SUPPLEMENT A: Modern Hopfield Networks and Attention for Immune Repertoire Classification

Michael Widrich[*] **Bernhard Schäfl**[*] **Milena Pavlović**[†,‡] **Geir Kjetil Sandve**[‡]

Sepp Hochreiter[*,§] **Victor Greiff**[†]

Günter Klambauer[*]
[*]ELLIS Unit Linz and LIT AI Lab,
Institute for Machine Learning,
Johannes Kepler University Linz, Austria
[†]Department of Immunology, University of Oslo, Norway
[‡]Department of Informatics, University of Oslo, Norway
[§]Institute of Advanced Research in Artificial Intelligence (IARAI)

Contents

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

A1 Introduction

This document is a supplement to the paper "Modern Hopfield Networks and Attention for Immune Repertoire Classification". All datasets and code will be fully released at `https://github.com/ml-jku/DeepRC`. The *CMV dataset* is publicly available at `https://clients.adaptivebiotech.com/pub/Emerson-2017-NatGen`.

A2 Notation overview

Definition	Symbol/Notation	Dimension
bag / input object / repertoire	X	set of N instances
sequence / instance	s_i	$d_l \times (20+3)$
space of sequences	$\mathcal{S}$	
instance-level representation of sequence s_i	$\boldsymbol{z}_i$	d_v
instance-level representation matrix of $(s_1, \ldots, s_N)$	$\boldsymbol{Z}$	$N \times d_v$
repertoire-level representation of bag X	$\boldsymbol{z}$	d_v
attention value of the i-th instance	a_i	
standard basis vector	$\boldsymbol{e}_m$	d_v
k-mer representation of a sequence s_i	$\boldsymbol{u}_i$	d_u
k-mer representation of a repertoire X	$\boldsymbol{u}$	d_u
pattern	$\boldsymbol{x}_i$	d or d_k
pattern matrix	$\boldsymbol{X}$	$d \times N$ or $d_k \times N$
query	$\boldsymbol{\xi}$	d or d_k
key matrix	$\boldsymbol{K}$	$N \times d_k$
query matrix	$\boldsymbol{Q}$	$N \times d_k$ or $d_q \times d_k$
value matrix	$\boldsymbol{V}$	$N \times d_v$
key embedding matrix	$\boldsymbol{W}_K$	$d_y \times d_k$
query embedding matrix	$\boldsymbol{W}_Q$	$d_y \times d_k$
value embedding matrix	$\boldsymbol{W}_V$	$d_k \times d_v$
pattern before embedding	$\boldsymbol{y}_i$	d_y
pattern matrix before embedding	$\boldsymbol{Y}$	$N \times d_y$
number of patterns or instances	N	
largest norm of a pattern	M	
separation of pattern $\boldsymbol{x}_i$	Δ_i	
scale/temperature parameter	β	
sequence embedding function	h	
pooling function	f	
bag classifier function	g	
output function / output layer	o	
classifier output / prediction for bag X	$\hat{y}$	
k-mer extraction function	h_{kmer}	
sub-network of DeepRC	h_1	
sub-network of DeepRC	h_2	
network parameters of h_1	$\boldsymbol{\theta}_1$	
network parameters of h_2	$\boldsymbol{\theta}_2$	
network parameters of o	$\boldsymbol{\theta}_o$	
dimension of a key	d_k	
dimension of a value	d_v	
dimension of a query	d_q	
dimension of a k-mer representation	d_u	
dimension of pattern $\boldsymbol{x}_i$	d	
dimension of pattern $\boldsymbol{y}_i$	d_y	
length of input sequence	d_l	
witness ratio / motif frequency	ρ	

Table A1: Symbols and notations used in this paper.

A3 DeepRC implementation details

The implementation of our method is provided at `https://github.com/ml-jku/DeepRC`.

Input layer. For the input layer of the CNN, the characters in the input sequence, i.e. the amino acids (AAs), are encoded in a one-hot vector of length 20. To also provide information about the position of an AA in the sequence, we add 3 additional input features with values in range $[0, 1]$ to encode the position of an AA relative to the sequence. These 3 positional features encode whether the AA is located at the beginning, the center, or the end of the sequence, respectively, as shown in Figure A1. We concatenate these 3 positional features with the one-hot vector of AAs, which results in a feature vector of size 23 per sequence position. Each repertoire, now represented as a bag of feature vectors, is then normalized to unit variance.
We feed the sequences with 23 features per position into the CNN. Sequences of different lengths were zero-padded to the maximum sequence length per batch at the sequence ends.

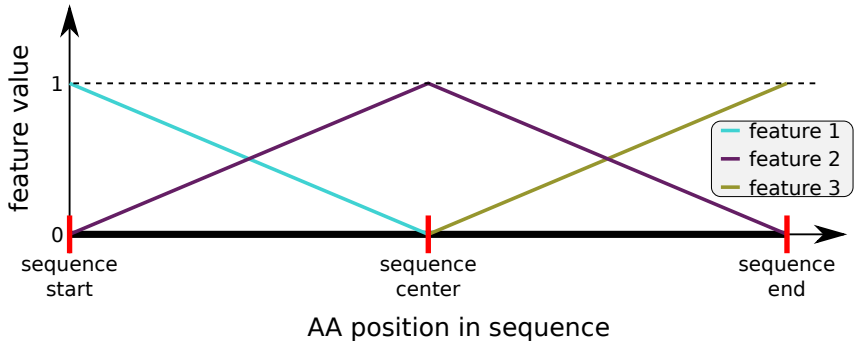

Figure A1: We use 3 input features with values in range $[0, 1]$ to encode the relative position of each AA in a sequence with respect to the sequence. "feature 1" encodes if an AA is close to the sequence start, "feature 2" to the sequence center, and "feature 3" to the sequence end. For every position in the sequence, the values of all three features sum up to 1.

Sequence duplicates and abundance. The cytomegalovirus dataset (*CMV dataset*) (Emerson et al., 2017) provides sequences with an associated pre-processed abundance value per sequence, which indicates the number of occurrences of a sequence in a repertoire. We incorporate this information into the input of DeepRC such that the one-hot AA features of a sequence are multiplied by a scaling factor of $\log(sequence\_abundance)$ before normalization.
Other duplicated sequences are fed as separate sequences into the network, i.e. multiple sequence instances might be identical if the sequence occurs multiple times in the repertoire.

1D CNN for motif recognition. In the following, we describe how DeepRC identifies patterns in the individual sequences and reduces each sequence in the input object to a fixed-size feature vector. DeepRC employs 1D convolution layers to extract patterns, where trainable weight kernels are convolved over the sequence positions. In principle, also recurrent neural networks (RNNs) or transformer networks could be used instead of 1D CNNs, however, (a) the computational complexity of the network must be low to be able to process millions of sequences for a single update. Additionally, (b) the learned network should be able to provide insights in the recognized patterns in form of motifs. Both properties (a) and (b) are fulfilled by 1D convolution operations that are used by DeepRC.
We use one 1D CNN layer (Hu et al., 2014) with SELU activation functions (Klambauer et al., 2017) to identify the relevant patterns in the input sequences with a computationally light-weight operation. The larger the kernel size, the more surrounding sequence positions are taken into account, which influences the length of the motifs that can be extracted. We therefore adjust the kernel size during hyperparameter search. In prior works (Ostmeyer et al., 2019), a k-mer size of 4 yielded good predictive performance, which could indicate that a kernel size in the range of 4 may be a proficient choice. For d_v trainable kernels, this produces a feature vector of length d_v at each sequence position. Subsequently, global max-pooling over all sequence positions of a sequence reduces the sequence-representations $\boldsymbol{z}_i$ to vectors of the fixed length d_v. Given the challenging size of the input data per repertoire, the computation of the CNN activations and weight updates is performed using

16-bit floating point values. A list of hyperparameters evaluated for DeepRC is given in Table A4. A comparison of RNN-based and CNN-based sequence embedding for motif recognition in a smaller experimental setting is given in Sec. A11.

Regularization. During training, we apply random subsampling of repertoire sequences, which can be interpreted as random drop-out (Hinton et al., 2012) on the input sequences or attention weights, to reduce over-fitting and decrease the computational cost. For this, each repertoire is subsampled to $10,000$ input sequences, which are randomly drawn from the respective repertoire.
Additionally, one might employ further regularization techniques, which we only partly investigated further in a smaller experimental setting in Sec. A11 due to high computational demands. Such regularization techniques include $l1$ and $l2$ weight decay, noise in the form of random AA permutations in the input sequences, noise on the attention weights, or random shuffling of sequences between repertoires that belong to the negative class. The last regularization technique assumes that the sequences in positive-class repertoires carry a signal, such as an AA motif corresponding to an immune response, whereas the sequences in negative-class repertoires do not. Hence, the sequences can be shuffled randomly between negative class repertoires without obscuring the signal in the positive class repertoires.

Reduction of computational cost and memory consumption. We took measures to address the high computational demands, especially GPU memory consumption, in order to make the large number of experiments feasible:
We train the DeepRC model with a small batch size of 4 samples and perform computation of inference and updates of the 1D CNN using 16-bit floating point values. The rest of the network is trained using 32-bit floating point values. The Adam parameter for numerical stability was therefore increased from the default value of $\epsilon = 10^{-8}$ to $\epsilon = 10^{-4}$.
During training and evaluation, we apply attention-based subsampling of repertoire sequences to reduce the memory consumption and computational cost. For this, the attention weights computed by the attention network are used to rank the input sequences. Based on this ranking, the repertoire is reduced to the 10% of sequences with the highest attention weights. These top 10% of sequences are then used to compute the weight updates and the prediction for the repertoire.

Computation time. Training was performed on various GPU types, mainly `NVIDIA RTX 2080 Ti`. Computation times were highly dependent on the number of sequences in the repertoires and the number and sizes of CNN kernels. A single forward and backward pass with weight update on an `NVIDIA RTX 2080 Ti` GPU took approximately 0.0109 to 0.0135 seconds, while requiring approximately 8 to 11 GB GPU memory. The average total required time for one update step, including the loading of 4 samples, their reduction to the 10% of sequences with the highest attention weights, the weight update, the computation of validation scores for early-stopping (pro rata), and the logging of results, was approximately 0.1 seconds. Taking GPUs with larger memory (≥ 16 GB) into account, it is already possible to train DeepRC on larger datasets, possibly with multi-head attention and a larger network architectures (see Sec. A11). Our network implementation is based on `PyTorch 1.3.1` (Paszke et al., 2019).

