Appendix for "Disentangled Wasserstein Autoencoder for Protein Engineering"

Anonymous Author(s)
Affiliation
Address
email

1 Data preparation

1.1 Combination of data sources

TCR-peptide interaction data are obtained from VDJDB [1] and MCPAS [2]. Only peptides with > 10 observed pairs are used for downstream filtering. Because VDJDB and MCPAS only report interacting pairs, we first combine the dataset with the training set from NETTCR [3] which contains experimentally-validated non-interacting pairs. Conflicting records are removed.

1.2 Filtering by ERGO performance

Since ERGO [4] trains two separate models for VDJDB and MCPAS, the following filtering process is also performed separately on the two datasets. For this and all subsequent ERGO-based predictions, we use the pre-trained weights from `https://github.com/louzounlab/ERGO`.

Additional negative samples are generated as follows: a random TCR sequence is first selected from the dataset and is paired with all existing peptides in the dataset. Any unobserved pair is treated as negative. We repeat this process until the size of the negative set is 5x that of the positive set. The expanded dataset is then provided to the respective ERGO model. Predictive performance is evaluated for each peptide. We keep the peptides with AUROC and AUPR > 0.9 and select those among top 10 positive sample counts (Table 1).

To ensure the specificity of TCR recognition in the following study, we did a second round of filtering of both the TCRs and the peptides. We pair all TCRs with at least one positive binding event and all peptides in the filtered dataset. Any unobserved pair is treated as negative. This dataset is then provided to ERGO. Performance is shown in Table S2. We discard peptides with AUPR < 0.7 and TCRs that have more than one positive prediction or have at least one wrong prediction.

After that, we downsample all peptides to at most 400 positive TCRs. This number is chosen so that the resulting dataset is more balanced across peptides. The final number of samples for each peptide can be found in S3. To make sure the model captures peptide-specific information, for every TCR in the positive set, we add its unobserved pairings with other peptides to the negative set. We then split the TCRs into train/test/validation sets with a ratio of 8:1:1, and put all pairings of each TCR to the respective subset, to ensure all TCRs in the test and validation sets are not seen in the training. For the training set, the positive samples are up-sampled by the negative/positive ratio of the original dataset.

Notation	Meaning
Θ_f	functional encoder
Θ_s	structural encoder
Γ	decoder
Ψ	auxiliary functional classifier
$\{\mathbf{x}, \mathbf{u}, y\}$	a data point with TCR $\mathbf{x}$, peptide $\mathbf{u}$ and binding label y
$\mathbf{z}_f$	functional embedding
$\mathbf{z}_s$	structural embedding
$\mathbf{z}$	concatenation of $\{\mathbf{z}_f, \mathbf{z}_s\}$
$\mathbf{x}'$	reconstructed/generated sequence from the decoder
$\mathbf{x}^{(i)}$	the probability distribution over amino acids at the i-th position in $\mathbf{x}$
concat$(\mathbf{x}_1, ..., \mathbf{x}_n)$	concatenation of vectors $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$

Table. S1: Notations used for this paper. Sequences are represented as $l \times |V|$ matrices where l is the length $|V|$ is the number of amino acids.

2 Model details

2.1 Proof of Theorem 1

We use density functions for simplicity. Let $q_\theta(\mathbf{z} \mid \mathbf{x})$ be the encoder and $p_\gamma(\mathbf{x} \mid \mathbf{z})$ be the decoder. We have the joint generative distribution:

$$p(\mathbf{x}, \mathbf{z}) = p_\gamma(\mathbf{x} \mid \mathbf{z})p(\mathbf{z}),$$

where $p(\mathbf{z})$ is the prior. Also, we have the joint inference distribution:

$$q(\mathbf{x}, \mathbf{z}) = q_\theta(\mathbf{z} \mid \mathbf{x})p_D(\mathbf{x}),$$

where $p_D(\mathbf{x})$ is the data distribution.

$$\begin{aligned} I(\mathbf{X}; \mathbf{Z}) &= \mathbb{E}_{q(\mathbf{x},\mathbf{z})} \log \frac{q(\mathbf{x}, \mathbf{z})}{p_D(\mathbf{x})q(\mathbf{z})} \\ &= \mathbb{E}_{p_D} \sum_{\mathbf{z}} p_D(\mathbf{x}) q_\theta(\mathbf{z} \mid \mathbf{x}) \log \frac{q_\theta(\mathbf{z} \mid \mathbf{x}) p_D(\mathbf{x})}{p_D(\mathbf{x})q(\mathbf{z})} \\ &= \mathbb{E}_{p_D} \sum_{\mathbf{z}} q_\theta(\mathbf{z} \mid \mathbf{x}) \log \frac{q_\theta(\mathbf{z} \mid \mathbf{x})}{q(\mathbf{z})} \\ &= \mathbb{E}_{p_D} \sum_{\mathbf{z}} q_\theta(\mathbf{z} \mid \mathbf{x}) \log \frac{q_\theta(\mathbf{z} \mid \mathbf{x})}{p(\mathbf{z})} - \mathbb{E}_{p_D} \sum_{\mathbf{z}} q_\theta(\mathbf{z} \mid \mathbf{x}) \log \frac{q(\mathbf{z})}{p(\mathbf{z})} \\ &= \mathbb{E}_{p_D} \sum_{\mathbf{z}} q_\theta(\mathbf{z} \mid \mathbf{x}) \log \frac{q_\theta(\mathbf{z} \mid \mathbf{x})}{p(\mathbf{z})} - \sum_{\mathbf{z}} q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z})} \\ &= \mathbb{E}_{p_D} [\mathbb{D}_{\text{KL}}(Q_\theta(\mathbf{Z} \mid \mathbf{X}) \parallel P(\mathbf{Z}))] - \mathbb{D}_{\text{KL}}(Q(\mathbf{Z}) \parallel P(\mathbf{Z})). \end{aligned}$$

