edit distance. Within each distribution, center black bar denotes the inter-quartile range, and the white dot indicates the median. We observe a strong correlation between edit and embedding distance, which lends additional support to the observation made in (C).



Supplementary Figure 2: Additional antigen cross-validation performance comparisons

In each experiment, we take one of 26 antigens and its associated binding TRBs and spike in background human TRBs at a ratio of 5:1. We evaluate test AUPRC using a random 70/30 train/test split. In each panel, the grey line indicates performance of a random classifier, and the orange line indicates equal performance between the compared methods. Points above the orange line indicate antigens where the y-axis model performs better, and vice versa. Annotated p-values are computed using a two-sided Wilcoxon test evaluating whether TCR-BERT (y-axis) exceeds performance of the benchmarked method (x-axis). As with Figures 2A/B, the size of each point indicates the amount of data available for that TCR.

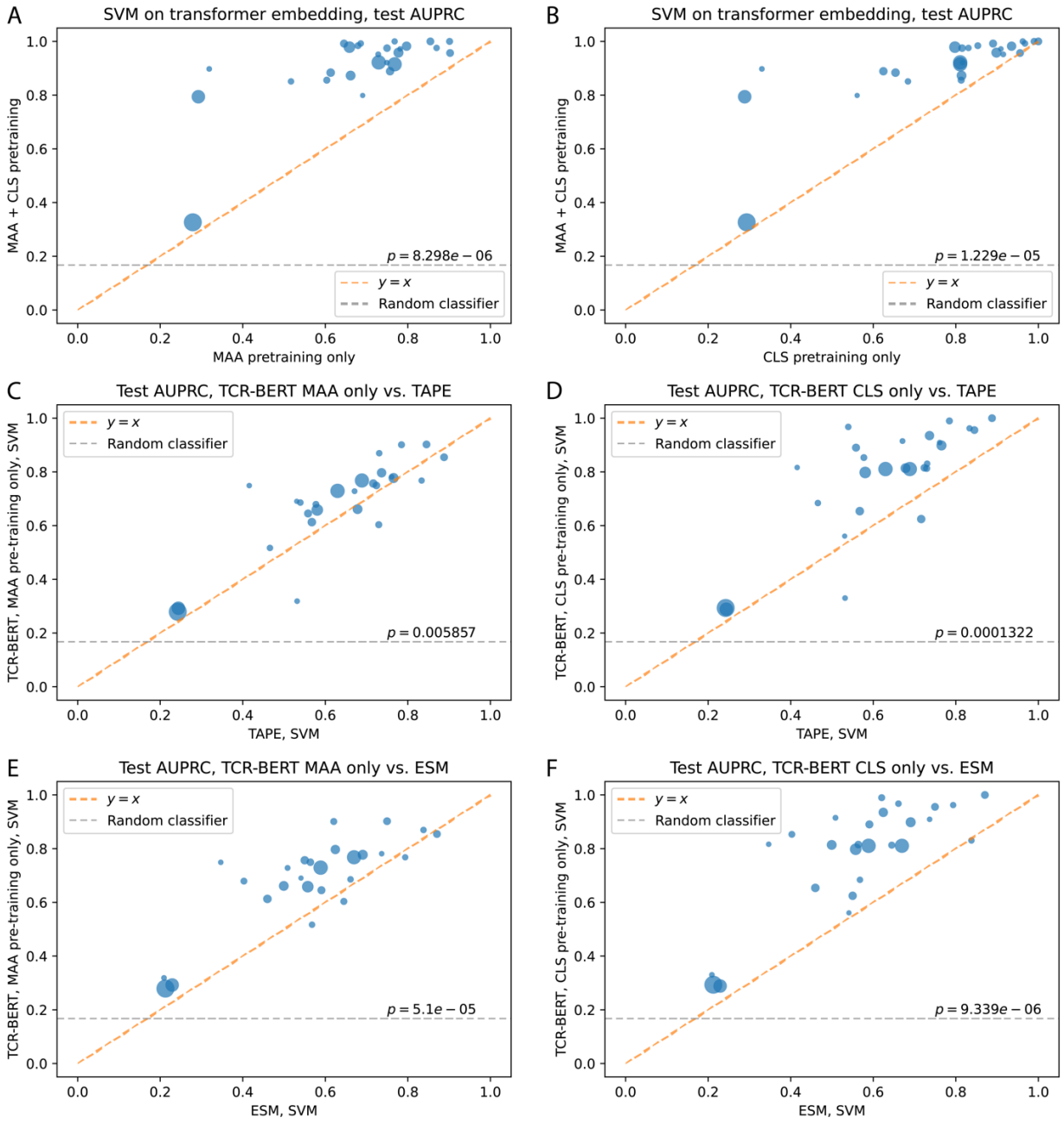
(A) Compares TCR-BERT's performance against that of logistic regression applied to k-mer featurization of the TCR sequences. This represents one of the simplest modelling approaches for sequence classification. TCR-BERT exceeds the performance of this simple baseline in all cases.