Incorporation of additional inputs and metadata. Additional metadata in the form of sequence-level or repertoire-level features could be incorporated into the input via concatenation with the feature vectors that result from taking the maximum of the 1D CNN outputs w.r.t. the sequence positions. This has the benefit that the attention mechanism and output network can utilize the sequence-level or repertoire-level features for their predictions. Sparse metadata or metadata that is only available during training could be used as auxiliary targets to incorporate the information via gradients into the DeepRC model.

Limitations. The current methods are mostly limited by computational complexity, since both hyperparameter and model selection is computationally demanding. For hyperparameter selection, a large number of hyperparameter settings have to be evaluated. For model selection, a single repertoire requires the propagation of many thousands of sequences through a neural network and keeping those quantities in GPU memory in order to perform the attention mechanism and weight update. Thus, increased GPU memory would significantly boost our approach. Increased computational power would also allow for more advanced architectures and attention mechanisms, which may further improve predictive performance. Another limiting factor is over-fitting of the model due to the currently relatively small number of samples (bags) in real-world immunosequencing datasets in comparison to the large number of instances per bag and features per instance.

Hyperparameters. For the hyperparameter search of DeepRC for the category "simulated immunosequencing data", we only conducted a full hyperparameter search on the more difficult datasets with motif implantation probabilities below 1%, as described in Table A4. This process was repeated for all 5 folds of the 5-fold cross-validation (CV) and the average score on the 5 test sets constitutes the final score of a method.
Table A4 provides an overview of the hyperparameter search, which was conducted as a grid search for each of the datasets in a nested 5-fold CV procedure, as described in section A5.

A4 Datasets

We aimed at constructing immune repertoire classification scenarios with varying degree of realism and difficulties in order to compare and analyze the suggested machine learning methods. To this end, we either use simulated or experimentally-observed immune receptor sequences and we implant signals, which are sequence motifs (Akbar et al., 2019; Weber et al., 2020), into sequences of repertoires of the positive class. It has been shown previously that interaction of immune receptors with antigens occur via short sequence stretches (Akbar et al., 2019). Thus, implantation of short motif sequences simulating an immune signal is biologically meaningful. Our benchmarking study comprises four different categories of datasets: (a) Simulated immunosequencing data with implanted signals (where the signal is defined as sets of motifs), (b) LSTM-generated immunosequencing data with implanted signals, (c) real-world immunosequencing data with implanted signals, and (d) real-world immunosequencing data. Each of the first three categories consists of multiple datasets with varying difficulty depending on the type of the implanted signal and the ratio of sequences with the implanted signal. The ratio of sequences with the implanted signal, where each sequence carries at most 1 implanted signal, corresponds to the *witness rate* (WR). We consider binary classification tasks to simulate the immune status of healthy and diseased individuals. We randomly generate immune repertoires with varying numbers of sequences, where we implant sequence motifs in the repertoires of the diseased individuals, i.e. the positive class. The sequences of a repertoire are also randomly generated by different procedures (detailed below). Each sequence is composed of 20 different characters, corresponding to amino acids, and has an average length of 14.5 AAs.

A4.1 Simulated immunosequencing data

In the first category, we aim at investigating the impact of the signal frequency, i.e. the WR, and the signal complexity on the performance of the different methods. To this end, we created 21 datasets, whereas each dataset contains a large number of repertoires with a large number of random AA sequences per repertoire. We then implanted signals in the AA sequences of the positive class repertoires, where the 21 datasets differ in frequency and complexity of the implanted signals. In detail, the AAs were sampled randomly independent of their respective position in the sequence, while the frequencies of AAs, distribution of sequence lengths, and distribution of the number of sequences per repertoire, i.e. the number of instances per bag, are following the respective distributions observed in the real-world *CMV dataset* (Emerson et al., 2017). For this, we first sampled the number of sequences for a repertoire from a Gaussian $\mathcal{N}(\mu = 316k, \sigma = 132k)$ distribution and rounded to the nearest positive integer. We re-sampled if the size was below $5k$. We then generated random sequences of AAs with a length of $\mathcal{N}(\mu = 14.5, \sigma = 1.8)$, again rounded to the nearest positive integers. Each simulated repertoire was then randomly assigned to either the positive or negative class, with $2,500$ repertoires per class. In the repertoires assigned to the positive class, we implanted motifs with an average length of 4 AAs, following the results of the experimental analysis of antigen-binding motifs in antibodies and T-cell receptor sequences by Akbar et al. (2019). We varied the characteristics of the implanted motifs for each of the 21 datasets with respect to the following parameters: (a) ρ, the probability of a motif being implanted in a sequence of a positive repertoire, i.e. the average ratio of sequences containing the motif, which is the witness rate. (b) The number of wildcard positions in the motif. A wildcard position contains a random AA, which is randomly sampled for each sequence. Wildcard positions are located in the center of the implanted motif. (c) The number of deletion positions in the implanted motif. A deletion position has a probability of 0.5 of being removed from the motif. Deletion positions are located in the center of the implanted motifs. In this way, we generated 18 different datasets of variable difficulty containing in total roughly 28.7 billion sequences. Additionally, we added 3 datasets in which every position has a 20% probability of behaving like a wildcard position to evaluate the performance on motifs with increased noise, resulting in a total of 21 different datasets. See Table A2 for an overview of the properties of the implanted motifs in the 21 datasets.

A4.2 LSTM-generated data

In the second dataset category, we investigate the impact of the signal frequency and complexity in combination with more plausible immune receptor sequences by taking into account the positional AA distributions and other sequence properties. To this end, we trained an LSTM (Hochreiter & Schmidhuber, 1997) in a standard next character prediction (Graves, 2013) setting to create AA sequences with properties similar to experimentally observed immune receptor sequences.
In the first step, the LSTM model was trained on all immuno-sequences in the *CMV dataset* (Emerson et al., 2017) that contain valid information about sequence abundance and have a known CMV label. Such an LSTM model is able to capture various properties of the sequences, including position-dependent probability distributions and combinations, relationships, and order of AAs. We then used the trained LSTM model to generate $1,000$ repertoires in an autoregressive fashion, starting with a start sequence that was randomly sampled from the trained-on dataset. Based on a visual inspection of the frequencies of 4-mers (see section A8), the similarity of LSTM generated sequences and real sequences was deemed sufficient for the purpose of generating the AA sequences for the datasets in this category. Further details on LSTM training and repertoire generation are given in Section A8.
After generation, each repertoire was assigned to either the positive or negative class, with 500 repertoires per class. We implanted motifs of length 4 with varying properties in the center of the sequences of the positive class to obtain 5 different datasets. Each sequence in the positive repertoires has a probability ρ to carry the motif, which was varied throughout 5 datasets and corresponds to the WR (see Table A2). Each position in the motif has a probability of 0.9 to be implanted and consequently a probability of 0.1 that the original AA in the sequence remains, which can be seen as noise on the motif.

	Simulated	LSTM gen.	Real-world
seq. per bag	$N(316k, 132k)$	$N(285k, 156k)$	$10k$
repertoires	$5,000$	$1,000$	$1,500$
motif noise	0%	10%	$*$
wildcards	$\{0; 1; 2\}$	0	0
deletions	$\{0; 1\}$	0	0
mot. freq. ρ (in $\%$)	$\{1; 0.1; 0.01\}$	$\{10; 1; 0.5; 0.1; 0.05\}$	$\{1; 0.1\}$

Table A2: Properties of simulated repertoires, variations of motifs, and motif frequencies, i.e. the witness rate, for the datasets in categories "simulated immunosequencing data", "LSTM-generated data", and "real-world data with implanted signals". Noise types for $*$ are explained in paragraph "real-world data with implanted signals".

A4.3 Real-world data with implanted signals

In the third category, we implanted signals into experimentally obtained immuno-sequences, where we considered 4 dataset variations. Each dataset consists of 750 repertoires for each of the two classes, where each repertoire consists of $10k$ sequences. In this way, we aim to simulate datasets with a *low sequencing coverage*, which means that only relatively few sequences per repertoire are available. The sequences were randomly sampled from healthy (CMV negative) individuals from the *CMV dataset* (see below paragraph for explanation). Two signal types were considered: (a) **One signal with one motif.** The AA motif `LDR` was implanted in a certain fraction of sequences. The pattern is randomly altered at one of the three positions with probabilities 0.2, 0.6, and 0.2, respectively. (b) **One signal with multiple motifs.** One of the three possible motifs `LDR`, `CAS`, and `GL-N` was implanted with equal probability. Again, the motifs were randomly altered before implantation. The AA motif `LDR` changed as described above. The AA motif `CAS` was altered at the second position with probability 0.6 and with probability 0.3 at the first position. The pattern `GL-N`, where `-` denotes a gap location, is randomly altered at the first position with probability 0.6 and the gap has a length of 0, 1, or 2 AAs with equal probability.
Additionally, the datasets differ in the values for ρ, the average ratio of sequences carrying a signal, which were chosen as 1% or 0.1%. The motifs were implanted at positions 107, 109, and 114 according to the IMGT numbering scheme for immune receptor sequences (Lefranc et al., 2003) with probabilities 0.3, 0.35 and 0.2, respectively. With the remaining 0.15 chance, the motif is implanted

at any other sequence position. This means that the motif occurrence in the simulated sequences is biased towards the middle of the sequence.

A4.4 Real-world data: CMV dataset

We used a real-world dataset of 785 repertoires, each of which containing between $4,371$ to $973,081$ (avg. $299,319$) TCR sequences with a length of 1 to 27 (avg. 14.5) AAs, originally collected and provided by Emerson et al. (2017). 340 out of 785 repertoires were labelled as positive for cytomegalovirus (CMV) serostatus, which we consider as the positive class, 420 repertoires with negative CMV serostatus, considered as negative class, and 25 repertoires with unknown status. We changed the number of sequence counts per repertoire from -1 to 1 for 3 sequences. Furthermore, we exclude a total of 99 repertoires with unknown CMV status or unknown information about the sequence abundance within a repertoire, reducing the dataset for our analysis to 686 repertoires, 312 of which with positive and 374 with negative CMV status.