Thus,

$$\mathbb{D}_{\text{KL}}(Q(\mathbf{Z}) \parallel P(\mathbf{Z})) = \mathbb{E}_{p_D} [\mathbb{D}_{\text{KL}}(Q_\theta(\mathbf{Z} \mid \mathbf{X}) \parallel P(\mathbf{Z}))] - I(\mathbf{X}; \mathbf{Z}).$$

2.2 Implementation and training details

All input sequences are padded to the same length (25). The peptide $\mathbf{u}$ is represented as the average BLOSUM50 score [5] for all its amino acids. The model is trained from end to end using the Adam optimizer [6]. The first layer of the model is an embedding layer that transforms the one-hot encoded sequence $\mathbf{x}$ into continuous-valued vectors of 128 dimensions:

$$\mathbf{e} = W^{emb}\mathbf{x}.$$

Both $\mathbf{z}_f$ and $\mathbf{z}_s$ encoders are 1-layer transformer encoders with 8 attention heads and an intermediate size of 128. The transformer layer utilizes the multi-head self-attention mechanism. For each attention head i:

$$Q_i = W_i^Q \mathbf{e}, K_i = W_i^K \mathbf{e}, V_i = W_i^V \mathbf{e}$$

$$\text{Attn}_i(\mathbf{e}) = \text{softmax}(\frac{Q_i K_i^T}{\sqrt{d_k}})V_i,$$

where d_k is the dimension of Q_i and K_i. The outputs of the attention heads are then aggregated as follows:

$$\text{Multihead}(\mathbf{e}) = \text{concat}(\text{Attn}_1(\mathbf{e}), \text{Attn}_2(\mathbf{e}), ...)W^O.$$

A 2-layer MLP with a 128-dimension hidden layer is then built on top of the transformer (which has the same dimension as the input embeddings) to transform the output to the dimensions of $\mathbf{z}_f$ and $\mathbf{z}_s$, respectively. The functional classifier is a 2-layer MLP with a 32-dimension hidden layer. The decoder is a 2-layer LSTM with 256 hidden dimensions.

The hyperparameters are selected with grid search and models with the best generation results are reported. Specifically, weights of all losses are selected from $[1.0, 10.0]$ and learning rate (lr) are selected from $[1e-4, 1e-5]$. The dimension of $\mathbf{z}_f$ is fixed to 8 and $\mathbf{z}_s$ to 32. We train each model with 200 epochs and evaluate the checkpoint of every 50 epochs. We find the variance of the RBF kernel (for the calculation of the Wasserstein loss) does not have a strong impact on the results significantly, so the value is fixed to 1.0.

The model is trained with four different random seeds (42, 456, 789, 987). We report the hyperparameter setting with the best average performance (i.e. one that generates the highest average number of qualified positive sequences for the well-classified peptides).

The hyperparameter setting of the models for comparison and visualization are:

$$[\beta_1 = 1.0, \beta_2 = 0.1, \text{lr} = 1e-4, \text{epoch} = 200]$$

where β's are weights of the losses:

$$\mathcal{L} = \mathcal{L}_{recon} + \beta_1 \mathcal{L}_{f\_cls} + \beta_2 \mathcal{L}_{\textit{Wass}}.$$

For the visualization and analysis of the model trained on VDJDB, we use random seed $= 789$.

We use the scheduled sampling technique [7] for the LSTM decoder during training, where for each position in the input sequence, there is a 0.5 probability of using the previous predicted token, instead of the original token, to calculate the hidden state for the next position. This is employed to avoid the discrepancy between the training and the generation, as the former uses the original sequence to calculate the hidden states and the latter uses predicted tokens.

The model is trained on 2 rtx3090 GPUs with a batch size of 256 (128 per GPU). Training with 200 epochs typically takes ~ 4 hours.

3 Baseline methods

We compare our model with two types of methods for the generation of the optimized TCR $\mathbf{x}'$: (1) mutation-based, which iteratively adds random mutations to the template sequence; and (2) generation-based, which generates novel sequences of the pre-determined length range. For both types of methods, the modified/generated sequences are selected by peptide binding scores from the respective pre-trained ERGO. The experiments are performed on each peptide in the dataset independently.

3.1 Mutation-based baselines

Random mutation (`naive rm`) The TCR is randomly mutated by one amino acid for 8 times progressively. This process is repeated for 10 runs for each TCR and the resulting one with the highest ERGO prediction score is reported.

Greedy mutation (`greedy`) For each TCR, 10 randomly mutated sequences are generated, each with one amino acid difference from the original sequence. Among the 10 mutated sequences, we select the one that gives the highest binding prediction with the given peptide as the template for the next run. This process is repeated 8 times.