(B) Compares SETE, a supervised tree-based model on *k*-mers (x-axis) to SVM on TCR-BERT's embedding (y-axis). For all antigens, TCR-BERT exhibits improved test set AUPRC.

(C) Compares performance of SVM using TAPE's embeddings (x-axis) versus TCR-BERT's embeddings (y-axis). In every case, TCR-BERT provides superior performance. This test isolates the effect of using different protein language models to embed TCRs while keeping the classifier module constant. Along with Figure 2B, this indicates that TCR-BERT outperforms general purpose protein language models for TCR embedding and modelling.

(D) Evaluates a SVM classifier on TCR-BERT's representation against logistic regression on ESM's representation. Overall, we see that ESM's embedding performs similarly when consumed by an SVM or logistic regression. We see that TCR-BERT similarly outperforms ESM here as well, indicating that regardless of the classifier used for benchmark models, TCR-BERT provides superior performance.

(E) Similarly evaluates SVM on TCR-BERT's representation against logistic regression on TAPE's representation. Again, TCR-BERT provides superior performance regardless of the model applied on top of benchmark models' representation. In fact, TAPE's embedding yields significantly better performance when coupled with an SVM compared to logistic regression (measured across antigen cross-validation, $p=0.003$, two-sided Wilcoxon test).

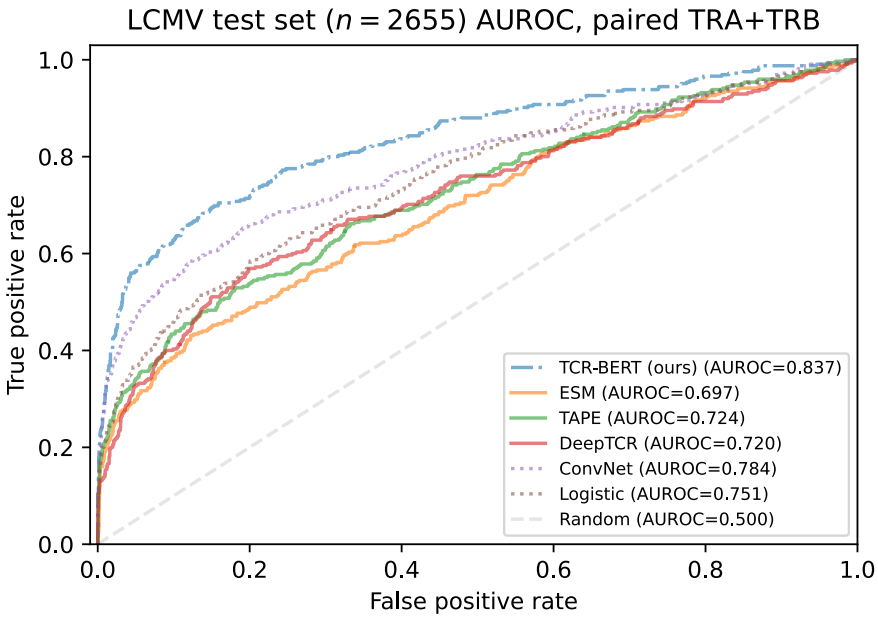


Supplementary Figure 3: Ablation of pre-training steps

We apply antigen cross-validation to evaluate the importance of each of TCR-BERT's two pre-training steps. We omit each pre-training step and evaluate the resulting model against the full TCR-BERT model with both pre-training steps, or against other protein language models like TAPE and ESM. As in Supplementary Figure 2, points are sized proportionally to the amount of training data available, and all p-values are computed using a two-sided Wilcoxon test.

