DeepRC: Immune repertoire classification with attention-based deep massive multiple instance learning

Michael Widrich [1 2] **Bernhard Schäfl** [1 2] **Milena Pavlović** [3 4] **Geir Kjetil Sandve** [4] **Sepp Hochreiter** [2 1 5]
Victor Greiff [3] **Günter Klambauer** [2 1]

Abstract

High-throughput immunosequencing allows reconstructing the immune repertoire of an individual, which is an exceptional opportunity for new immunotherapies, immunodiagnostics, and vaccine design. Such immune repertoires are shaped by past and current immune events, for example infection and disease, and thus record an individual's state of health. Consequently, immune repertoire sequencing data may enable the prediction of health and disease using machine learning. However, finding the connections between an individual's repertoire and the individual's disease class, with potentially hundreds of thousands to millions of short sequences per individual, poses a difficult and unique challenge for machine learning methods. In this work, we present our method DeepRC that combines a Deep Learning architecture with attention-based multiple instance learning. To validate that DeepRC accurately predicts an individual's disease class based on its immune repertoire and determines the associated class-specific sequence motifs, we applied DeepRC in four large-scale experiments encompassing ground-truth simulated as well as real-world virus infection data. We demonstrate that DeepRC outperforms all tested methods with respect to predictive performance and enables the extraction of those sequence motifs that are connected to a given disease class.

[1]Institute for Machine Learning, Johannes Kepler University Linz, Austria [2]LIT AI Lab, Johannes Kepler University Linz, Austria [3]Department of Immunology, University of Oslo, Oslo, Norway [4]Department of Informatics, University of Oslo, Oslo, Norway [5]Institute of Advanced Research in Artificial Intelligence (IARAI). Correspondence to: Günter Klambauer <klambauer@ml.jku.at>.

Pre-print.

1. Introduction

Immune receptors enable the immune system to specifically recognize disease agents or pathogens. Each immune encounter is recorded as an immune event into immune memory by preserving and amplifying immune receptors in the repertoire used to fight a given disease. This is, for example, the principle of vaccination. Each human has about 10^7–10^8 unique immune receptors with low overlap across individuals and sampled from a potential diversity of $> 10^{14}$ receptors (Mora & Walczak, 2019). The ability to sequence and analyze human immune receptors at large scale has led to fundamental and mechanistic insight into the adaptive immune system and has also opened the opportunity for the development of novel diagnostics and therapy approaches (Georgiou et al., 2014; Brown et al., 2019).

Each individual is uniquely characterized by their immune repertoire, which is acquired and changed during life. This repertoire may be influenced by all diseases that an individual is exposed to during their lives and hence contains highly valuable information about those diseases. As a prominent example, Emerson et al. (2017) were able to discriminate between cytomegalovirus (CMV) positive and negative individuals based solely on the presence of CMV-associated immune receptor T-cell receptor (TCR) sequences and could identify receptor sequences associated with the immune status. In a similar approach, Ostmeyer et al. (2019) have identified motifs in TCR sequence repertoires that distinguish between tumor-infiltrating lymphocyte and adjacent healthy tissues. Those studies established that the sequenced immune repertoire is a highly complex data type, which can be tackled by a large variety of computational methods. It is envisioned that the analysis of such immune repertoires at large scale could bring insights into diseases that are currently difficult to treat, such as autoimmune diseases (Bashford-Rogers et al., 2019; Liu et al., 2019).

Immune repertoire classification, i.e. classifying the immune status based on the immune repertoire sequences, is essentially a text-book example for a *multiple instance learning* problem (Dietterich et al., 1997; Maron & Lozano-Pérez, 1998; Wang et al., 2018). Briefly, the immune repertoire of an individual consists of an immensely large bag of immune