A4.5 Comparison to other MIL datasets

Dataset	Total number of bags	Total number of instances	Approx. number of features per instance	Avg. number of instances per bag	Source	Dataset reference
Simulated immuno-sequencing data (ours)	5,000	1,597,024,310 x 21 datasets	14.5x20 AA sequence	316,000	this work	
LSTM-generated data (ours)	1,000	304,825,671 x 5 datasets	14.5x20 AA sequence	285,000	this work	
Real-world data with implanted signals (ours)	1,500	14,715,421 x 4 datasets	14.5x20 AA sequence	10,000	this work	
CMV (pre-processed by us)	785	234,965,729	14.5x20 AA sequence	299,000	this work	Emerson et al. (2017)
MNIST bags	50–500	500–50,000	28x28x1 image	100	Ilse et al. (2018)	
Breast Cancer	58	approx. 39,000	32x32x3 H&E image	672	Ilse et al. (2018)	Gelasca et al. (2008)
Basal cell carcinomas	820	7,588,767	1024x1024x3 H&E image	9,056	Kimeswenger et al. (2019)	
Birds	548	10,232	38	9	Ruiz et al. (2018)	Briggs et al. (2012)
Scene	2,000	18,000	15	9	Ruiz et al. (2018)	Zhang & Zhang (2007)
Reuters	2,000	7,119	243	4	Ruiz et al. (2018)	Sebastiani (2002)
CK+	430	7,915	4,391	18	Ruiz et al. (2018)	Lucey et al. (2010)
UniProt (Geobacter sulfurreducens)	379	1,250	216	3	Ruiz et al. (2018)	Wu et al. (2014)
MODIS (aerosol data)	1,364	136,400	12	100	Uriot (2019)	`https://aeronet.gsfc.nasa.gov`
MISR1 (aerosol data)	800	80,000	16	100	Uriot (2019)	`https://aeronet.gsfc.nasa.gov`
MISR2 (aerosol data)	800	80,000	12	54	Uriot (2019)	`https://aeronet.gsfc.nasa.gov`
CORN (crop yield)	525	52,500	92	100	Uriot (2019)	`https://aeronet.gsfc.nasa.gov`
WHEAT (crop yield)	525	52,500	92	100	Uriot (2019)	`https://aeronet.gsfc.nasa.gov`

Table A3: MIL datasets with their numbers of bags and numbers of instances. “total number of instances” refers to the total number of instances in the dataset. The simulated and real-world immunosequencing datasets considered in this work contain a by orders of magnitudes larger number of instances per bag than MIL datasets that were considered by machine learning methods up to now.

A5 Compared methods

We evaluate and compare the performance of DeepRC against a set of machine learning methods that serve as baseline, were suggested, or can readily be adapted to immune repertoire classification. In this section, we describe these compared methods.

A5.1 Known motif

This method serves as an estimate for the achievable classification performance using prior knowledge about which motif was implanted. Note that this does not necessarily lead to perfect predictive performance since motifs are implanted with a certain amount of noise and could also be present in the negative class by chance. The *known motif* method counts how often the known implanted motif occurs per sequence for each repertoire and uses this count to rank the repertoires. From this ranking, the Area Under the receiver operator Curve (AUC) is computed as performance measure. Probabilistic AA changes in the known motif are not considered for this count, with the exception of gap positions. We consider two versions of this method: (a) **Known motif binary:** counts the occurrence of the known motif in a sequence and (b) **Known motif continuous:** counts the maximum number of overlapping AAs between the known motif and all sequence positions, which corresponds to a convolution operation with a binary kernel followed by max-pooling. Since the implanted signal is not known in the experimentally obtained *CMV dataset*, this method cannot be applied to this dataset.

A5.2 Support Vector Machine (SVM)

The Support Vector Machine (SVM) approach uses a fixed mapping from a bag of sequences to the corresponding k-mer counts. The function h_{kmer} maps each sequence s_i to a vector representing the occurrence of k-mers in the sequence. To avoid confusion with the sequence-representation obtained from the CNN layers of DeepRC, we denote $\boldsymbol{u}_i = h_{\text{kmer}}(s_i)$, which is analogous to $\boldsymbol{z}_i$. Specifically, $u_{im} = (h_{\text{kmer}}(s_i))_m = \#\{p_m \in s_i\}$, where $\#\{p_m \in s_i\}$ denotes how often the k-mer pattern p_m occurs in sequence s_i. Afterwards, average-pooling is applied to obtain $\boldsymbol{u} = 1/N \sum_{i=1}^{N} \boldsymbol{u}_i$, the *k-mer representation* of the input object X. For two input objects $X^{(n)}$ and $X^{(l)}$ with representations $\boldsymbol{u}^{(n)}$ and $\boldsymbol{u}^{(l)}$, respectively, we implement the *MinMax kernel* (Ralaivola et al., 2005) as follows:

$$\begin{aligned} k(X^{(n)}, X^{(l)}) &= k_{\text{MinMax}}(\boldsymbol{u}^{(n)}, \boldsymbol{u}^{(l)}) \\ &= \frac{\sum_{m=1}^{d_u} \min(u_m^{(n)}, u_m^{(l)})}{\sum_{m=1}^{d_u} \max(u_m^{(n)}, u_m^{(l)})}, \end{aligned} \tag{1}$$

where $u_m^{(n)}$ is the m-th element of the vector $u^{(n)}$. The *Jaccard kernel* (Levandowsky & Winter, 1971) is identical to the MinMax kernel except that it operates on binary $\boldsymbol{u}^{(n)}$. We used a standard C-SVM, as introduced by Cortes & Vapnik (1995). The corresponding hyperparameter C is optimized by random search. The settings of the full hyperparameter search as well as the respective value ranges are given in Table A5.

A5.3 K-Nearest Neighbor (KNN)

The same *k-mer representation* of a repertoire, as introduced above for the SVM baseline, is used for the K-Nearest Neighbor (KNN) approach. As this method clusters samples according to distances between them, the previous kernel definitions cannot be applied directly. It is therefore necessary to transform the MinMax as well as the Jaccard kernel from similarities to distances by constructing the following (Levandowsky & Winter, 1971):

$$\begin{aligned} d_{\text{MinMax}}(\boldsymbol{u}^{(n)}, \boldsymbol{u}^{(l)}) &= 1 - k_{\text{MinMax}}(\boldsymbol{u}^{(n)}, \boldsymbol{u}^{(l)}), \\ d_{\text{Jaccard}}(\boldsymbol{u}^{(n)}, \boldsymbol{u}^{(l)}) &= 1 - k_{\text{Jaccard}}(\boldsymbol{u}^{(n)}, \boldsymbol{u}^{(l)}). \end{aligned} \tag{2}$$

The amount of neighbors is treated as the hyperparameter and optimized by an exhaustive grid search. The settings of the full hyperparameter search as well as the respective value ranges are given in Table A6.

A5.4 Logistic regression

We implemented logistic regression on the *k-mer representation* $\boldsymbol{u}$ of an immune repertoire. The model is trained by gradient descent using the Adam optimizer (Kingma & Ba, 2014). The learning

rate is treated as the hyperparameter and optimized by grid search. Furthermore, we explored two regularization settings using combinations of $l1$ and $l2$ weight decay. The settings of the full hyperparameter search as well as the respective value ranges are given in Table A7.

A5.5 Burden test

We implemented a burden test (Emerson et al., 2017; Li & Leal, 2008; Wu et al., 2011) in a machine learning setting. The burden test first identifies sequences or k-mers that are associated with the individual's class, i.e., immune status, and then calculates a burden score per individual. Concretely, for each k-mer or sequence, the phi coefficient of the contingency table for absence or presence and positive or negative immune status is calculated. Then, J k-mers or sequences with the highest phi coefficients are selected as the set of associated k-mers or sequences. J is a hyperparameter that is selected on a validation set. Additionally, we consider the type of input features, sequences or k-mers, as a hyperparameter. For inference, a burden score per individual is calculated as the sum of associated k-mers or sequences it carries. This score is used as raw prediction and to rank the individuals. Hence, we have extended the burden test by Emerson et al. (2017) to k-mers and to adaptive thresholds that are adjusted on a validation set.

A5.6 Logistic MIL (Ostmeyer et al)

The logistic multiple instance learning (MIL) approach for immune repertoire classification (Ostmeyer et al., 2019) applies a logistic regression model to each k-mer representation in a bag. The resulting scores are then summarized by max-pooling to obtain a prediction for the bag. Each amino acid of each k-mer is represented by 5 features, the so-called Atchley factors (Atchley et al., 2005). As k-mers of length 4 are used, this gives a total of $4 \times 5 = 20$ features. One additional feature per 4-mer is added, which represents the relative frequency of this 4-mer with respect to its containing bag, resulting in 21 features per 4-mer. Two options for the relative frequency feature exist, which are (a) whether the frequency of the 4-mer ("4MER") or (b) the frequency of the sequence in which the 4-mer appeared ("TCRβ") is used. We optimized the learning rate, batch size, and early stopping parameter on the validation set. The settings of the full hyperparameter search as well as the respective value ranges are given in Table A9.

A6 Hyperparameter selection

For all competing methods a hyperparameter search was performed, for which we split each of the 5 training sets into an inner training set and inner validation set. The models were trained on the inner training set and evaluated on the inner validation set. The model with the highest AUC score on the inner validation set is then used to calculate the score on the respective test set. Here we report the hyperparameter sets and search strategy that is used for all methods.

DeepRC. The set of hyperparameters of DeepRC is shown in Table A4. These hyperparameter combinations are adjusted via a grid search procedure.

learning rate	10^{-4}
number of kernels (d_v)	$\{8; 16; 32; 64^*; 128^*; 256^*\}$
number of CNN layers	$\{1\}$
number of layers in key-NN	$\{2\}$
number of units in key-NN	$\{32\}$
kernel size	$\{5; 7; 9\}$
subsampled seqences	$10,000$
batch size	4

Table A4: **DeepRC hyperparameter search space.** We apply early stopping, where the model with the best loss on the validation fold after $5 \cdot 10^5$ updates was selected. For this, the model was evaluated against the validation fold every $5 \cdot 10^3$ updates during training. *) Experiments for $\{64; 128; 256\}$ kernels were omitted for simulated datasets with motif implantation probabilities $> 0.1\%$.

Known motif. This method does not have hyperparameters and has been applied to all datasets except for the *CMV dataset*.

SVM. The corresponding hyperparameter C of the SVM is optimized by randomly drawing 10^3 values in the range of $[-6; 6]$ according to a uniform distribution. These values act as the exponents of a power of 10 and are applied for each of the two kernel types (see Table A5).

C	$10^{\{-6;6\}}$
type of kernel	$\{\text{MinMax}; \text{Jaccard}\}$
number of trials	10^3

Table A5: Settings used in the hyperparameter search of the SVM baseline approach. The number of trials defines the quantity of random values of the C penalty term (per type of kernel).