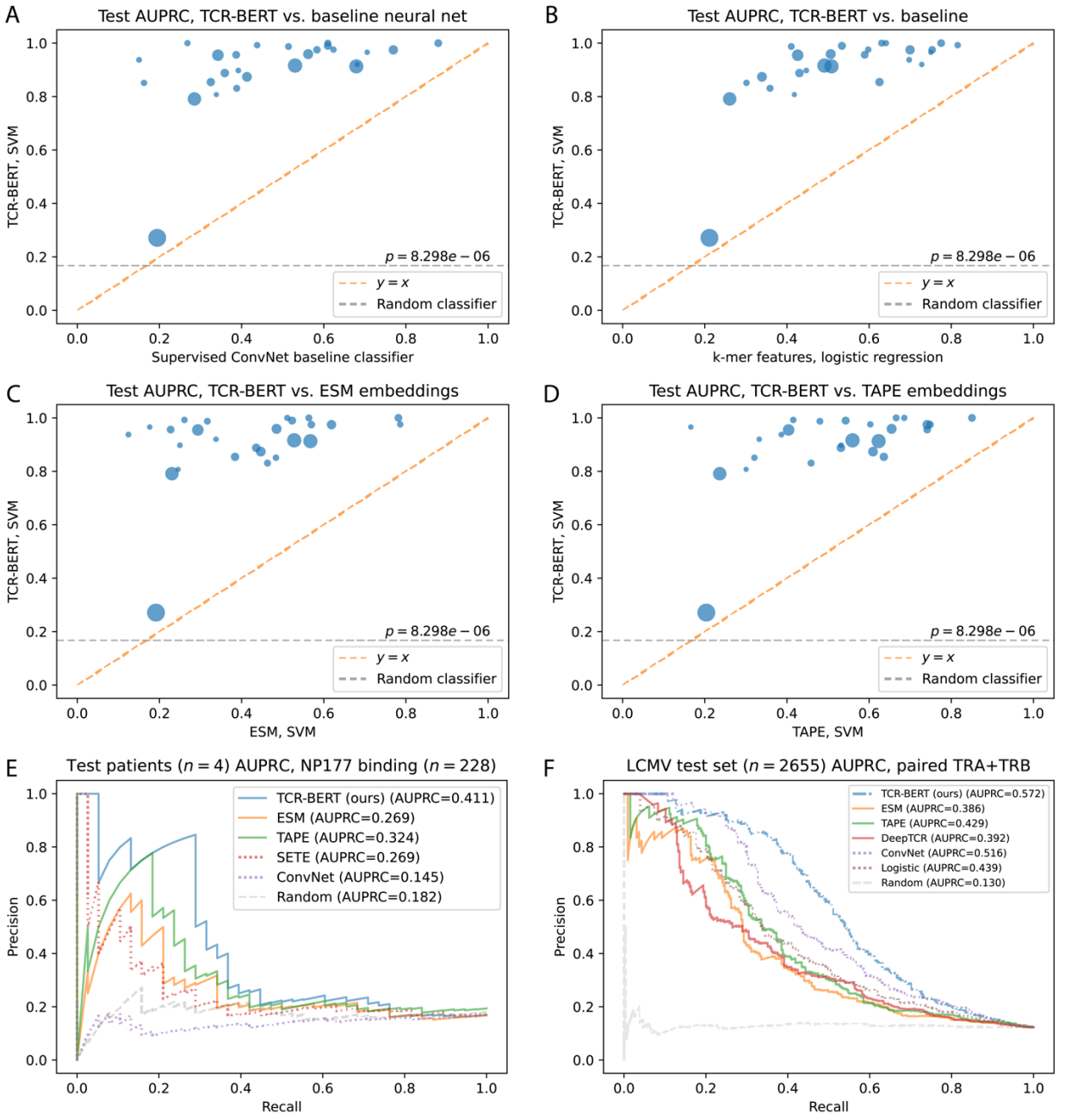
(A) We leave out the classification pre-training step (i.e., using MAA only, y-axis), comparing it to the TCR-BERT model with both pretraining steps (x-axis). Omitting classification pre-training results in a drop in performance across all cases.

(B) We similarly evaluate TCR-BERT's full pre-training (x-axis) against using only classification pre-training (i.e., no MAA, y-axis) and observe a significant overall decrease in performance.
(C) In addition to comparing to itself, we also compare TCR-BERT's training ablations to other protein language models. This panel compares TCR-BERT with only MAA pre-training against TAPE. While the performance difference between these is much less dramatic than in Supplementary Figure 2C, TCR-BERT retains a significant performance improvement.
(D) Comparing TCR-BERT with only classification pre-training against TAPE, we find that TCR-BERT retains a significant performance lead. This suggests that TCR-BERT's second pre-training task is more powerful in learning an embedding for TCRs, despite using relatively few training examples.
(E) We find that TCR-BERT with only MAA pre-training significantly outperforms ESM. However, the improvement is much less drastic compared to Figure 2B. TCR-BERT's retained performance improvement here, along with panel C, suggests that its TCR-specific MAA pre-training vocabulary lends some performance advantages.
(F) As with TAPE, TCR-BERT with only its second pre-training task active outperforms ESM. TCR-BERT's performance gain with only classification pre-training (see also panel D) appears larger than its performance gain with only MAA pre-training (panels C, E). Altogether, these results suggest that while both of TCR-BERT's pre-training tasks are important for its high performance, the second classification pre-training is more impactful.



Supplementary Figure 4: LCMV test set AUROC

AUROC curves describing various classifiers' test performance predicting GP33 binding given paired TRA/TRB sequences. TCR-BERT provides the best performance when evaluated on AUROC (shown here) as well on AUPRC, our primary evaluation metric (Figure 2D). Solid line indicates models leveraging pre-training, dashed lines indicate models that use only supervised training, and TCR-BERT's dot-dash line indicates that is fine-tuned from a pre-trained model.