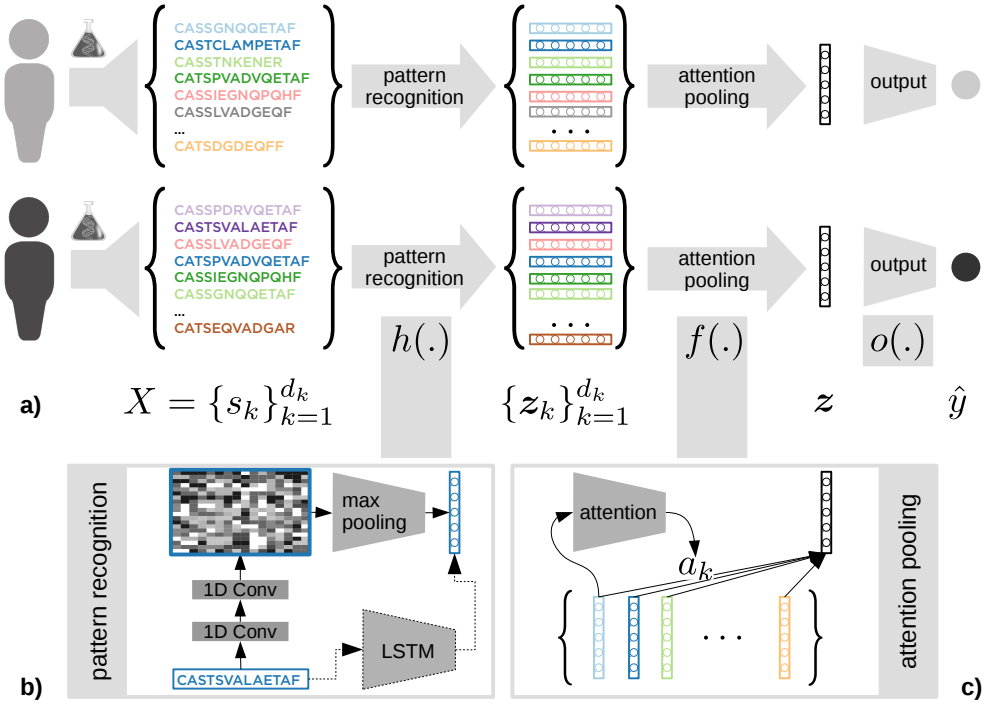


Figure 1. Schematic representation of the DeepRC approach. **a)** An immune repertoire X is represented by large bags of immune receptor sequences (colored). A neural network (NN) h serves to recognize patterns in each of the sequences s_k and maps them to sequence-representations $\boldsymbol{z}_k$. A pooling function f is used to obtain a repertoire-representation $\boldsymbol{z}$ for the input object. Finally, an output network o predicts the class label $\hat{y}$. **b)** DeepRC uses stacked 1D convolutions for a parameterized function h due to their computational efficiency. Potentially, millions of sequences have to be processed for each input object. In principle, also RNNs, such as LSTMs (Hochreiter et al., 2007), or Transformer networks (Vaswani et al., 2017c) may be used but are currently computationally too costly. **c)** Attention-pooling is used to obtain a repertoire-representation $\boldsymbol{z}$ for each input object, where DeepRC uses weighted averages of sequence-representations. The weight values are predicted by a Transformer-like attention mechanism.

receptors, represented as biological sequences. Usually the presence of only a small fraction of particular receptor determines the immune status with respect to a particular disease (Christophersen et al., 2014; Emerson et al., 2017). This situation is exactly described by multiple instance learning (MIL), in which the presence of a single instance in a bag can determine the class of the whole bag. Attention-based MIL has been successfully used for image data, for example to identify tiny objects in large images (Ilse et al., 2018; Pawlowski et al., 2019; Tomita et al., 2019; Kimeswenger et al., 2019). MIL problems are typically the more difficult, the larger the bag of instances and the fewer the instances that cause the label are. Therefore, classification of immune repertoires bears a high difficulty since each immune repertoire can contain millions of sequences as instances and only few indicate the class. Further properties of the data that complicate the problem are (a) the overlap of immune repertoires of different individuals is low (in most cases, maximally low single-digit percentage values) (Greiff et al., 2017; Elhanati et al., 2018), (b) multiple different sequences can bind to the same pathogen (Wucherpfennig et al., 2007), and (c) only subsequences within the sequences determines whether binding to an pathogen is possible (Dash et al., 2017; Glanville et al., 2017; Akbar et al., 2019; Springer et al., 2020; Fischer et al., 2019). In summary, immune repertoire classification can be formulated as multiple instance learning with sparse signals and large numbers of instances, which represents a challenge for currently available machine learning methods. Additionally, ideally, methods should be interpretable, since the extraction of class-associated sequence motifs is desired to gain crucial biological insight.

In this work, we contribute the following:

- We propose a deep attention-based multiple-instance learning method for large bags of complex sequences, as they occur in immune-repertoire classification.
- We conduct a large comparative study of machine learning approaches for the classification of immune repertoires.

- We suggest an interpretation method for our approach to identify those sequence motifs that are associated with a given class.