Genetic algorithm (`genetic`) Let M be the sample size. For each TCR, 10 randomly mutated sequences are generated, each with one amino acid difference from the original sequence. All mutated sequences along with the original TCRs are then pooled together, and the top M sequences that give the highest binding prediction are used as the input for the next run. This process is repeated 8 times.

3.2 Generation-based baselines

Monte Carlo tree search (`MCTS`) TCRs are generated by adding amino acids iteratively, resulting in a search tree. When TCR length reaches 10, the binding score is estimated by ERGO. For each iteration, a random node is selected for the expansion and evaluated by ERGO, and the scores of all its parent nodes are updated accordingly. The tree expansion ends when the length reaches 20. For every generation process, the highest leaf node is added into the output TCR set.

3.3 IDEL

IDEL [8] is a VAE with a mutual information constraint on the latent space. For training, the loss comprises of the following components:

- The reconstruction loss: $\mathcal{L}_{recon}(\mathbf{x}, \mathbf{x}')$
- The KL divergence term for VAE: $\mathcal{L}_{KL} = \mathbb{D}_{\text{KL}}(q_\theta(\mathbf{z}_s, \mathbf{z}_f|\mathbf{x}) \parallel p(\mathbf{z}_s, \mathbf{z}_f))$.
- The reconstruction loss given $\mathbf{z}_s$: $\mathcal{L}_s(\mathbf{x}, \Phi(\mathbf{z}_s))$ where Φ is an auxiliary decoder.
- The classification loss given z_f: $\mathcal{L}_{f\_cls}(\hat{y}, y)$
- The sample-based MI upper bound between the embeddings: $\mathcal{L}_{MI}(\mathbf{z}_f, \mathbf{z}_s)$. This requires an approximation of the conditional distribution $p(\mathbf{z}_f|\mathbf{z}_s)$, which is achieved by a separate neural network.

Here we use our own notations, not the ones used in the original paper, for better comparison.

We performed a grid search from $[1.0, 10.0]$ for the weight of the loss terms and $[1e-4, 1e-5]$ for the learning rate. The model giving the best performance has 10.0 weight for $\mathcal{L}_{recon}$, $\mathcal{L}_s$ and $\mathcal{L}_{f\_cls}$, 1.0 for the other terms, and a learning rate of $1e-4$. In practice, we performed annealing [9] on $\mathcal{L}_{KL}$ and $\mathcal{L}_{MI}$ where their weights gradually increase through training, to make sure the embeddings are as informative as possible.

4 Evaluation of the optimized sequences

4.1 Training of the autoencoder

We train an LSTM-based autoencoder, which we denote as TCR-AE0, on the 277 million TCR sequences from TCRdb [10]. TCR-AE0 has a latent space of dimension 16 and is trained for 50,000 steps with a batch size of 256.

4.2 Validity score

The validity score combines two scores calculated from TCR-AE0:

- The reconstruction-based score is calculated as
$$r_r(\mathbf{x}') = 1 - \text{lev}(\mathbf{x}', \text{TCR-AE0}(\mathbf{x}'))/l(\mathbf{x}'),$$
where $\text{lev}(\mathbf{x}', \text{TCR-AE0}(\mathbf{x}'))$ is the Levenstein distance between the original sequence and the reconstructed sequence, and $l(\mathbf{x}')$ is the length of the reconstructed sequence. Higher r_r means $\mathbf{x}'$ is better reconstructed from TCR-AE0 and is thus more likely to be a valid TCR sequence.
- The density-based score calculates whether the embedding of $\mathbf{x}'$ follows the same distribution as known TCRs. We learn a Gaussian mixture model from the latent embeddings of known TCRs from TCR-AE0. The likelihood of the embedding $\mathbf{e}'$ of $\mathbf{x}'$ from TCR-AE0 falling

in the same Gaussian mixture distribution is denoted as $P(\mathbf{e}')$. The density-based score is calculated as

$$r_d(\mathbf{x}') = \exp(1 + \frac{\log P(\mathbf{e}')}{\tau}),$$

where $\tau = 10$. Higher r_d means the latent embedding of $\mathbf{x}'$ from TCR-AE0 is more likely to follow the same distribution as other valid TCR sequences.

We then define the validity score as $r_v = r_r + r_d$.

4.3 Validation of the metrics

We compare the TCR-AE-derived evaluation metric scores of three different sources:

(1) all unique CDR3β sequences from VDJDB.

(2) random segments of length $8 - 18$ (which is the most frequent lengths of CDR3β sequences) from random uniport [11] protein sequences of the same size as (1). The conservative 'C' at the beginning and 'F' at the end are added to the segments.

(3) random shuffling of the sequences from (1), where the first 'C' and the last 'F' are kept

We show in Fig. S1 that for both two scores r_d and r_r, as well as their sum, CDR3β sequences score much higher than random proteins or shuffled sequences. This shows these scores could be effectively used for the estimation of TCR sequence validity. We choose $r_v > 1.25$ as the criteria for valid sequences as it rejects most negatives.

5 Extended Results

5.1 Comparison of TCR Engineering Performance

We find consistently improved performance of our method over the baselines in both VDJDB (Table 1) and McPAS-TCR (Table S4). Also, the majority of generated sequences are unique (Table S5) and all are not observed in the original dataset (not shown).