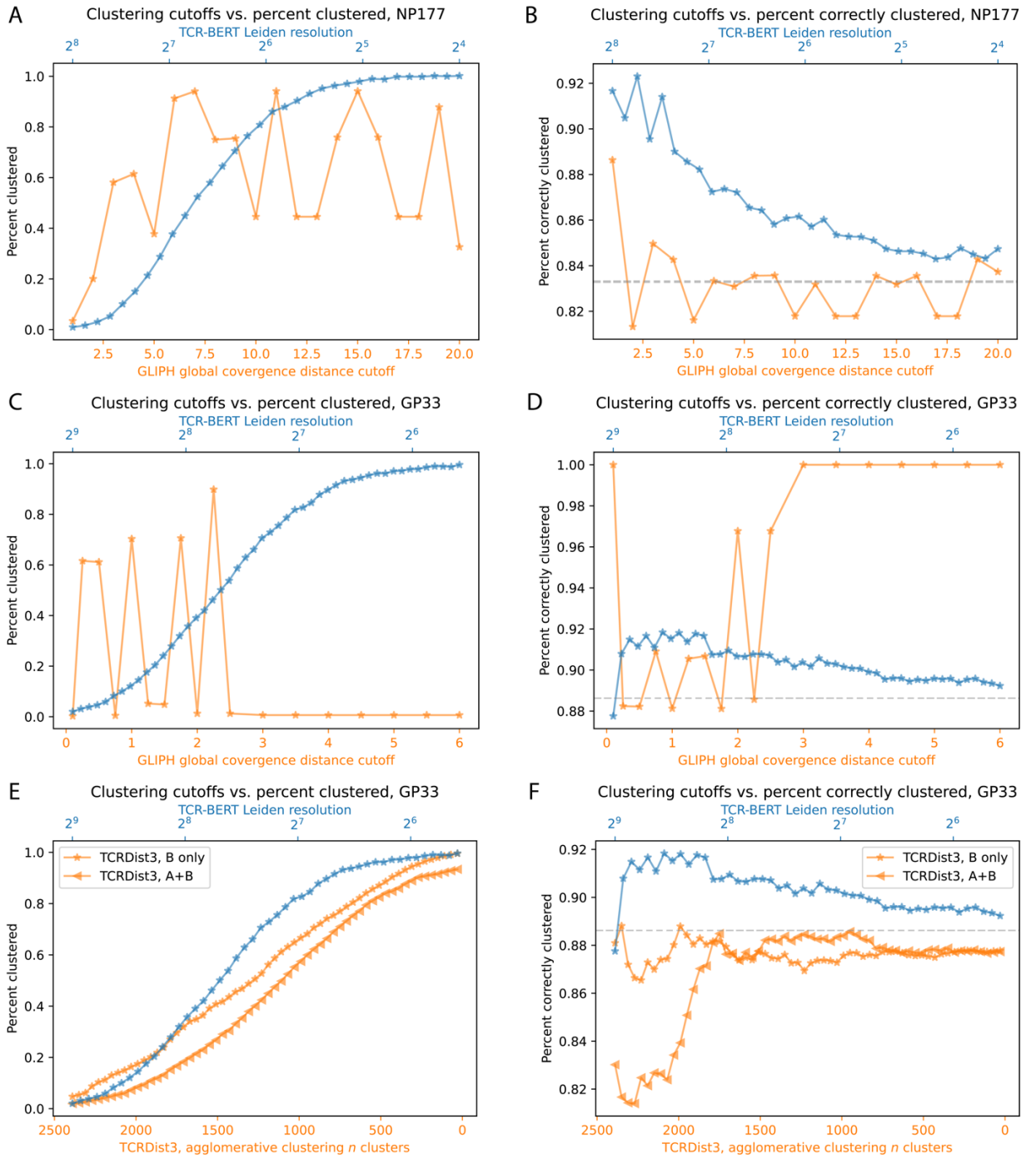


Supplementary Figure 5: Antigen classification with reduced train/test similarity

We repeat our range of antigen classification tasks with the additional constraint that no training sequence should be within 1 Levenshtein edit distance from any test sequence. These highly similar training sequences are excluded to reduce "bleed" of similar sequences between training and test (there were never exact sequences repeated between training and test). The test set in each case, on which we report metrics, is identical to prior tests. In panels A-D, TCR-BERT's performance is shown on the y-axis, and annotated p-values are derived from a two-sided Wilcoxon test.

(A) Compares TCR-BERT's AUPRC to that of a supervised convolutional neural network (x-axis) across antigen cross-validation. As in Figure 2A, we see that TCR-BERT unanimously outperforms this baseline method.