2. Related work

Immunosequencing data has been analyzed with computational methods for a variety of different tasks (Greiff et al., 2015; Shugay et al., 2015; Miho et al., 2018; Yaari & Kleinstein, 2015; Wardemann & Busse, 2017). A large part of the currently available machine learning methods for immune receptor data have focused on the individual immune receptors in a repertoire, with the aim to, for example, predict the antigen or antigen portion (epitope) to which these sequences bind or to predict sharing of receptors across individuals (Gielis et al., 2019; Springer et al., 2020; Jurtz et al., 2018; Moris et al., 2019; Fischer et al., 2019; Greiff et al., 2017; Sidhom et al., 2019; Elhanati et al., 2018). Recently, Jurtz et al. (2018) used 1D convolutional neural networks (CNNs) to predict antigen binding of TCR sequences (specifically, binding of TCR sequences to peptide-MHC complexes) and demonstrated that motifs can be extracted from these models. This result has motivated our approach to use 1D CNNs at the sequence level. Similarly, Konishi et al. (2019) use CNNs, Gradient Boosting, and other ML techniques on B-cell receptor (BCR) sequences to classify tumor tissue from normal tissue. However, the methods presented so far are working on individual sequences, which is a supervised machine learning task, in which a particular class, the epitope, has to be predicted based on a single input sequence.

Immune repertoire classification has been considered as multiple instance learning (MIL) problem in the following works. A Deep Learning framework called DeepTCR (Sidhom et al., 2019) implements several Deep Learning approaches for immunosequencing data. The computational framework, *inter alia*, allows for attention-based MIL repertoire classifiers and implements a basic form of attention-based averaging. Ostmeyer et al. (2019) already suggested a MIL method for immune repertoire classification. This method considers 4-mers, fixed sub-sequences of length 4, as instances of an input object and trained a logistic regression model with these 4-mers as input. The predictions of the logistic regression model for each 4-mer were max-pooled to obtain one prediction per input object. This approach is characterized by (a) the rigidity of the k-mer features as compared to convolutional kernels (Alipanahi et al., 2015; Zhou & Troyanskaya, 2015; Zeng et al., 2016), (b) the max-pooling operation, which constrains the network to learn from a single, top-ranked k-mer for each iteration over the input object, and (c) the pooling of predictions scores rather than representations (Wang et al., 2018). Our experiments also support that these choices in the design of the method can lead to constraints on the predictive performance (see Section 5).

Our DeepRC method also uses a multiple-instance learning approach but considers sequences rather than k-mers as instances within an input object and uses a transformer-like attention mechanism. DeepRC sets out to avoid the the above-mentioned constraints of current methods by (a) using 1D convolutions as feature extractors, (b) applying transformer-like attention-based pooling instead of max-pooling and learning a classifier on the repertoire-representation rather than a classifier on the sequence-representation, and (c) pooling learned representations rather than predictions.

3. Problem setting and notation

We consider a multiple-instance learning (MIL) problem, in which an input object X is a *bag* of K instances $X = \{s_1, \ldots, s_{d_k}\}$, which do not have dependencies nor orderings between them and K can be different for every object. We assume that each instance s_k is associated with a label $y_k \in \{0, 1\}$, assuming a binary classification task, to which we do not have access. We only have access to a label $Y = \max_k y_k$ for an input object or bag. Note that this poses a credit assignment problem, since the sequences that are responsible for the label Y have to be identified.

A model $\hat{y} = g(X)$ should be (a) invariant to permutations of the instances and (b) able to cope with the fact that K varies across input objects (Ilse et al., 2018), which is a problem also posed by point sets (Qi et al., 2017). Two principled approaches exist. The first approach is to learn an instance-level scoring function $h : \mathcal{S} \mapsto [0, 1]$, which is then pooled across instances with a pooling function f, for example by average-pooling or max-pooling (see below). The second approach is to construct an instance representation $\boldsymbol{z}_k$ of each instance by $h : \mathcal{S} \mapsto \mathbb{R}^{d_v}$ and then code the bag, or the input object, by pooling these instance representations (Wang et al., 2018) via a function f. An output function $o : \mathbb{R}^{d_v} \mapsto [0, 1]$ subsequently classifies the bag. The second approach, in which representations rather than scoring functions are pooled, is currently best performing (Wang et al., 2018). This approach is enhanced by an attention-mechanisms for pooling.

In the problem at hand, the input object X is the immune repertoire of an individual that consists of a large set of immune receptor sequences (T-cell receptors or antibodies). Immune receptors are primarily represented as sequences s_k from a space $s_k \in \mathcal{S}$. These sequences act as the instances in the MIL problem. An immune repertoire X is a bag of a large number of sequences $X = \{s_1, \ldots, s_{d_k}\}$, which corresponds to an input object.