5.2 Analysis of the Model

We show extended $\mathbf{z}_f$ and $\mathbf{z}_s$ T-SNE patterns in Fig. S2, colored by the ground truth label as well as the predicted label. For the well-classified peptides, there is a clear separation of positives and negatives in the $\mathbf{z}_f$ space but not $\mathbf{z}_s$. There are cases where the true positives are not separable from the true negatives using $\mathbf{z}_f$, but the predicted positives and the predicted negatives (by the function classifier Ψ) are still separated. We consider the latter as a problem with data quality and classification accuracy, not embedding. Meanwhile, the classifier shows consistent performance over the peptides across random seeds with (Fig. S3, left) and without (Fig. S3, right) the Wasserstein loss.

As a result of the Wasserstein loss, the distribution of the embedding space is closer to a multivariate Gaussian (Fig. S4A. It becomes less regularized without the Wasserstein loss (Fig. S4B). Contrary to $\mathbf{z}_f$ (Fig. 3B in the main text), the T-SNE of $\mathbf{z}_s$ and first-layer embedding of the encoder for the positive samples cannot distinguish the binding targets from each other (Fig. S5A).

5.3 Analysis of the Generated Sequences

In addition to the results presented in the main text, we also selected 500 random positive and negative sequences from the training set and replaced their $\mathbf{z}_f$ with the most positive/negative one in the subset. The generated sequences using their original $\mathbf{z}_s$ and the new $\mathbf{z}_f$ have binding scores mostly related to the $\mathbf{z}_f$, regardless of whether the $\mathbf{z}_s$ source is positive or negative. This shows $\mathbf{z}_f$ can be used to encode and transfer binding information, which lays the foundation for the following TCR engineering experiments (Fig. S5B).

The generated sequences have a similar length distribution as their templates (Fig. S5C), meaning no drastic changes are made. We further find that the $\mathbf{z}_s$ of the modified TCRs show high cosine similarity with those of their templates, while the $\mathbf{z}_f$ are more similar to the $\mathbf{z}_f$ used for their generation (Fig.

S5D), but not with that of the template. These show that the modified TCRs preserve the "structural" information from $\mathbf{z}_s$ and incorporate the new "functional" information from the modified $\mathbf{z}_f$.

	source	#pos	auroc	aupr
AVFDRKSDAK	vdjdb	1641	0.94	0.71
CTPYDINQM	vdjdb	500	0.99	0.81
ELAGIGILTV	vdjdb	1410	0.95	0.79
FRDYVDRFYKTLRAEQASQE	vdjdb	367	0.98	0.85
GILGFVFTL	vdjdb	3408	0.95	0.89
GLCTLVAML	vdjdb	962	0.92	0.73
IVTDFSVIK	vdjdb	548	0.94	0.62
KRWIILGLNK	vdjdb	319	0.95	0.54
NLVPMVATV	vdjdb	4421	0.94	0.85
RAKFKQLL	vdjdb	830	0.94	0.75
SSLENFRAYV	vdjdb	322	0.99	0.57
SSYRRPVGI	vdjdb	337	0.99	0.81
STPESANL	vdjdb	234	0.99	0.35
TTPESANL	vdjdb	511	0.99	0.75
ASNENMETM	mcpas	265	0.98	0.63
CRVLCCYVL	mcpas	435	0.95	0.7
EAAGIGILTV	mcpas	272	0.97	0.55
FRCPRRFCF	mcpas	266	0.96	0.58
GILGFVFTL	mcpas	1142	0.96	0.9
GLCTLVAML	mcpas	828	0.95	0.85
LPRRSGAAGA	mcpas	2142	0.96	0.88
NLVPMVATV	mcpas	543	0.93	0.78
RFYKTLRAEQASQ	mcpas	304	0.99	0.91
SSLENFRAYV	mcpas	416	0.99	0.78
SSYRRPVGI	mcpas	337	0.99	0.83
TPRVTGGGAM	mcpas	274	0.95	0.52
VTEHDTLLY	mcpas	273	0.95	0.45
WEDLFCDESLSSPEPPSSSE	mcpas	364	0.98	0.93

Table. S2: Statistics and ERGO prediction performance for the selected peptides from the first round.

VDJDB			MCPAS		
	#pos	#all		#pos	#all
NLVPMVATV	2880	5478	NLVPMVATV	1792	3810
GLCTLVAML	2880	5478	RFYKTLRAEQASQ	1528	3579
RAKFKQLL	2880	5478	WEDLFCDESLSSPEPPSSSE	1928	3929
AVFDRKSDAK	2880	5478	GILGFVFTL	2560	4482
SSYRRPVGI	2268	4934	SSYRRPVGI	1504	3558
GILGFVFTL	2880	5478	SSLENFRAYV	1824	3838
TTPESANL	2286	4950	CRVLCCYVL	1680	3712
FRDYVDRFYKTLRAEQASQE	2034	4726	LPRRSGAAGA	2560	4482
ELAGIGILTV	2880	5478	GLCTLVAML	2560	4482
CTPYDINQM	2394	5046			

Table. S3: Statistics of the training data by peptide.