(B) We also compare TCR-BERT to logistic regression trained on k-mer features (x-axis) across antigen cross-validation. As in Supplementary Figure 2A, TCR-BERT significantly outperforms this baseline.
(C) In addition to the two supervised baselines, we also evaluate TCR-BERT against more general-purpose protein transformers like ESM. As in Figure 2B, which shows this same test without the edit distance constraint, TCR-BERT exhibits significantly improved performance.
(D) The same is the same for the TAPE general purpose protein transformer (compare to Supplementary Figure 2C).
(E) In Figure 2C, we evaluate patient-based data splits for predicting NP177 binding. Here, we repeat this test, additionally removing any training patient TRB sequences that are within 1 edit of any of the test patients' sequences. This is an artificially challenging test, as there is no reason to purposely restrict our training set based on new patients' data. Nonetheless, TCR-BERT still provides the strongest performance.
(F) In Figure 2D, we evaluate murine GP33 binding given TRA/TRB pairs. Here, we repeat this evaluation, removing training pairs with a total edit distance of 2 or fewer (summed across TRA and TRB) or less to any test sequence. In other words, all train sequence pairs are at least 3 total edits to any test pair. As in Figure 2D, dotted lines indicate supervised methods, solid lines indicate pre-trained models, and TCR-BERT's dot-dash line indicates it is fine-tuned from a pre-trained model. As before, TCR-BERT with fine-tuning provides the best overall performance.



Supplementary Figure 6: Detailed comparison of GLIPH, TCRDist3, and TCR-BERT antigen specificity groupings

Both GLIPH, TCRDist3, and TCR-BERT produce TRB groupings with configurable degrees of granularity. This is controlled using the resolution parameter to the Leiden clustering algorithm applied to TCR-BERT's embedding (larger values are more granular), the global convergence distance cutoff for GLIPH (smaller values are more granular), and the number of clusters in agglomerative clustering applied to TCRDist3's pairwise distances (larger values are more granular). We compare the performance of these methods for the human NP177 antigen (A, B) and murine LCMV GP33 antigen (C-F). TCRDist3 is only evaluated on the

latter LCMV dataset as it requires V/J gene annotations, which are not available for the NP177 dataset. Generally, we expect some variability in percent correctly clustered for all methods (B, D, F) at high granularity configurations, as these configurations tend to produce few clusters, which amplifies the impact of any mistakes.

(A) Shows percent clustered for a mixture of human NP177 antigen for TCR-BERT (blue) and GLIPH (orange). TCR-BERT converges to having all sequences clustered as the clustering becomes less stringent, while GLIPH produces highly variable, unpredictable results.

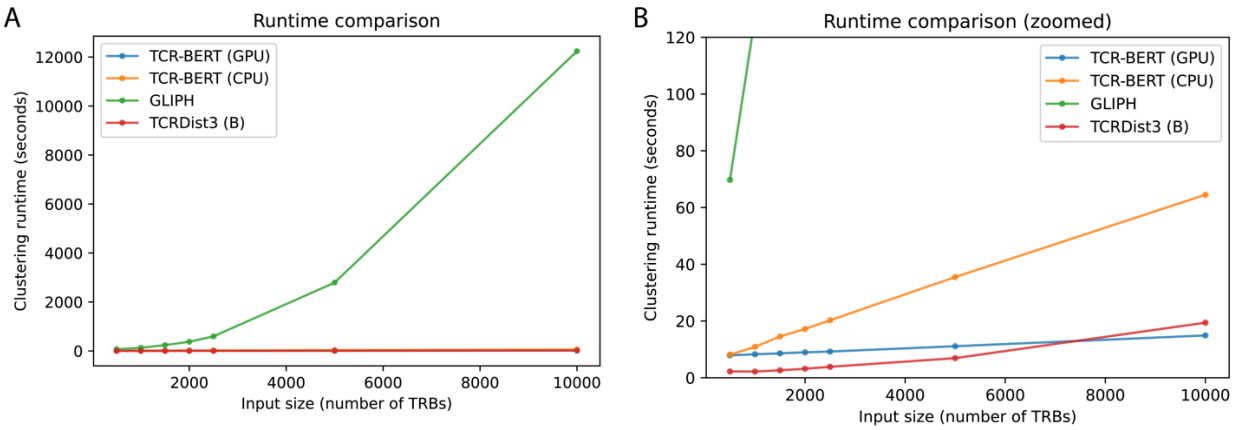
(B) Shows the percent correctly clustered for these two methods, again for the NP177 antigen. Both methods trend towards lower percent correctly clustered as clustering granularity decreases. However, TCR-BERT's clustering stabilizes above the expected correctness of a random classifier, whereas GLIPH's behavior is much less consistent, frequently dipping below random.