KNN. The amount of neighbors is treated as the hyperparameter and optimized by grid search operating in the discrete range of $[1; \max\{N, 10^3\}]$ with a step size of 1. The corresponding tight upper bound is automatically defined by the total amount of samples $N \in \mathbb{N}_{>0}$ in the training set, capped at 10^3 (see Table A6).

number of neighbors	$\{1; \max\{N, 10^3\}\}$
type of kernel	$\{\text{MinMax}; \text{Jaccard}\}$

Table A6: Settings used in the hyperparameter search of the KNN baseline approach. The number of trials (per type of kernel) is automatically defined by the total amount of samples $N \in \mathbb{N}_{>0}$ in the training set, capped at 10^3.

Logistic regression. For this method, we applied a grid search over the hyperparameters listed in Table A7. We varied the learning rate and the strength of the weight decay.

learning rate	$10^{-\{1;2;3;4\}}$
batch size	4
max. updates	10^5
coefficient β_1 (Adam)	0.9
coefficient β_2 (Adam)	0.999
$l1$ weight decay factor	10^{-7}
$l2$ weight decay factor	$10^{-\{3;5\}}$

Table A7: Settings used in the hyperparameter search of the logistic regression method.

Burden test. The burden test selects two hyperparameters: the number of features in the burden set and the type of features, as listed in Table A8. Due to the lack of shared sequences between repertoires in category "simulated immunosequencing data", we omitted the sequence-based burden test for this category.

number of features in burden set	$\{50, 100, 150, 250\}$
type of features	$\{\text{4MER}; \text{sequence}^*\}$

Table A8: Settings used in the hyperparameter search of the burden test approach. *) Experiments for sequence features were omitted for datasets of category "simulated immunosequencing data" due to the lack of shared sequences between repertoires.

Logistic MIL. For this method, we adjusted the learning rate as well as the batch size as hyperparameters by randomly drawing 25 different hyperparameter combinations from a uniform distribution. The corresponding range of the learning rate is $[-4.5; -1.5]$, which acts as the exponent of a power of 10. The batch size lies within the range of $[1; 32]$. For each hyperparameter combination, a model is optimized by gradient descent using Adam, whereas the early stopping parameter is adjusted according to the corresponding validation set (see Table A9).

learning rate	$10^{\{-4.5;-1.5\}}$
batch size	$\{1; 32\}$
relative abundance term	$\{\text{4MER}; \text{TCR}\beta\}$
number of trials	25
max. epochs	10^2
coefficient β_1 (Adam)	0.9
coefficient β_2 (Adam)	0.999

Table A9: Settings used in the hyperparameter search of the logistic MIL baseline approach. The number of trials (per type of relative abundance) defines the quantity of combinations of random values of the learning rate as well as the batch size.

A7 Results

In this section, we report the detailed results on all four categories of datasets (a) simulated immunosequencing data (Table A10) (b) LSTM-generated data (Table A11), (c) real-world data with implanted signals (Table A12), and (d) real-world data on the *CMV dataset* (Table A13), as discussed in the main paper.

ID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	avg.
motif freq. ρ	1%	0.1%	0.01%	1%	0.1%	0.01%	1%	0.1%	0.01%	1%	0.1%	0.01%	1%	0.1%	0.01%	1%	0.1%	0.01%	0.1%	0.1%	0.1%	–
implanted motif	SFEN	SFEN	SFEN	SF^{d}EN	SF^{d}EN	SF^{d}EN	SFZN	SFZN	SFZN	SF^{d}ZN	SF^{d}ZN	SF^{d}ZN	SZZN	SZZN	SZZN	SZ^{d}ZN	SZ^{d}ZN	SZ^{d}ZN	S^{r}F^{r}E^{r}N^{r}	S^{r}F^{r}ZN^{r}	S^{r}ZZN^{r}	–
DeepRC	**1.000**	**1.000**	0.703	**1.000**	**1.000**	0.600	**1.000**	**1.000**	0.509	**1.000**	**1.000**	0.492	**1.000**	**0.997**	0.487	0.999	**0.942**	0.492	**1.000**	**1.000**	**0.947**	0.865
	± 0.000	± 0.000	± 0.271	± 0.000	± 0.000	± 0.218	± 0.000	± 0.000	± 0.029	± 0.000	± 0.001	± 0.017	± 0.001	± 0.002	± 0.023	± 0.001	± 0.048	± 0.013	± 0.000	± 0.000	± 0.028	± 0.211
SVM (MinMax)	**1.000**	**1.000**	0.764	**1.000**	**1.000**	0.603	**1.000**	0.998	**0.539**	**1.000**	0.994	**0.529**	**1.000**	0.741	**0.513**	**1.000**	0.706	0.503	1.000	0.941	0.640	0.832
	± 0.000	± 0.000	± 0.016	± 0.000	± 0.000	± 0.021	± 0.000	± 0.002	± 0.024	± 0.000	± 0.004	± 0.016	± 0.000	± 0.024	± 0.006	± 0.000	± 0.013	± 0.013	± 0.000	± 0.004	± 0.941	± 0.203
SVM (Jaccard)	0.783	0.505	0.500	0.656	0.504	0.492	0.629	0.499	0.505	0.594	0.508	0.497	0.620	0.496	0.506	0.595	0.507	0.505	0.508	0.501	0.503	0.543
	± 0.010	± 0.009	± 0.010	± 0.009	± 0.018	± 0.018	± 0.011	± 0.010	± 0.009	± 0.007	± 0.017	± 0.013	± 0.007	± 0.006	± 0.019	± 0.013	± 0.012	± 0.017	± 0.016	± 0.015	± 0.501	± 0.076
KNN (MinMax)	0.669	0.802	0.503	0.722	0.757	0.493	0.766	0.678	0.496	0.762	0.652	0.489	0.797	0.512	0.498	0.796	0.511	0.503	0.743	0.568	0.501	0.629
	± 0.204	± 0.265	± 0.038	± 0.214	± 0.255	± 0.017	± 0.241	± 0.165	± 0.014	± 0.237	± 0.139	± 0.015	± 0.271	± 0.023	± 0.014	± 0.270	± 0.037	± 0.006	± 0.178	± 0.069	± 0.568	± 0.126
KNN (Jaccard)	0.516	0.493	0.497	0.506	0.500	0.492	0.509	0.493	0.497	0.495	0.504	0.500	0.502	0.497	0.500	0.502	0.503	**0.513**	0.498	0.506	0.491	0.501
	± 0.035	± 0.020	± 0.013	± 0.015	± 0.019	± 0.014	± 0.017	± 0.011	± 0.018	± 0.013	± 0.004	± 0.017	± 0.011	± 0.017	± 0.022	± 0.015	± 0.020	± 0.012	± 0.021	± 0.019	± 0.506	± 0.007
Logistic Regression	**1.000**	**1.000**	**0.797**	**1.000**	**1.000**	**0.614**	**1.000**	0.997	**0.539**	**1.000**	0.994	0.523	**1.000**	0.723	0.507	**1.000**	0.701	0.505	**1.000**	0.946	0.632	0.832
	± 0.000	± 0.000	± 0.026	± 0.000	± 0.000	± 0.011	± 0.000	± 0.002	± 0.010	± 0.000	± 0.004	± 0.015	± 0.000	± 0.020	± 0.020	± 0.000	± 0.007	± 0.027	± 0.001	± 0.004	± 0.946	± 0.204
Logistic MIL (KMER)	**1.000**	**1.000**	0.509	**1.000**	0.783	0.489	**1.000**	0.544	0.517	**1.000**	0.529	0.483	0.579	0.498	0.502	0.550	0.488	0.498	0.867	0.554	0.509	0.662
	± 0.000	± 0.000	± 0.039	± 0.000	± 0.216	± 0.023	± 0.000	± 0.038	± 0.018	± 0.000	± 0.043	± 0.007	± 0.042	± 0.017	± 0.018	± 0.051	± 0.009	± 0.005	± 0.206	± 0.009	± 0.554	± 0.216
Logistic MIL (TCRβ)	0.544	0.505	0.493	0.487	0.476	0.500	0.520	0.495	0.510	0.492	0.506	0.503	0.509	0.505	0.500	0.475	0.489	0.500	0.498	0.501	0.505	0.501
	± 0.078	± 0.014	± 0.018	± 0.021	± 0.019	± 0.022	± 0.053	± 0.009	± 0.022	± 0.014	± 0.019	± 0.010	± 0.034	± 0.009	± 0.011	± 0.013	± 0.024	± 0.019	± 0.009	± 0.017	± 0.501	± 0.015
Burden test	0.770	0.523	0.510	0.666	0.510	0.509	0.652	0.508	0.505	0.583	0.508	0.509	0.564	0.508	0.507	0.536	0.508	0.504	0.510	0.507	0.504	0.543
	± 0.013	± 0.013	± 0.014	± 0.011	± 0.009	± 0.007	± 0.008	± 0.011	± 0.012	± 0.012	± 0.007	± 0.014	± 0.017	± 0.010	± 0.020	± 0.012	± 0.016	± 0.016	± 0.010	± 0.012	± 0.013	± 0.070
Known motif b.	1.000	1.000	0.973	1.000	1.000	0.865	1.000	1.000	0.700	1.000	0.989	0.609	1.000	0.946	0.570	1.000	0.834	0.532	1.000	0.982	0.870	0.899
	± 0.000	± 0.000	± 0.004	± 0.000	± 0.000	± 0.004	± 0.000	± 0.000	± 0.020	± 0.000	± 0.002	± 0.017	± 0.000	± 0.010	± 0.024	± 0.000	± 0.016	± 0.020	± 0.000	± 0.004	± 0.982	± 0.158
Known motif c.	0.999	0.720	0.529	0.999	0.698	0.534	0.999	0.694	0.532	1.000	0.696	0.527	0.997	0.666	0.520	0.998	0.668	0.509	0.685	0.657	0.635	0.727
	± 0.001	± 0.014	± 0.020	± 0.001	± 0.013	± 0.017	± 0.001	± 0.012	± 0.012	± 0.001	± 0.018	± 0.018	± 0.002	± 0.010	± 0.009	± 0.002	± 0.012	± 0.013	± 0.018	± 0.007	± 0.657	± 0.189

Table A10: AUC estimates based on 5-fold CV for all 21 datasets in category "simulated immunosequencing data". The reported errors are standard deviations across the 5 cross-validation folds except for the last column "avg.", in which they show standard deviations across datasets. Wildcard characters in motifs are indicated by Z, characters with 50% probability of being removed by ^{d}, characters with 20% probability of behaving like a wildcard Z by ^{r}.