MCPAS

	$\bar{r_v}$	$\bar{r_b}$	%valid	# mutations	%positive valid
TCR-dWAE (best)	1.32±0.05	0.38±0.07	0.48±0.02	0.51±0.03	0.15±0.04
TCR-dWAE (avg)	1.4±0.07	0.31±0.03	0.59±0.03	0.44±0.03	0.15±0.02
TCR-dWAE (random)	1.38±0.07	0.29±0.03	0.68±0.07	0.47±0.03	0.16±0.01
IDEL (best)	1.42±0.01	0.34±0.07	0.42±0.11	0.43±0.03	0.11±0.02
IDEL (avg)	1.47±0.01	0.31±0.05	0.49±0.11	0.4±0.02	0.11±0.01
IDEL (random)	1.46±0.01	0.29±0.04	0.64±0.04	0.42±0.02	0.15±0.01
greedy	0.33±0.0	0.92±0.0	0.02±0.0	0.34±0.0	0.02±0.0
genetic	0.34±0.03	1.0±0.0	0.02±0.0	0.96±0.08	0.02±0.0
naive rm	0.31±0.0	0.43±0.0	0.02±0.0	0.35±0.01	0.01±0.0
mcts	-0.11±0.0	0.94±0.0	0.0±0.0	0.04±0.08	0.0±0.0
TCR-dWAE (null)	1.45±0.06	0.08±0.0	0.79±0.05	0.41±0.03	0.06±0.01

Table. S4: ; Performance comparison for MCPAS, averaged across selected peptides (SSYRRPVGI, WEDLFCDESLSSPEPPSSSE, SSLENFRAYV, RFYKTLRAEQASQ, GLCTLVAML, CRVLCCYVL)

	VDJDB		MCPAS	
	valid:all	unique:valid	valid:all	unique:valid
TCR-dWAE-best	0.59±0.02	0.69±0.1	0.67±0.06	0.72±0.05
TCR-dWAE-avg	0.63±0.03	0.74±0.09	0.74±0.07	0.8±0.06
TCR-dWAE-random	0.66±0.02	0.9±0.02	0.72±0.07	0.95±0.01
TCR-dWAE-null	0.86±0.01	0.99±0.0	0.8±0.05	0.99±0.0
IDEL-best	0.73±0.02	0.73±0.15	0.76±0.0	0.56±0.14
IDEL-avg	0.78±0.02	0.76±0.14	0.81±0.01	0.61±0.14
IDEL-random	0.78±0.02	0.83±0.07	0.8±0.01	0.81±0.06
greedy	0.02±0.0	1.0±0.0	0.02±0.0	1.0±0.0
genetic	0.03±0.02	0.74±0.04	0.03±0.0	0.71±0.08
naive rm	0.03±0.0	1.0±0.0	0.02±0.0	1.0±0.0
mcts	0.0±0.0	0.0±0.0	0.0±0.0	0.04±0.08

Table. S5: Additional performance comparison. This table shows the ratio of valid sequences and unique valid sequences, as well as the running time.

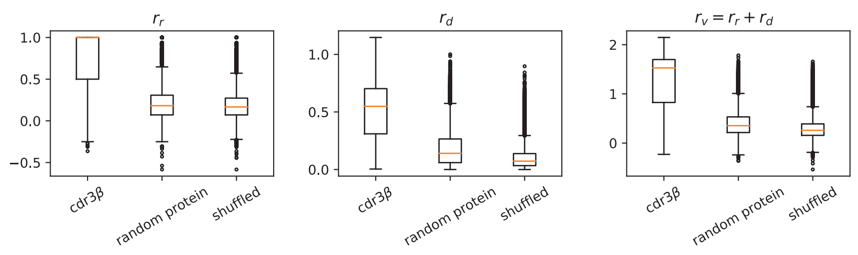

Fig. S1: Distribution of TCR-AE-based evaluation metrics on known CDR3β's, randomly selected protein segments and randomly shuffled CDR3β's.