Although immune repertoire classification can readily be formulated as multiple-instance learning problem, it is yet

unclear how well machine learning methods solve the above-described problem with a large number of instances $d_k \gg 10,000$ and with instances s_k being complex sequences. We next describe currently used pooling functions for multiple-instance learning problems.

Pooling functions for MIL problems. Different pooling functions equip a model g with the property to be invariant to permutations of instances and with the ability to process different numbers of instances. Typically, a neural network $h_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$ is trained to obtain a function that maps each instance onto a representation: $\boldsymbol{z}_k = h_{\boldsymbol{\theta}}(s_k)$. A representation of the input object $X = \{s_1, \ldots, s_{d_k}\}$ is constructed via one of the following pooling functions:

- Average-pooling:

$$\boldsymbol{z} = \frac{1}{d_k} \sum_{k=1}^{d_k} \boldsymbol{z}_k \tag{1}$$

- Max-pooling:

$$\boldsymbol{z} = \sum_{m=1}^{d_v} \boldsymbol{e}_m \left(\max_{k, 1 \leq k \leq d_k} \{z_{km}\} \right), \tag{2}$$

  where $\boldsymbol{e}_m = (0, \ldots, \underbrace{1}_{m\text{-th position}}, \ldots)$.

- Attention-pooling:

$$\boldsymbol{z} = \sum_{k=1}^{d_k} a_k \boldsymbol{z}_k, \tag{3}$$

  where a_k are non-negative ($a_k \geq 0$), sum to one ($\sum_{k=1}^{d_k} a_k = 1$), and are determined by an attention mechanism.

These pooling-functions are invariant to permutations of $\{1, \ldots, d_k\}$ and are differentiable. Therefore, they are suited as building blocks for Deep Learning architectures. Other types of pooling functions, that operate on predictions rather than representations, have also been suggested, for example the noisy-AND function (Kraus et al., 2016). We employ attention pooling in our DeepRC model as detailed in the following.

Transformer-like attention mechanism. The key-value-attention mechanism has been very successful and became popular through the Transformer (Vaswani et al., 2017a;b) and BERT (Devlin et al., 2018; 2019) models in natural language processing. Recently it was found that the key-value-attention mechanism corresponds to a modern Hopfield network with a storage capacity that is exponential in the dimension of the vector space and which converges after just one update (Ramsauer et al., 2020). Therefore using the key-value-attention mechanism is the natural choice for our task and is theoretically justified for the large number of vectors (sequence patterns) that appear in the immune repertoire classification task.

The attention mechanism assumes a similarity space of dimension d_e for keys and queries to compare them. A set of d_k key vectors are combined to the matrix $\boldsymbol{K} \in \mathbb{R}^{d_k \times d_e}$. A set of d_q query vectors are combined to the matrix $\boldsymbol{Q} \in \mathbb{R}^{d_q \times d_e}$. Similarities between queries and keys are computed by inner products, therefore queries can search for similar keys that are stored. Another set of d_k value vectors are combined to the matrix $\boldsymbol{V} \in \mathbb{R}^{d_k \times d_v}$. The output of the attention mechanism is a weighted average of the value vectors for each query $\boldsymbol{q}$. The i-th vector $\boldsymbol{v}_i$ is weighted by the similarity between the i-th key $\boldsymbol{k}_i$ and the query $\boldsymbol{q}$. The similarity is given by the softmax of the inner products of the query $\boldsymbol{q}$ with the keys $\boldsymbol{k}_i$. All queries are calculated in parallel via matrix operations. Consequently, the attention function $\mathrm{Att}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$ maps queries $\boldsymbol{Q}$, keys $\boldsymbol{K}$, and values $\boldsymbol{V}$ to d_v-dimensional outputs:

$$\mathrm{Att}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}; \beta) = \mathrm{softmax}(\beta \boldsymbol{Q} \boldsymbol{K}^T) \boldsymbol{V}, \tag{4}$$

where typically $\beta = \frac{1}{\sqrt{d_e}}$ and the softmax-function is applied row-wise. While this attention mechanism has originally been developed for sequence tasks (Vaswani et al., 2017c), it can readily be transferred to sets (Lee et al., 2019; Ye et al., 2018). We will employ this attention mechanism in DeepRC.

4. Deep Repertoire Classification

We propose a novel method **Deep R**epertoire **C**lassification (**DeepRC**) for immune repertoire classification with attention-based deep massive multiple instance learning and compare it against other machine learning approaches. For DeepRC, we consider *immune repertoires as input objects*, which are represented as bags of instances. In a bag, *each instance is an immune receptor sequence* and each bag can contain a large number of sequences. Note that we will use $\boldsymbol{z}_k$ to denote the *sequence-representation* of the k-th sequence and $\boldsymbol{z}$ to denote the *repertoire-representation*. At the core, DeepRC consists of a Transformer-like attention-mechanism that extracts the most important information from each repertoire. We first give an overview of the attention mechanism and then provide details on each of the sub-networks h_1, h_2, and o of DeepRC. (Overview: Fig. 1; Architecture: Fig.2; Implementation details: App. A2.)