ID	0	1	2	3	4	avg.
motif freq. ρ	10%	1%	0.5%	0.1%	0.05%	–
implanted motif	$\mathtt{G^rS^rA^rF^r}$	$\mathtt{G^rS^rA^rF^r}$	$\mathtt{G^rS^rA^rF^r}$	$\mathtt{G^rS^rA^rF^r}$	$\mathtt{G^rS^rA^rF^r}$	–
DeepRC	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	**0.998** $\pm$ 0.002	**1.000** $\pm$ 0.001
SVM (MinMax)	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	0.999 $\pm$ 0.001	0.999 $\pm$ 0.002	0.985 $\pm$ 0.014	0.997 $\pm$ 0.007
SVM (Jaccard)	0.981 $\pm$ 0.041	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	0.904 $\pm$ 0.036	0.768 $\pm$ 0.068	0.931 $\pm$ 0.099
KNN (MinMax)	0.699 $\pm$ 0.272	0.717 $\pm$ 0.263	0.732 $\pm$ 0.263	0.536 $\pm$ 0.156	0.516 $\pm$ 0.153	0.640 $\pm$ 0.105
KNN (Jaccard)	0.698 $\pm$ 0.285	0.606 $\pm$ 0.237	0.523 $\pm$ 0.164	0.550 $\pm$ 0.186	0.539 $\pm$ 0.194	0.583 $\pm$ 0.071
Logistic Regression	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	0.697 $\pm$ 0.164	0.466 $\pm$ 0.103	0.833 $\pm$ 0.243
Logistic MIL (KMER)	0.997 $\pm$ 0.004	0.718 $\pm$ 0.112	0.637 $\pm$ 0.144	0.571 $\pm$ 0.146	0.528 $\pm$ 0.129	0.690 $\pm$ 0.186
Logistic MIL (TCRβ)	0.541 $\pm$ 0.086	0.566 $\pm$ 0.162	0.468 $\pm$ 0.086	0.505 $\pm$ 0.067	0.500 $\pm$ 0.121	0.516 $\pm$ 0.038
Burden test	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000	0.999 $\pm$ 0.003	0.792 $\pm$ 0.280	0.958 $\pm$ 0.093
Known motif b.	1.000 $\pm$ 0.000	1.000 $\pm$ 0.000	1.000 $\pm$ 0.000	0.999 $\pm$ 0.003	0.999 $\pm$ 0.003	1.000 $\pm$ 0.001
Known motif c.	1.000 $\pm$ 0.000	1.000 $\pm$ 0.000	0.989 $\pm$ 0.011	0.722 $\pm$ 0.085	0.626 $\pm$ 0.094	0.867 $\pm$ 0.180

Table A11: AUC estimates based on 5-fold CV for all 5 datasets in category "LSTM-generated data". The reported errors are standard deviations across the 5 cross-validation folds except for the last column "avg.", in which they show standard deviations across datasets. Characters affected by noise, as described in A4, paragraph "LSTM-generated data", are indicated by $\mathtt{^r}$.

	s.m. 1%	s.m. 0.1%	m.m. 1%	m.m. 0.1%	Avg.
DeepRC	**1.000** ± 0.000	**0.984** ± 0.008	0.999 ± 0.001	**0.938** ± 0.009	**0.980** ± 0.029
SVM (MinMax)	**1.000** ± 0.000	0.578 ± 0.020	**1.000** ± 0.000	0.531 ± 0.019	0.777 ± 0.258
SVM (Jaccard)	0.988 ± 0.003	0.527 ± 0.016	**1.000** ± 0.000	0.574 ± 0.019	0.772 ± 0.257
KNN (MinMax)	0.744 ± 0.237	0.486 ± 0.031	0.674 ± 0.182	0.500 ± 0.022	0.601 ± 0.128
KNN (Jaccard)	0.652 ± 0.155	0.484 ± 0.025	0.695 ± 0.200	0.508 ± 0.025	0.585 ± 0.104
Logistic Regression	**1.000** ± 0.000	0.585 ± 0.045	**1.000** ± 0.000	0.512 ± 0.015	0.774 ± 0.262
Logistic MIL (KMER)	0.541 ± 0.074	0.506 ± 0.034	0.994 ± 0.004	0.620 ± 0.153	0.665 ± 0.224
Logistic MIL (TCRβ)	0.503 ± 0.032	0.501 ± 0.016	0.992 ± 0.003	0.782 ± 0.030	0.695 ± 0.238
Burden test	**1.000** ± 0.000	0.640 ± 0.048	**1.000** ± 0.000	0.891 ± 0.016	0.883 ± 0.170
Known motif b.	1.000 ± 0.000	0.704 ± 0.028	0.994 ± 0.003	0.620 ± 0.038	0.830 ± 0.196
Known motif c.	0.920 ± 0.004	0.562 ± 0.028	0.647 ± 0.030	0.515 ± 0.031	0.661 ± 0.181

Table A12: AUC estimates based on 5-fold CV for all 4 datasets in category "real-world data with implanted signals". The reported errors are standard deviations across the 5 cross-validation folds except for the last column "avg.", in which they show standard deviations across datasets. **s.m. 1%:** In this dataset, a single motif with a frequency of 1% was implanted. **s.m. 0.1%:** In this dataset, a single motif with a frequency of 0.1% was implanted. **m.m. 1%:** In this dataset, multiple motifs with a frequency of 1% were implanted. **m.m. 0.1%:** In this dataset, multiple motifs with a frequency of 0.1% were implanted. A detailed description of the motifs is provided in section A4, paragraph "Real-world data with implanted signals.".

	AUC	F1 score	Balanced accuracy	Accuracy
DeepRC	**0.832** ± 0.022	**0.721** ± 0.030	**0.734** ± 0.032	0.735 ± 0.037
SVM (MinMax)	0.825 ± 0.022	0.680 ± 0.056	**0.734** ± 0.037	**0.742** ± 0.031
SVM (Jaccard)	0.546 ± 0.021	0.272 ± 0.184	0.523 ± 0.026	0.542 ± 0.032
KNN (MinMax)	0.679 ± 0.076	0.000 ± 0.000	0.500 ± 0.000	0.545 ± 0.044
KNN (Jaccard)	0.534 ± 0.039	0.073 ± 0.101	0.508 ± 0.012	0.551 ± 0.042
Logistic regression	0.613 ± 0.044	0.405 ± 0.211	0.558 ± 0.046	0.577 ± 0.058
Logistic MIL (KMER)	0.582 ± 0.065	0.118 ± 0.264	0.503 ± 0.007	0.515 ± 0.058
Logistic MIL (TCRβ)	0.515 ± 0.073	0.000 ± 0.000	0.496 ± 0.008	0.541 ± 0.039
Burden test	0.699 ± 0.041	-	-	-

Table A13: Results on the *CMV dataset* (real-world data) in terms of AUC, F1 score, balanced accuracy, and accuracy. For F1 score, balanced accuracy, and accuracy, all methods use their default thresholds. Each entry shows mean and standard deviation across 5 cross-validation folds.

A8 Repertoire generation via LSTM

We trained a conventional next-character LSTM model (Graves, 2013) based on the implementation in `https://github.com/spro/practical-pytorch` (access date 1st of May, 2020) using `PyTorch 1.3.1` (Paszke et al., 2019). For this, we applied an LSTM model with 100 LSTM blocks in 2 layers, which was trained for $5,000$ epochs using the Adam optimizer (Kingma & Ba, 2014) with learning rate 0.01, an input batch size of 100 character chunks, and a character chunk length of 200. As input we used the immuno-sequences in the `CDR3` column of the ***CMV dataset***, where we repeated sequences according to their counts in the repertoires, as specified in the `templates` column of the ***CMV dataset***. We excluded repertoires with unknown CMV status and unknown sequence abundance from training.

After training, we generated $1,000$ repertoires using a `temperature` value of 0.8. The number of sequences per repertoire was sampled from a Gaussian $\mathcal{N}(\mu = 285k, \sigma = 156k)$ distribution, where the whole repertoire was generated by the LSTM at once. That is, the LSTM can base the generation of the individual AA sequences in a repertoire, including the AAs and the lengths of the sequences, on the generated repertoire. A random immuno-sequence from the trained-on repertoires was used as initialization for the generation process. This immuno-sequence was not included in the generated repertoire.

Finally, we randomly assigned 500 of the generated repertoires to the positive (diseased) and 500 to the negative (healthy) class. We then implanted motifs in the positive class repertoires as described in section A4.2.

As illustrated in the comparison of histograms given in Fig. A2, the generated immuno-sequences exhibit a very similar distribution of 4-mers and AAs compared to the original ***CMV dataset***.