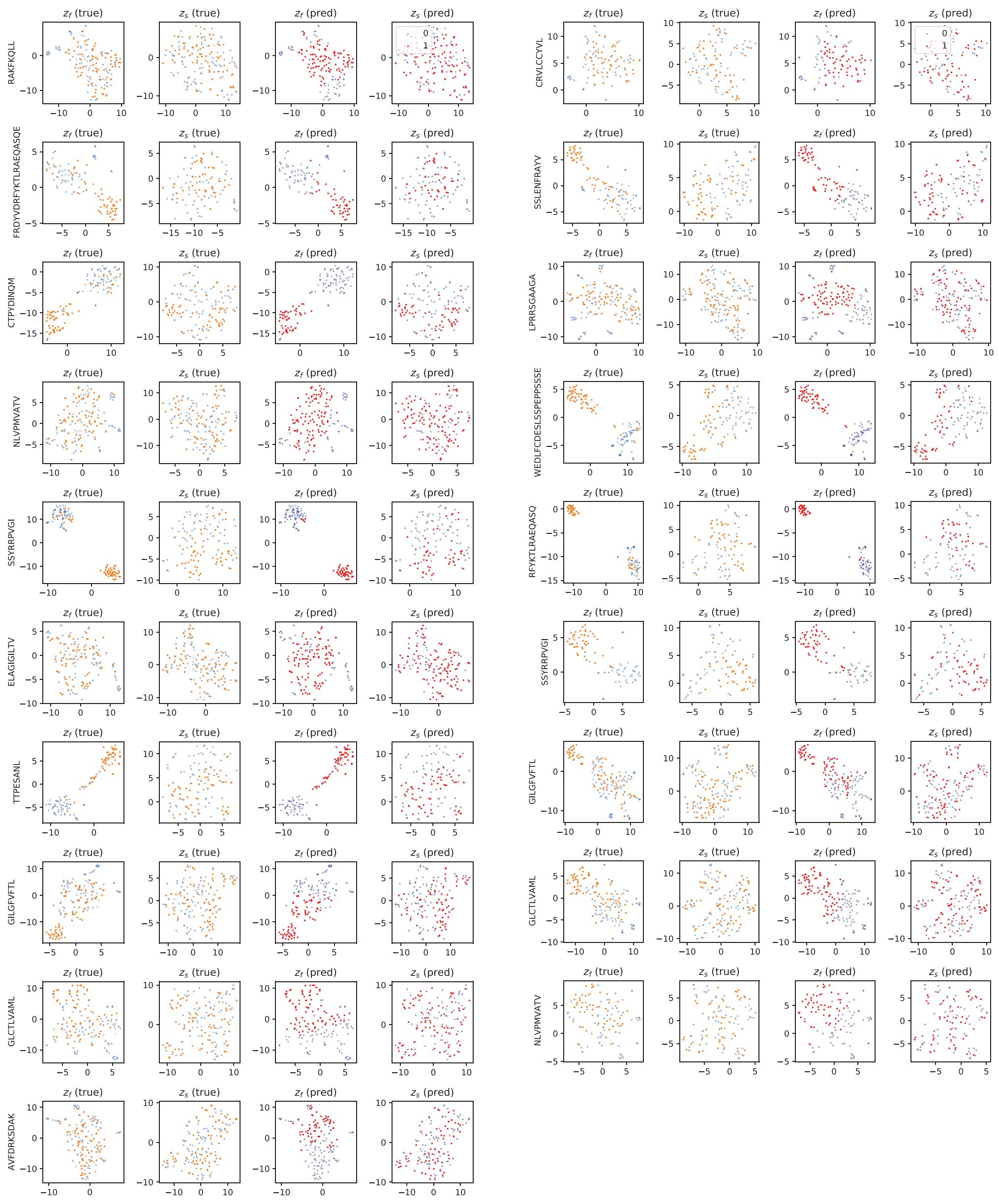

Fig. S2: T-SNE of $\mathbf{z}_f$ and $\mathbf{z}_s$ embeddings for all peptides in VDJDB (left) and MCPAS (right). Points are colored by the label. “True" means the ground truth label. “Pred" refers to label predcited by the function classifier Ψ.