(C) Shows percent clustered for GLIPH and TCR-BERT on the murine LCMV GP33 dataset. With TCR-BERT, looser clustering configurations allow for clustering of the entire dataset (blue) as it does for the NP177 dataset (panel A). GLIPH on the other hand, produces no consistent relationship between its configuration and percentage clustered (orange). Indeed, GLIPH struggles to consistently cluster any meaningful proportion of the given TRBs. This reveals that GLIPH not only exhibits erratic behavior across granularity configurations, but also behaves inconsistently when evaluated on different datasets.

(D) Shows percent correctly clustered for these GLIPH and TCR-BERT on the LCMV GP33 dataset. As before, TCR-BERT produces more predictable performance (blue). While it might appear that GLIPH (orange) achieves perfect accuracy towards higher cutoffs, this is at the cost of extremely few sequences being clustered (panel C), which vastly reduces the usefulness of the small handful of correctly clustered sequences. Increasing GLIPH's global convergence distance cutoff seems to increase the percent correctly clustered in this case, which is contrary to the behavior observed for the NP177 human dataset (panel B). TCR-BERT's behavior, on the other hand, is consistent across parameters for both these datasets. Overall, these results show that TCR-BERT exhibits consistent, predictable, and improved clustering behavior compared to GLIPH.

(E) Compares percent clustered for TCR-BERT (blue) and TCRDist3 (orange). TCRDist3 is evaluated with the TRB chain only (* markers), and with full TRA/TRB pairs (◀️ markers). Both methods are consistent in their behavior, where less granular configurations consistently produce higher percent clustered.

(F) Similarly compares percent correctly clustered for TCR-BERT (blue) and TCRDist3 (orange, * and ◀️ markers). Compared to TCR-BERT, TCRDist3 produces lower correctness, particularly when run using TRA/TRB pairs (◀️ markers). TCRDist3 also converges to an overall correctness that is worse than random, whereas TCR-BERT converges above random.

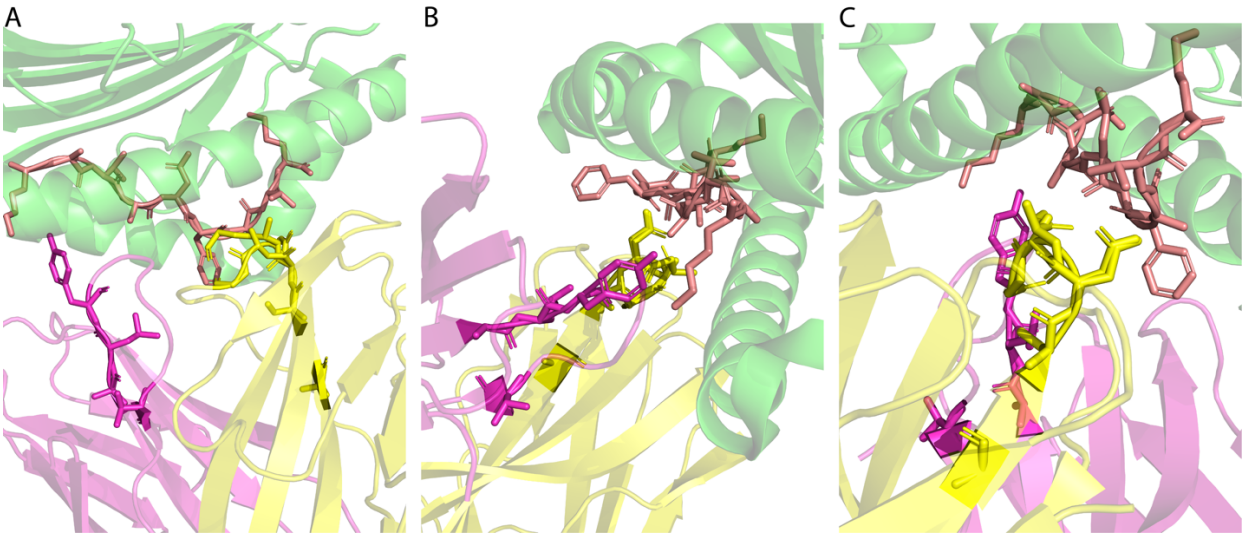


Supplementary Figure 7: Runtime comparison for methods generating groupings for antigen specificity

Comparison of elapsed time, in seconds, required to cluster TRB sequences with predicted shared antigen specificity. All methods are benchmarked on the same LCMV GP33 murine TRB sequences, subsetted to various input sizes (x-axis), and are run on the same machine. TCR-BERT can be run with GPU acceleration or using only the CPU; both cases exhibit similar runtime complexity. GLIPH and TCRDist3 not support GPU acceleration.