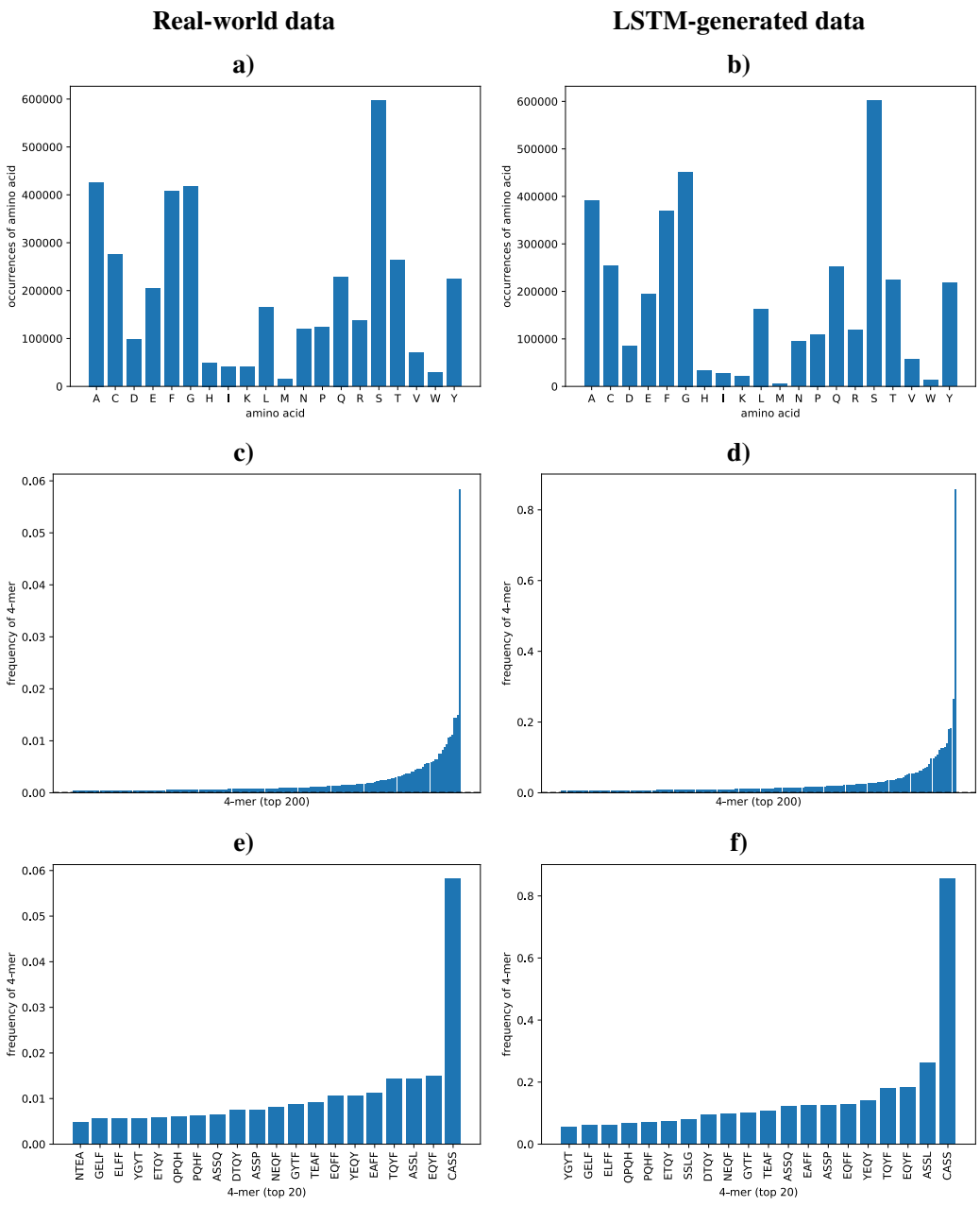

Figure A2: Distribution of AAs and k-mers in real-world *CMV dataset* and LSTM-generated data. **Left:** Histograms of real-world data. **Right:** Histograms of LSTM-generated data. **a)** Frequency of AAs in sequences of the *CMV dataset*. **b)** Frequency of AAs in sequences of the LSTM-generated datasets. **c)** Frequency of top 200 4-mers in sequences of the *CMV dataset*. **d)** Frequency of top 200 4-mers in sequences of the LSTM-generated datasets. **e)** Frequency of top 20 4-mers in sequences of the *CMV dataset*. **f)** Frequency of top 20 4-mers in sequences of the LSTM-generated datasets. Overall the distributions of AAs and 4-mers are similar in both datasets.

A9 Interpreting DeepRC

DeepRC allows for two forms of interpretability methods. (a) Due to its attention-based design, a trained model can be used to compute the attention weights of a sequence, which directly indicates its importance. (b) DeepRC furthermore allows for the usage of contribution analysis methods, such as Integrated Gradients (IG) (Sundararajan et al., 2017) or Layer-Wise Relevance Propagation (Montavon et al., 2018; Arras et al., 2019; Montavon et al., 2019; Preuer et al., 2019). We apply IG to identify the input patterns that are relevant for the classification. To identify AA patterns with high contributions in the input sequences, we apply IG to the AAs in the input sequences. Additionally, we apply IG to the kernels of the 1D CNN, which allows us to identify AA motifs with high contributions. In detail, we compute the IG contributions for the AAs and positional features in the kernels for every repertoire in the validation and test set, so as to exclude potential artifacts caused by over-fitting. Averaging the IG values over these repertoires then results in concise AA motifs. We include qualitative visual analyses of the IG method on different datasets below.
Here, we provide examples for the interpretation of trained DeepRC models using Integrated Gradients (IG) (Sundararajan et al., 2017) as contribution analysis method. The following illustrations were created using 50 IG steps, which we found sufficient to achieve stable IG results.
A visual analysis of DeepRC models on the simulated datasets, as illustrated in Tab. A14 and Fig. A3, shows that the implanted motifs can be successfully extracted from the trained model and are straightforward to interpret. In the real-world *CMV dataset*, DeepRC finds complex patterns with high variability in the center regions of the immuno-sequences, as illustrated in figure A4.

	Simulated			
extracted motif				
implanted motif(s)	SFEN	SF^{d}EN	SZZN	SZ^{d}ZN
motif freq. ρ	0.01%	0.01%	0.1%	0.1%

	LSTM-generated	**Real-world data with implanted signals**	
extracted motif			
implanted motif(s)	G^{r}S^{r}A^{r}F^{r}	L^{r}D^{r}R^{r}	{L^{r}D^{r}R^{r}; C^{r}A^{r}S; G^{r}L-N}
motif freq. ρ	0.05%	0.1%	0.1%

Table A14: Visualization of motifs extracted from trained DeepRC models for datasets from categories "simulated immunosequencing data", "LSTM-generated data", and "real-world data with implanted signals". Motif extraction was performed using Integrated Gradients on the 1D CNN kernels over the validation set and test set repertoires of one CV fold. Wildcard characters are indicated by Z, random noise on characters by ^{r}, characters with 50% probability of being removed by ^{d}, and gap locations of random lengths of $\{0; 1; 2\}$ by -. Larger characters in the extracted motifs indicate higher contribution, with blue indicating positive contribution and red indicating negative contribution towards the prediction of the diseased class. Contributions to positional encoding are indicated by $<$ (beginning of sequence), $\wedge$ (center of sequence), and $>$ (end of sequence). Only kernels with relatively high contributions are shown, i.e. with contributions roughly greater than the average contribution of all kernels.

Figure A3: Integrated Gradients applied to input sequences of positive class repertoires. Three sequences with the highest contributions to the prediction of their respective repertoires are shown. **a)** Input sequence taken from "simulated immunosequencing data" with implanted motif $\texttt{SZ}^{\texttt{d}}\texttt{Z}^{\texttt{d}}\texttt{N}$ and motif implantation probability 0.1%. The DeepRC model reacts to the S and N at the 5^{th} and 8^{th} sequence position, thereby identifying the implanted motif in this sequence. **b)** and **c)** Input sequence taken from "real-world data with implanted signals" with implanted motifs $\{\texttt{L}^{\texttt{r}}\texttt{D}^{\texttt{r}}\texttt{R}^{\texttt{r}}; \texttt{C}^{\texttt{r}}\texttt{A}^{\texttt{r}}\texttt{S}; \texttt{G}^{\texttt{r}}\texttt{L-N}\}$ and motif implantation probability 0.1%. The DeepRC model reacts to the fully implanted motif CAS (b) and to the partly implanted motif AAs C and A at the 5^{th} and 7^{th} sequence position (c), thereby identifying the implanted motif in the sequences. Wildcard characters in implanted motifs are indicated by Z, characters with 50% probability of being removed by $^{\text{d}}$, and gap locations of random lengths of $\{0; 1; 2\}$ by -. Larger characters in the sequences indicate higher contribution, with blue indicating positive contribution and red indicating negative contribution towards the prediction of the diseased class.

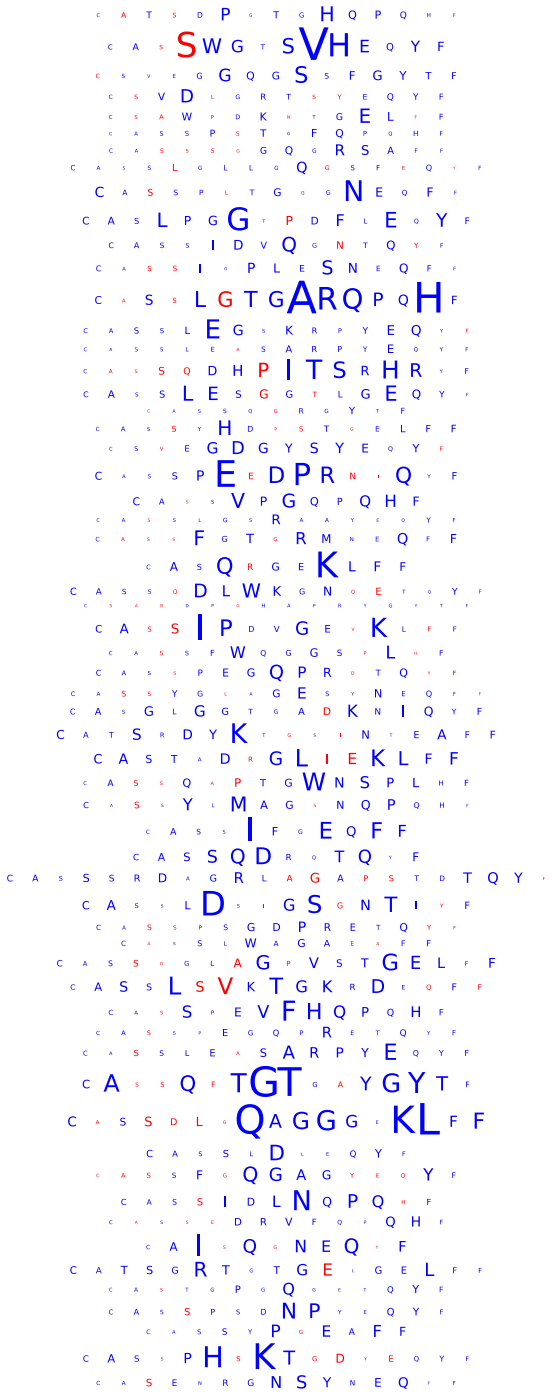

Figure A4: Visualization of the contributions of characters within a sequence via IG. Each sequence was selected from a different repertoire and showed the highest contribution in its repertoire. The model was trained on *CMV dataset*, using a kernel size of 9, 32 kernels and 137 repertoires for early stopping. Larger characters in the extracted motifs indicate higher contribution, with blue indicating positive contribution and red indicating negative contribution towards the prediction of the disease class.