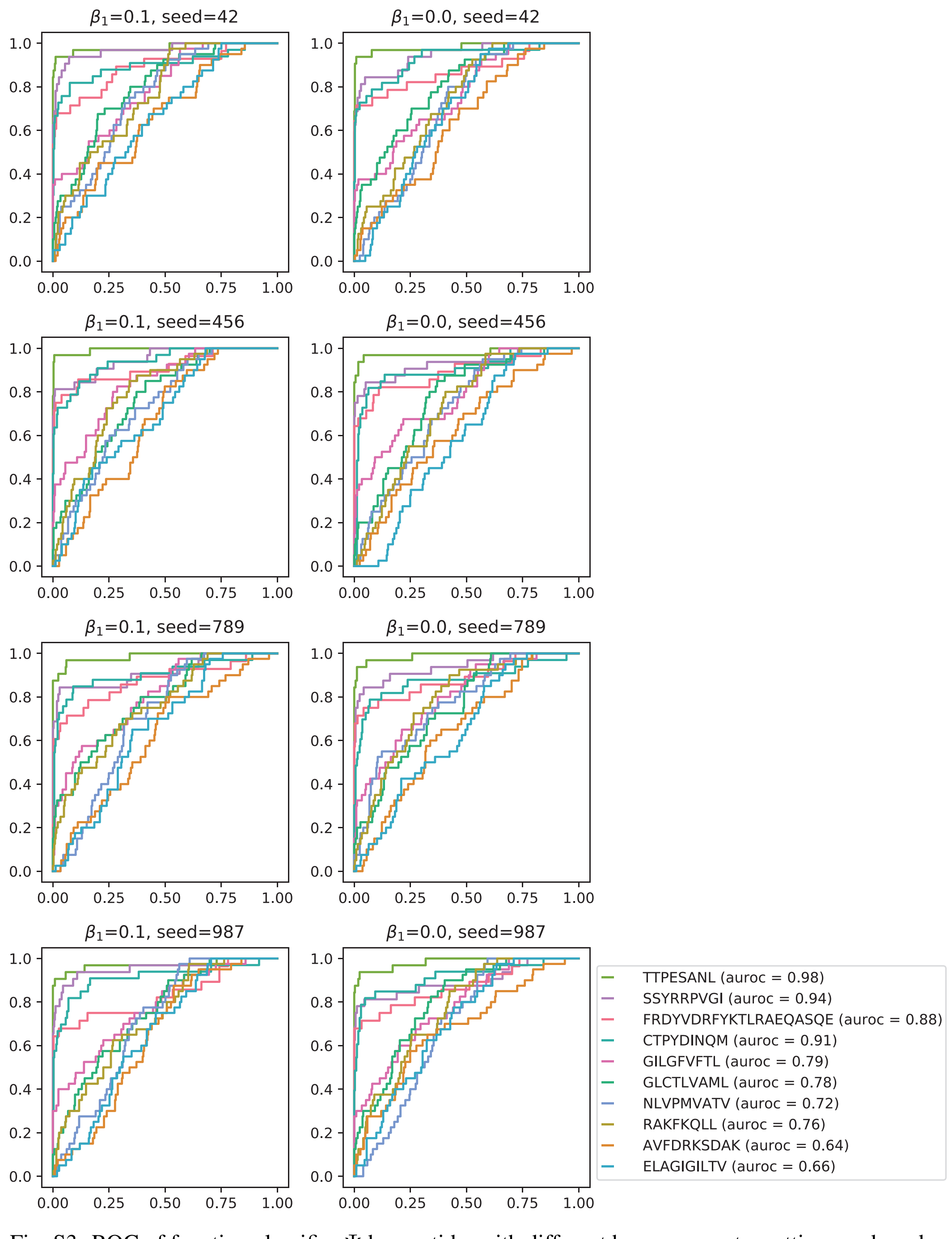

Fig. S3: ROC of function classifier Ψ by peptide, with different hyperparameter settings and random seeds.

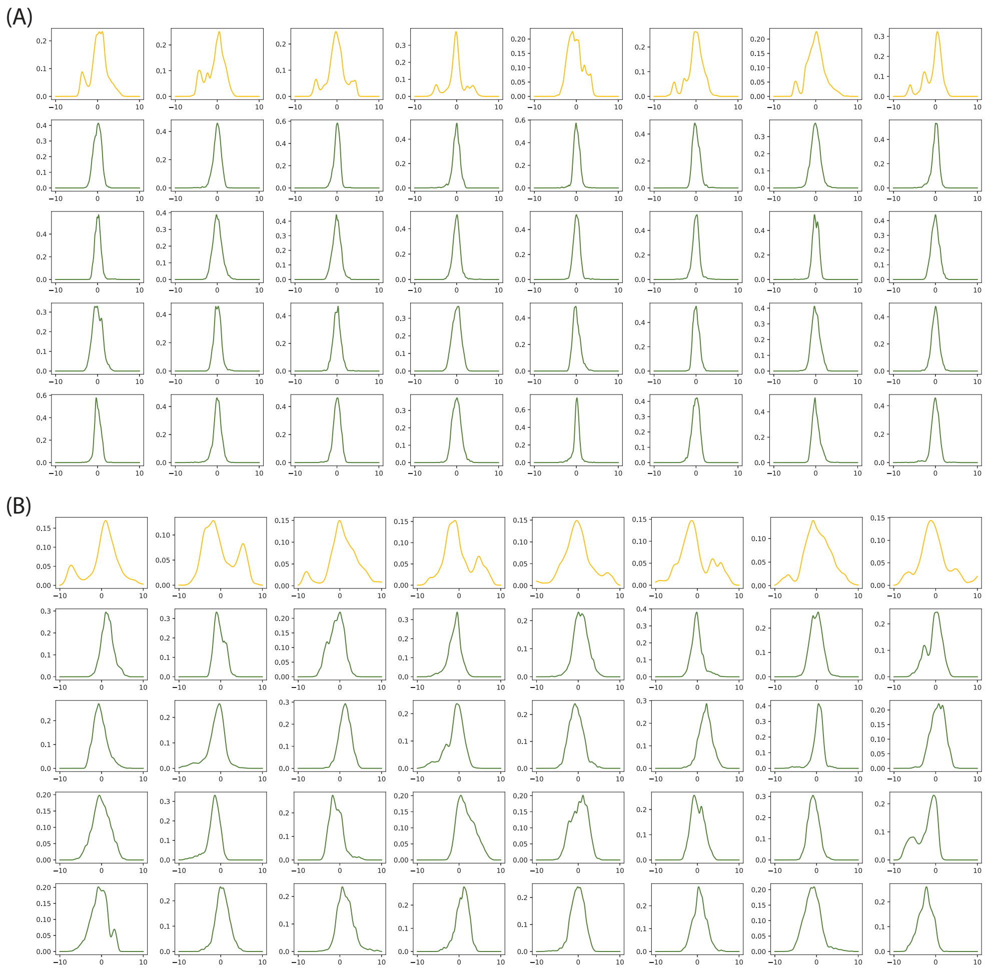

Fig. S4: Distribution of the latent embeddings with (A) and without (B) Wasserstein loss. Orange lines correspond to dimensions of $\mathbf{z}_f$ and green lines $\mathbf{z}_f$. The distribution is estimated using `gaussian_kde` from the scipy package.