(A) Shows runtime performance as we scale from 500 to 10,000 input sequences. GLIPH's runtime scales super-linearly, dwarfing both TCR-BERT and TCRDist3. For example, increasing the number of input sequences by 4x from 2500 to 10000 results in a ~21x increase in runtime from about 10 minutes to over 3 hours. This makes running GLIPH prohibitively time-consuming on larger datasets, especially when evaluating multiple parameters/configurations.

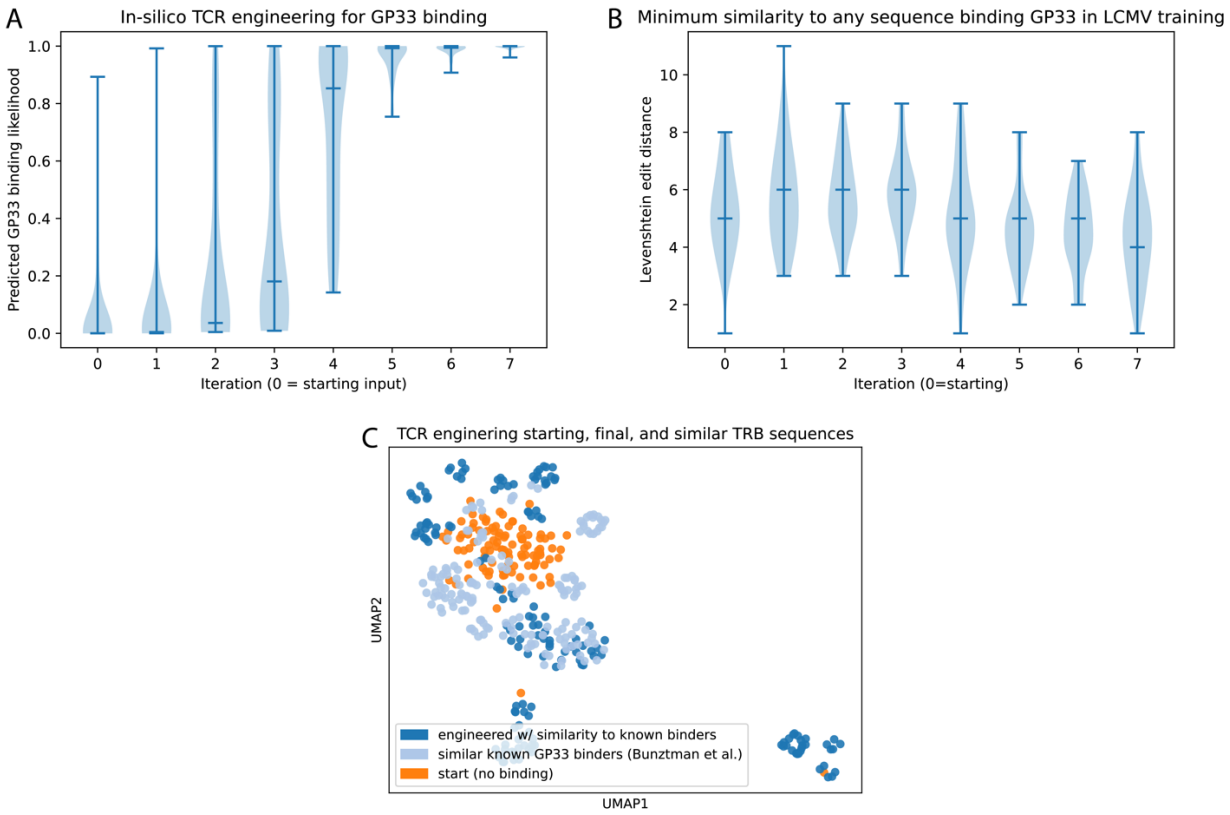
(B) Truncates the y-axis to lower values to clearly show runtime characteristics for TCR-BERT and TCRDist3. Even without GPU acceleration, TCR-BERT can process 10,000 inputs in less time than it takes GLIPH to process just 500. TCRDist3 runs extremely quickly, particularly on small inputs, but as it is designed to perform pairwise sequence comparisons, it necessarily exhibits super-linear, quadratic runtime complexity. We can see this behavior as TCRDist3 becomes slower than TCR-BERT (GPU) on the largest benchmarked input of 10,000 sequences, despite being faster in all smaller inputs. Irrespective of CPU or GPU hardware, TCR-BERT exhibits linear runtime behavior, making it more scalable for analyzing larger and larger datasets.



Supplementary Figure 8: Additional views of 5m00 with high-attention TRA/TRB residues highlighted.

3D structures showing the MHC (green), modified GP33 antigen (salmon), and TRA (pink) and TRB (yellow). GP33 antigen side chains are shown, as well as side chains for TRA and TRB residues with TCR-BERT attentions in the top 33rd percentile of attention values in each chain. Other residues are shown in faded cartoon-ribbon illustrations without side chains. Attention values are derived from average attentions across test set sequences with identical lengths as sequences profiled in PDB structure 5m00. In all views, the TRA/TRB residues with the greatest attention are frequently in direct contact with the antigen peptide's side chains.