A10 Attention values for previously associated CMV sequences

index	sequence	attention	quantile	index	sequence	attention	quantile	index	sequence	attention	quantile	index	sequence	attention	quantile
1	CASSGQGAYEQYF	1.000	0.999	42	CASSLGGAGDTQYF	1.000	1.000	83	CASSYVRTGGNYGYTF	0.967	0.932	124	CASSLTGGNSGNTIYF	0.991	0.977
2	CASSIGPLEHNEQFF	0.947	0.900	43	CASNRDRGRYEQYF	0.991	0.978	84	CASSLAGVDYEQYF	0.999	0.996	125	CASSRNRGQETQYF	0.978	0.952
3	CASSPDRVGQETQYF	0.995	0.987	44	CSVRDNHNQPQHF	0.965	0.929	85	CASSLGAGNQPQHF	1.000	0.999	126	CASSLGQGLAEAFF	0.996	0.989
4	CASSLEAEYEQYF	0.992	0.980	45	CASSAQGAYEQYF	0.998	0.995	86	CASSRDRNYGYTF	0.998	0.995	127	CASRTGESGYTF	0.985	0.965
5	CASSIEGNQPQHF	0.993	0.983	46	CATSRGTVSYEQYF	0.990	0.975	87	CASGRDTYEQYF	0.999	0.997	128	CASSSDSGGTDTQYF	0.951	0.906
6	CATSDGDEQFF	0.998	0.996	47	CASSPPSGLTDTQYF	0.978	0.951	88	CAWSVSDLAKNIQYF	0.954	0.911	129	CASSVDGGRGTEAFF	0.995	0.987
7	CASSLVAGGRETQYF	0.988	0.971	48	CASSGDRLYEQYF	0.998	0.994	89	CASSPNQETQYF	0.999	0.996	130	CSVEVRGTDTQYF	0.955	0.912
8	CASSRGRQETQYF	0.997	0.993	49	CASSLNRGQETQYF	0.996	0.988	90	CSASDHEQYF	0.995	0.986	131	CASSESGDPSSYEQYF	0.980	0.955
9	CASSAGQGVTYEQYF	0.998	0.995	50	CASSLGVGPYNEQFF	0.986	0.967	91	CASSWDRDNSPLHF	0.918	0.855	132	CASSEEAGGSGYTF	0.982	0.959
10	CASSQNRGQETQYF	0.995	0.987	51	CATSDSVTNTGELFF	0.989	0.973	92	CASSPGQEAGANVLTF	0.823	0.728	133	CAISESQDRGHEQYF	0.823	0.728
11	CASSPQRNTEAFF	1.000	0.999	52	CASSRNRESNQPQHF	0.968	0.934	93	CASSLVAAGRETQYF	0.959	0.919	134	CASSPTGGELFF	0.989	0.974
12	CASSLAPGATNEKLFF	0.976	0.949	53	CASSEARTRAFF	0.927	0.869	94	CASSPHRNTEAFF	0.999	0.998	135	CASSVETGGTEAFF	0.995	0.986
13	CASSLIGVSSYNEQFF	0.983	0.961	54	CASSYNPYSNQPQHF	0.892	0.819	95	CASRGQGWDEKLFF	0.994	0.984	136	CASASANYGYTF	0.816	0.720
14	CSVRDNFNQPQHF	0.915	0.851	55	CASSLGHRDSSYEQYF	0.987	0.969	96	CASSQVETDTQYF	0.994	0.984	137	CASSSRTGEETQYF	0.996	0.988
15	CASSQTGGRNQPQHF	0.997	0.992	56	CASSRLAASTDTQYF	0.992	0.979	97	CASRDWDYTDTQYF	0.994	0.984	138	CASSLGRGYEKLFF	0.985	0.965
16	CASSLVIGGDTEAFF	0.966	0.931	57	CASSVTGGTDTQYF	1.000	0.999	98	CASSSDRVGQETQYF	0.980	0.955	139	CASSGLNEQFF	0.994	0.984
17	CASSLRREKLFF	0.998	0.993	58	CASSPPGQGSDTQYF	0.975	0.946	99	CASSLGDRPDTQYF	0.940	0.889	140	CASSRNRAQETQYF	0.994	0.984
18	CASSFHGFNQPQHF	0.991	0.978	59	CATSDSRTGGQETQYF	0.900	0.829	100	CASSLEGQGFGYTF	0.944	0.895	141	CASTPGDEQFF	0.988	0.971
19	CATSRDTQGSYGYTF	0.917	0.854	60	CASSSPGRSGANVLTF	0.995	0.986	101	CASSSGQVYGYTF	0.999	0.996	142	CASSLGIDTQYF	0.997	0.991
20	CASSRLAGGTDTQYF	0.999	0.998	61	CASSPLSDTQYF	0.998	0.994	102	CASSEEGIQPQHF	0.998	0.994	143	CASSIRTNYYGYTF	0.996	0.990
21	CASSFPTSGQETQYF	0.982	0.959	62	CASSLTGGRNQPQHF	0.999	0.997	103	CASSLETYGYTF	0.998	0.995	144	CASSPISNEQFF	0.967	0.933
22	CASSPGDEQYF	0.998	0.993	63	CASSIQGYSNQPQHF	0.993	0.983	104	CASSFPGGETQYF	0.992	0.979	145	CASSQNRAQETQYF	0.984	0.962
23	CASSLPSGLTDTQYF	0.994	0.985	64	CASSTTGGDGYTF	0.978	0.952	105	CASSSGQVQETQYF	0.997	0.993	146	CASSALGGAGTGELFF	0.985	0.964
24	CASSEIPNTEAFF	0.997	0.992	65	CASSVLAGPTDTQYF	0.951	0.906	106	CASSEGARQPQHF	0.999	0.998	147	CASSLAVLPTDTQYF	0.996	0.989
25	CASSIWGLDTEAFF	0.959	0.919	66	CASSHRDRNYEQYF	0.987	0.969	107	CSALGHSNQPQHF	0.926	0.867	148	CASSLQAGANEQFF	0.969	0.935
26	CASSPGDEQFF	0.999	0.997	67	CASSPSRNTEAFF	0.999	0.998	108	CASSLLWDQPQHF	0.986	0.967	149	CASSTGGAQPQHF	0.998	0.993
27	CATSRDSQGSYGYTF	0.980	0.955	68	CASSLGGPGDTQYF	0.993	0.982	109	CASSLVGDGYTF	1.000	1.000	150	CASSLGASGSRTDTQYF	0.932	0.876
28	CASSYGGLGSYEQYF	0.995	0.987	69	CASSEARGGVEKLFF	0.989	0.974	110	CASSSRGTGELFF	0.999	0.997	151	CASSRGTGATDTQYF	0.999	0.998
29	CASSPSTGTEAFF	0.997	0.992	70	CASSTGTSGSYEQYF	0.999	0.998	111	CATSRVAGETQYF	0.980	0.955	152	CASSYPGETQYF	0.997	0.992
30	CSVEEDEGIYGYTF	0.964	0.927	71	CASRSDSGANVLTF	0.973	0.942	112	CASRGQGAGELFF	0.987	0.969	153	CASSLTDTGELFF	0.994	0.984
31	CASSPAGLNTEAFF	0.996	0.988	72	CASSLEAENEQFF	0.973	0.943	113	CASSPGGTQYF	0.999	0.996	154	CASRPQGNYGYTF	0.998	0.996
32	CASSLGLKGTQYF	0.964	0.928	73	CASSEAPSTSTDTQYF	0.989	0.973	114	CASSLQGINQPQHF	0.999	0.997	155	CASSTSGNTIYF	1.000	0.999
33	CASMGGASYEQYF	0.991	0.978	74	CASSLQGADTQYF	0.997	0.991	115	CASSQGRHTDTQYF	0.960	0.921	156	CASSSGTGDEQYF	1.000	1.000
34	CASSQVPGQGDNEQFF	0.983	0.961	75	CASSLEGQQPQHF	0.994	0.984	116	CASSPRWQETQYF	0.991	0.978	157	CASSPPAGTNYGYTF	0.947	0.900
35	CATSDGDTQYF	0.996	0.989	76	CASSYGGEGYTF	0.999	0.996	117	CASRDRDRVNTEAFF	0.970	0.938	158	CASSPLGGTTEAFF	0.995	0.988
36	CATSDGETQYF	0.998	0.994	77	CASSLRGSSYNEQFF	0.999	0.998	118	CASSWDRGTEAFF	0.999	0.999	159	CASSLGWTEAFF	0.999	0.997
37	CSVRDNYNQPQHF	0.998	0.993	78	CASSISAGEAFF	0.992	0.979	119	CASSRPGQGNTEAFF	0.994	0.984	160	CATSREGSGYEQYF	0.987	0.969
38	CASSLVASGRETQYF	0.997	0.991	79	CASRPTGYEQYF	0.987	0.969	120	CASSPGSGANVLTF	0.999	0.997	161	CASSYAGDGYTF	0.992	0.980
39	CSASPGQGASYGYTF	0.987	0.969	80	CAWRGTGNSPLHF	0.964	0.927	121	CASRRGSSYEQYF	0.999	0.998	162	CASSDRGNTGELFF	0.995	0.986
40	CASSESGHRNQPQHF	0.999	0.997	81	CASSLGDRAYNEQFF	0.996	0.988	122	CASRTDSGANVLTF	0.994	0.986	163	CSARRGPGELFF	0.839	0.749
41	CASSLGHRDPNTGELFF	0.981	0.958	82	CASSLQGYSNQPQHF	1.000	0.999	123	CASSQDPRGTEAFF	0.950	0.905	164	CASSQGLQETQYF	0.996	0.990

Table A15: TCRβ sequences that had been discovered by Emerson et al. (2017) with their associated attention values by DeepRC. These sequences have significantly (p-value 1.3e-93) higher attention values than other sequences. The column "quantile" provides the quantile values of the empiricial distribution of attention values across all sequences in the dataset.

A11 DeepRC variations and ablation study

In this section we investigate the impact of different variations of DeepRC on the performance on the *CMV dataset*. We consider both a CNN-based sequence embedding, as used in the main paper, and an LSTM-based sequence embedding. In both cases we vary the number of attention heads and the β parameter for the softmax function the attention mechanism (see Eq. 2 in main paper). For the CNN-based sequence embedding we also vary the number of CNN kernels and the kernel sizes used in the 1D CNN. For the LSTM-based sequence embedding we use one one-directional LSTM layer, where the output values at the last sequence position (without padding) are taken as embedding of the sequence. Here we vary the number of LSTM blocks in the LSTM layer. To counter over-fitting due to the increased complexity of these DeepRC variations, we added a $l2$ weight penalty to the training loss. The factor with which the $l2$ weight penalty contributes to the training loss is varied over 3 orders of magnitudes, where suitable value ranges were manually determined on one of the training folds beforehand.
To reduce the computational cost, we do not consider all numbers of kernels that were considered in the main paper. Furthermore, we only compute the AUC scores on 3 of the 5 cross-validation folds. The hyperparameters, which were used in a grid search procedure, are listed in Tab. A16 for the CNN-based sequence embedding and Tab. A17 for the LSTM-based sequence embedding.