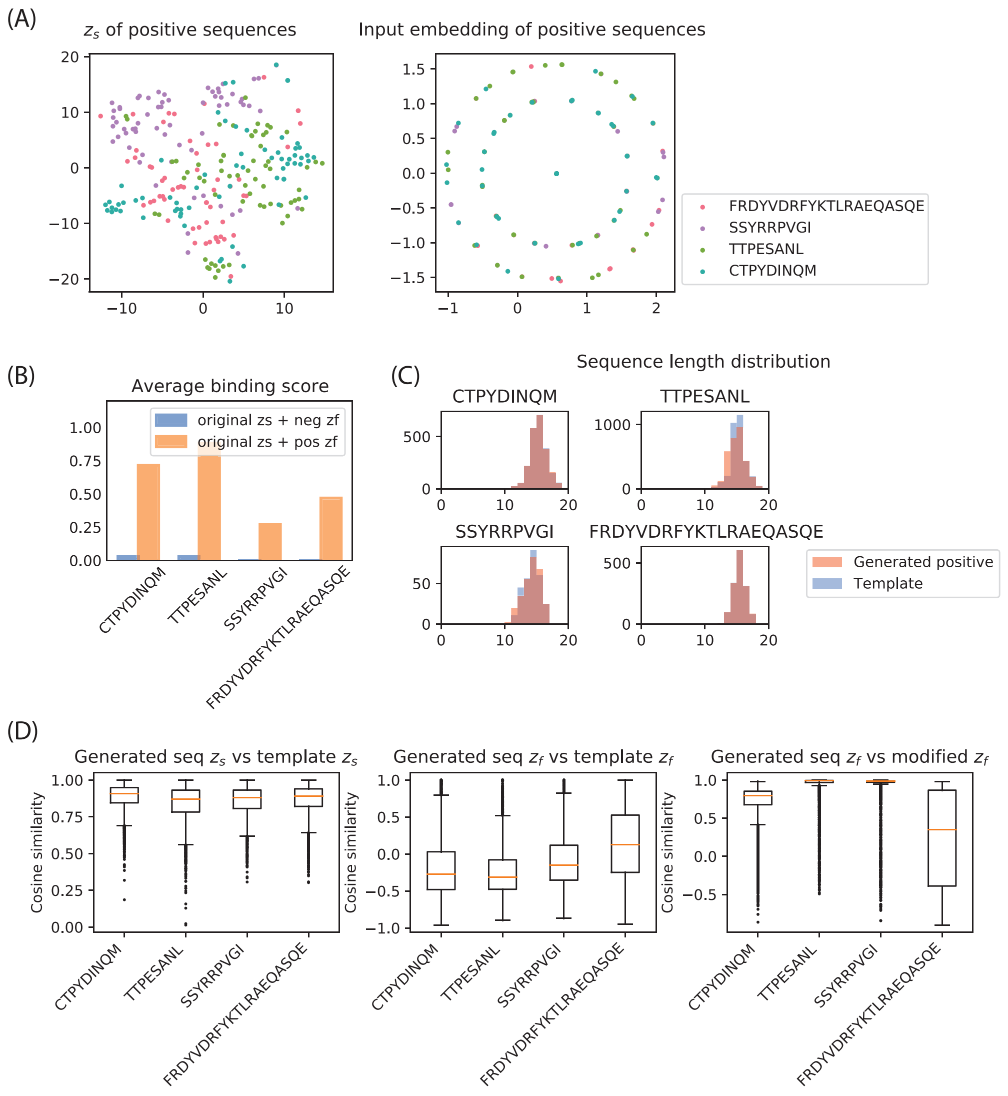

Fig. S5: (A) T-SNE of $\mathbf{z}_s$ (left) and first layer embedding of the encoder (right) of positive TCRs, colored by their binding peptides. (B) The average binding score of generated positive and negative TCRs. (C) The length distribution of template and optimized TCRs (CDR3β region) from VDJDB. (D) Cosine similarity between $\mathbf{z}_s$ of the optimized sequences vs their templates (left), $\mathbf{z}_f$ of the optimized sequences vs their templates (middle), $\mathbf{z}_f$ of the optimized sequences vs the modified $\mathbf{z}_f$ (right).

References

[1] Dmitry V Bagaev, Renske MA Vroomans, Jerome Samir, Ulrik Stervbo, Cristina Rius, Garry Dolton, Alexander Greenshields-Watson, Meriem Attaf, Evgeny S Egorov, Ivan V Zvyagin, et al. Vdjdb in 2019: database extension, new analysis infrastructure and a t-cell receptor motif compendium. *Nucleic Acids Research*, 48(D1):D1057–D1062, 2020.

[2] Nili Tickotsky, Tal Sagiv, Jaime Prilusky, Eric Shifrut, and Nir Friedman. Mcpas-tcr: a manually curated catalogue of pathology-associated t cell receptor sequences. *Bioinformatics*, 33(18): 2924–2929, 2017.

[3] Alessandro Montemurro, Viktoria Schuster, Helle Rus Povlsen, Amalie Kai Bentzen, Vanessa Jurtz, William D Chronister, Austin Crinklaw, Sine R Hadrup, Ole Winther, Bjoern Peters, et al. Nettcr-2.0 enables accurate prediction of tcr-peptide binding by using paired tcrα and β sequence data. *Communications biology*, 4(1):1–13, 2021.

[4] Ido Springer, Hanan Besser, Nili Tickotsky-Moskovitz, Shirit Dvorkin, and Yoram Louzoun. Prediction of specific tcr-peptide binding from large dictionaries of tcr-peptide pairs. *Frontiers in immunology*, page 1803, 2020.

[5] Steven Henikoff and Jorja G Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

[8] Pengyu Cheng, Martin Renqiang Min, Dinghan Shen, Christopher Malon, Yizhe Zhang, Yitong Li, and Lawrence Carin. Improving disentangled text representation learning with information-theoretic guidance. *arXiv preprint arXiv:2006.00693*, 2020.

[9] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

[10] Si-Yi Chen, Tao Yue, Qian Lei, and An-Yuan Guo. Tcrdb: a comprehensive database for t-cell receptor sequences with powerful search function. *Nucleic Acids Research*, 49(D1):D468–D474, 2021.

[11] Uniprot: the universal protein knowledgebase in 2021. *Nucleic acids research*, 49(D1):D480–D489, 2021.