(A) Flipped view of Figure 4D
(B) View with the TRA in the foreground
(C) View with the TRB in the foreground



Supplementary Figure 9: TCR engineering supplement

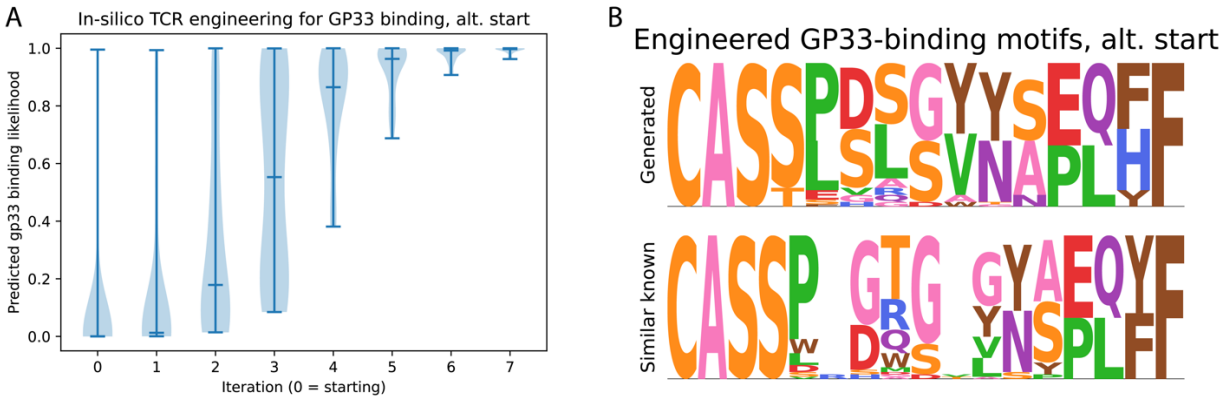
(A) Our algorithm for engineering TCR sequences successfully increases the predicted binding of sequences (y-axis) with each iteration (x-axis). Violin plot indicates full range of values, with medians marked. The 0-th iteration represents the input sequences. We stop our engineering iterations (Figure 5A) when all sequences exceed 95% predicted binding probability, which occurs after 7 iterations.

(B) Our algorithm for engineering TCR sequences for antigen binding works without simply regurgitating training sequences. We show this by computing the Levenshtein edit distance (larger values indicate greater dissimilarity, y-axis) between each generated TRA/TRB sequence pair and the most similar sequence pair in our training dataset of GP33-binding TCRs. This is expressed as a violin plot for each iteration of our procedure (x-axis) spanning the full range of values with medians marked. At no point during the TCR engineering process do we produce a sequence directly seen in training (i.e., no instances of 0 edit distance). This indicates that we are generating truly novel sequences that TCR-BERT has not seen before.

(C) We can also visualize the progression from starting to engineered sequences in TCR-BERT's embedding space. Here, we embed each TRB sequence (using the same variant of TCR-BERT pre-trained on MAA and antigen classification as we use for other embedding visualizations) and visualize the embeddings using UMAP. Colors correspond to starting, engineered and similar binders (i.e., TRBs identified in a separate experiment to also bind GP33 that bear significant similarity to our final engineered set). Known GP33 binders (light blue) lie towards the "outskirts" of our starting set (orange), and our engineered set pull outwards toward these (dark blue). This shows that our TCR engineering process explores the landscape of TCR sequences without being confined to the area spanned by input sequences.

Supplementary Figure 10: Detailed sequence comparisons for 11 novel matches

Each of these 11 matches was not present in the initial pool of matches corresponding to the starting set of sequences for TCR engineering. For each match (bottom panels), we show the generated TRB with the highest bit score/lowest E-value to that match (top panels). Each pairing is annotated with the E-value corresponding to that match.



Supplementary Figure 11: Additional TCR engineering results

(A) We repeat TCR engineering with a different starting set of negative sequences, also drawn from the pool of LCMV test set negatives. As before, our TCR engineering procedure successfully creates sequences with increasingly greater predicted binding with each iteration. Violin plot indicates full range of values, with medians marked.

(B) We match the final set of engineered TRBs to known murine TRBs using BLAST (E-value ≤ 0.001). Among the significant hits, 61/189 correspond to GP33-binding TRB sequences. The bottom motif corresponds to these hits, and the top motif corresponds to our corresponding generated sequences. For context, proportionally fewer (69/393) matches for the starting input set corresponded to GP33-binding TRBs. As before, TCR-BERT proportionally enriches for GP33 binders ($p = 1.17 \times 10^{-4}$, Fisher's exact test).