Results. We show performance in terms of AUC score with single hyperparameters set to fixed values so as to investigate their influence in Tab. A19 for the CNN-based sequence embedding and Tab. A18 for the LSTM-based sequence embedding. We note that due to restricted computational resources this study was conducted with fewer different numbers of CNN kernels, with the AUC estimated from only 3 of the 5 cross-validation folds, which leads to a slight decrease of performance in comparison to the full hyperparameter search and cross-validation procedure used in the main paper. As can be seen in Tab. A19 and A18, the LSTM-based sequence embedding generalizes slightly better than the CNN-based sequence embedding. The performance of DeepRC, however, remains rather robust w.r.t. the different hyperparameter settings.

learning rate	10^{-4}
number of attention heads	$\{1; 16; 64\}$
β of attention softmax	$\{0.1; 1.0; 10.0\}$
$l2$ weight penalty	$\{1.0; 0.1; 0.01\}$
number of kernels	$\{8; 32; 128\}$
number of CNN layers	$\{1\}$
number of layers in key-NN	$\{2\}$
number of units in key-NN	$\{32\}$
kernel size	$\{5; 7; 9\}$
subsampled seqences	$10,000$
batch size	4

Table A16: Hyperparameter search space for DeepRC variations with CNN-based sequence embedding. Every $5 \cdot 10^3$ updates, the current model was evaluated against the validation fold. The early stopping hyperparameter was determined by selecting the model with the best loss on the validation fold after $2 \cdot 10^5$ updates.

learning rate	10^{-4}
number of attention heads	$\{1; 16; 64\}$
β of attention softmax	$\{0.1; 1.0; 10.0\}$
$l2$ weight penalty	$\{0.01; 0.001; 0.0001\}$
number of LSTM blocks	$\{8; 32; 128\}$
number of CNN layers	$\{1\}$
number of layers in key-NN	$\{2\}$
number of units in key-NN	$\{32\}$
subsampled seqences	$10,000$
batch size	4

Table A17: Hyperparameter search space for DeepRC variations with LSTM-based sequence embedding. Every $5 \cdot 10^3$ updates, the current model was evaluated against the validation fold. The early stopping hyperparameter was determined by selecting the model with the best loss on the validation fold after $2 \cdot 10^5$ updates.

Fixed parameter	**Test set**		**Validation set**		**Training set**	
	mean	std	mean	std	mean	std
beta=0.1	0.827	± 0.02	0.846	± 0.033	0.976	± 0.015
beta=1.0	0.82	± 0.012	0.853	± 0.031	0.979	± 0.016
beta=10.0	0.823	± 0.014	0.858	± 0.033	0.934	± 0.026
heads=1	0.838	± 0.033	0.856	± 0.029	0.966	± 0.012
heads=16	0.817	± 0.015	0.853	± 0.028	0.972	± 0.026
heads=64	0.823	± 0.014	0.858	± 0.033	0.934	± 0.026
lstms=8	0.818	± 0.011	0.837	± 0.025	0.881	± 0.013
lstms=32	0.814	± 0.015	0.853	± 0.029	0.948	± 0.033
lstms=128	0.818	± 0.018	0.859	± 0.032	0.943	± 0.028

Table A18: Impact of hyperparameters on DeepRC with LSTM for sequence encoding. Mean ("mean") and standard deviation ("std") for the area under the ROC curve over the first 3 folds of a 5-fold nested cross-validation for different sub-sets of hyperparameters ("sub-set") are shown. The following sub-sets were considered: "full": Full grid search over hyperparameters; "beta=*": Grid search over hyperparameters with reduction to specific value $*$ of beta value of attention softmax; "heads=*": Grid search over hyperparameters with reduction to specific number $*$ of attention heads; "lstms=*": Grid search over hyperparameters with reduction to specific number $*$ of LSTM blocks for sequence embedding.

Fixed parameter	Test set		Validation set		Training set	
	mean	std	mean	std	mean	std
beta=0.1	0.833	± 0.031	0.86	± 0.025	0.94	± 0.018
beta=1.0	0.799	± 0.007	0.873	± 0.017	0.954	± 0.005
beta=10.0	0.817	± 0.02	0.87	± 0.022	0.962	± 0.034
heads=1	0.822	± 0.036	0.869	± 0.022	0.943	± 0.032
heads=16	0.808	± 0.01	0.871	± 0.025	0.965	± 0.019
heads=64	0.796	± 0.039	0.864	± 0.018	0.927	± 0.024
ksize=5	0.822	± 0.036	0.866	± 0.021	0.926	± 0.026
ksize=7	0.817	± 0.02	0.87	± 0.022	0.962	± 0.034
ksize=9	0.821	± 0.016	0.869	± 0.025	0.95	± 0.031
kernels=8	0.825	± 0.024	0.86	± 0.027	0.928	± 0.019
kernels=32	0.801	± 0.001	0.877	± 0.018	0.974	± 0.017
kernels=128	0.824	± 0.027	0.864	± 0.023	0.931	± 0.062

Table A19: Impact of hyperparameters on DeepRC with 1D CNN for sequence encoding. Mean ("mean") and standard deviation ("std") for the area under the ROC curve over the first 3 folds of a 5-fold nested cross-validation for different sub-sets of hyperparameters ("sub-set") are shown. The following sub-sets were considered: "full": Full grid search over hyperparameters; "beta=*": Grid search over hyperparameters with reduction to specific value $*$ of beta value of attention softmax; "heads=*": Grid search over hyperparameters with reduction to specific number $*$ of attention heads; "ksize=*": Grid search over hyperparameters with reduction to specific kernel size $*$ of 1D CNN kernels for sequence embedding; "kernels=*": Grid search over hyperparameters with reduction to specific number $*$ of 1D CNN kernels for sequence embedding.

References

Akbar, R., Robert, P. A., Pavlović, M., Jeliazkov, J. R., Snapkov, I., Slabodkin, A., Weber, C. R., Scheffer, L., Miho, E., Haff, I. H., et al. A compact vocabulary of paratope-epitope interactions enables predictability of antibody-antigen binding. *bioRxiv*, 2019.

Arras, L., Arjona-Medina, J., Widrich, M., Montavon, G., Gillhofer, M., Müller, K.-R., Hochreiter, S., and Samek, W. Explaining and interpreting LSTMs. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pp. 211–238. Springer, 2019.

Atchley, W. R., Zhao, J., Fernandes, A. D., and Drüke, T. Solving the protein sequence metric problem. *Proceedings of the National Academy of Sciences*, 102(18):6395–6400, 2005.

Briggs, F., Fern, X. Z., and Raich, R. Rank-loss support instance machines for miml instance annotation. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 534–542, 2012.

Cortes, C. and Vapnik, V. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Emerson, R. O., DeWitt, W. S., Vignali, M., Gravley, J., Hu, J. K., Osborne, E. J., Desmarais, C., Klinger, M., Carlson, C. S., Hansen, J. A., et al. Immunosequencing identifies signatures of cytomegalovirus exposure history and HLA-mediated effects on the T cell repertoire. *Nature Genetics*, 49(5):659, 2017.

Gelasca, E. D., Byun, J., Obara, B., and Manjunath, B. Evaluation and benchmark for biological image segmentation. In *2008 15th IEEE International Conference on Image Processing*, pp. 1816–1819. IEEE, 2008.

Graves, A. Generating sequences with recurrent neural networks. *ArXiv*, 2013.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. *ArXiv*, 2012.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hu, B., Lu, Z., Li, H., and Chen, Q. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, pp. 2042–2050, 2014.

Ilse, M., Tomczak, J. M., and Welling, M. Attention-based deep multiple instance learning. *International Conference on Machine Learning (ICML)*, 2018.

Kimeswenger, S., Rumetshofer, E., Hofmarcher, M., Tschandl, P., Kittler, H., Hochreiter, S., Hötzenecker, W., and Klambauer, G. Detecting cutaneous basal cell carcinomas in ultra-high resolution and weakly labelled histopathological images. *ArXiv*, 2019.

Kingma, D. P. and Ba, J. Adam: a method for stochastic optimization. *ArXiv*, 2014.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems*, pp. 971–980, 2017.

Lefranc, M.-P., Pommié, C., Ruiz, M., Giudicelli, V., Foulquier, E., Truong, L., Thouvenin-Contet, V., and Lefranc, G. Imgt unique numbering for immunoglobulin and t cell receptor variable domains and ig superfamily v-like domains. *Developmental & Comparative Immunology*, 27(1):55–77, 2003.

Levandowsky, M. and Winter, D. Distance between sets. *Nature*, 234(5323):34–35, 1971.

Li, B. and Leal, S. M. Methods for detecting associations with rare variants for common diseases: application to analysis of sequence data. *The American Journal of Human Genetics*, 83(3):311–321, 2008.

Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z., and Matthews, I. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *2010 ieee computer society conference on computer vision and pattern recognition-workshops*, pp. 94–101. IEEE, 2010.

Montavon, G., Samek, W., and Müller, K.-R. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.

Montavon, G., Binder, A., Lapuschkin, S., Samek, W., and Müller, K.-R. Layer-wise relevance propagation: an overview. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pp. 193–209. Springer, 2019.

Ostmeyer, J., Christley, S., Toby, I. T., and Cowell, L. G. Biophysicochemical motifs in T-cell receptor sequences distinguish repertoires from tumor-infiltrating lymphocyte and adjacent healthy tissue. *Cancer Research*, 79(7):1671–1680, 2019.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: an imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

Preuer, K., Klambauer, G., Rippmann, F., Hochreiter, S., and Unterthiner, T. Interpretable deep learning in drug discovery. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pp. 331–345. Springer, 2019.

Ralaivola, L., Swamidass, S. J., Saigo, H., and Baldi, P. Graph kernels for chemical informatics. *Neural Networks*, 18(8):1093–1110, 2005.

Ruiz, A. T., Thiam, P., Schwenker, F., Palm, Günther", e. L., Schwenker, F., and Trentin, E. A $$k$$-nearest neighbor based algorithm for multi-instance multi-label active learning. In *Artificial Neural Networks in Pattern Recognition*, pp. 139–151, Cham, 2018. Springer International Publishing.

Sebastiani, F. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3319–3328. JMLR, 2017.

Uriot, T. Learning with sets in multiple instance regression applied to remote sensing. *ArXiv*, 2019.

Weber, C. R., Akbar, R., Yermanos, A., Pavlović, M., Snapkov, I., Sandve, G. K., Reddy, S. T., and Greiff, V. immuneSIM: tunable multi-feature simulation of B- and T-cell receptor repertoires for immunoinformatics benchmarking. *Bioinformatics*, 03 2020.

Wu, J.-S., Huang, S.-J., and Zhou, Z.-H. Genome-wide protein function prediction through multi-instance multi-label learning. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 11(5):891–902, 2014.

Wu, M. C., Lee, S., Cai, T., Li, Y., Boehnke, M., and Lin, X. Rare-variant association testing for sequencing data with the sequence kernel association test. *The American Journal of Human Genetics*, 89(1):82–93, 2011.

Zhang, Z.-L. and Zhang, M.-L. Multi-instance multi-label learning with application to scene classification. In *Advances in neural information processing systems*, pp. 1609–1616, 2007.