Attention mechanism in DeepRC. This mechanism is based on the three matrices $\boldsymbol{K}$ (the keys), $\boldsymbol{Q}$ (the queries),

and $\boldsymbol{V}$ (the values) together with a parameter β.

Values. DeepRC uses a 1D-convolutional network h_1 (LeCun et al., 1998; Hu et al., 2014; Kelley et al., 2016) that supplies a sequence-representation $\boldsymbol{z}_k = h_1(s_k)$, which acts as the values $\boldsymbol{V} = \boldsymbol{Z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{d_k})$ in the attention mechanism (see Figure 2).

Keys. A second neural network h_2, which shares its first layers with h_1, is used to obtain keys $\boldsymbol{K} \in \mathbb{R}^{d_k \times d_e}$ for each sequence in the repertoire. This network uses 2 self-normalizing layers (Klambauer et al., 2017) with 32 units per layer (see Figure 2).

Query. We use a fixed d_e-dimensional query (row-)vector $\boldsymbol{\xi}$ which is learned via backpropagation. For more attention heads, each head has a fixed query vector.

With the quantities introduced above, the Transformer attention mechanism (4) of DeepRC is implemented as follows:

$$\boldsymbol{z} = \mathrm{Att}(\boldsymbol{\xi}, \boldsymbol{K}, \boldsymbol{Z}; \frac{1}{\sqrt{d_e}}) = \mathrm{softmax}\left(\frac{\boldsymbol{\xi}\boldsymbol{K}^T}{\sqrt{d_e}}\right)\boldsymbol{Z}, \quad (5)$$

where $\boldsymbol{Z} \in \mathbb{R}^{d_k \times d_v}$ are the sequence-representations stacked row-wise, $\boldsymbol{K}$ are the keys, and $\boldsymbol{z}$ is the repertoire-representation and at the same time a weighted mean of sequence-representations $\boldsymbol{z}_k$. The attention mechanism can readily be extended to multiple queries and heads, however, computational efficiency currently constrains this (see paragraph "1D-ConvNet for pattern recognition").

Attention-pooling and interpretability. Each input object, i.e. repertoire, consists of a large number d_k of sequences, which are reduced to a single fixed-size feature vector of length d_v representing the whole input object by an attention-pooling function. To this end, a Transformer-like attention mechanism adapted to sets is realized in DeepRC which supplies a_k – the importance of the sequence s_k. This importance value for each sequence, is an interpretable quantity, which is highly desired for the immunological problem at hand.

Classification layer and network parameters. The repertoire-representation $\boldsymbol{z}$ is then used as input for a fully-connected output network $\hat{y} = o(\boldsymbol{z})$ that predicts the immune status, where we found it sufficient to train single-layer networks. In the simplest case, DeepRC is predicts a scalar target y. That is, it predicts the class label, e.g. the immune status of an immune repertoire, using one output value. For this, the output values would be activated using a sigmoid function. However, since DeepRC is an end-to-end deep learning model, multiple targets may be predicted simultaneously in classification or regression settings or a mix of both. This allows for the introduction of additional information into the system via auxiliary targets such as age, sex, or other metadata.

Network parameters, training, and inference. DeepRC is trained using standard gradient descent methods to minimize a regularized cross-entropy loss. The network parameters are $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_o$ for the sub-networks h_1, h_2, and o, respectively, and additionally $\boldsymbol{\xi}$. In more detail, we train DeepRC using Adam (Kingma & Ba, 2014) with a batch size 4 and learning rate 10^{-4}. To increase numerical stability for 16 bit float computations, the ϵ parameter was set to an increased value of $\epsilon = 10^{-4}$.

1D-ConvNet for pattern 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 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.

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 to the one-hot vector of AAs, which results in a feature vector of size 23 per sequence position. Each repertoire, now represented as bag of feature vectors, is then normalized to unit variance. Since the *CMV dataset* provides sequences with an associated abundance or absolute frequency value per sequence, we incorporate this information into the input of DeepRC. The one-hot AA features of a sequence are multiplied by a scaling factor $\max(1, \log c_a)$ before normalization, where c_a is the count of a sequence. We feed the sequences with 23 features per position into the CNN, as shown in Figure 2.

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

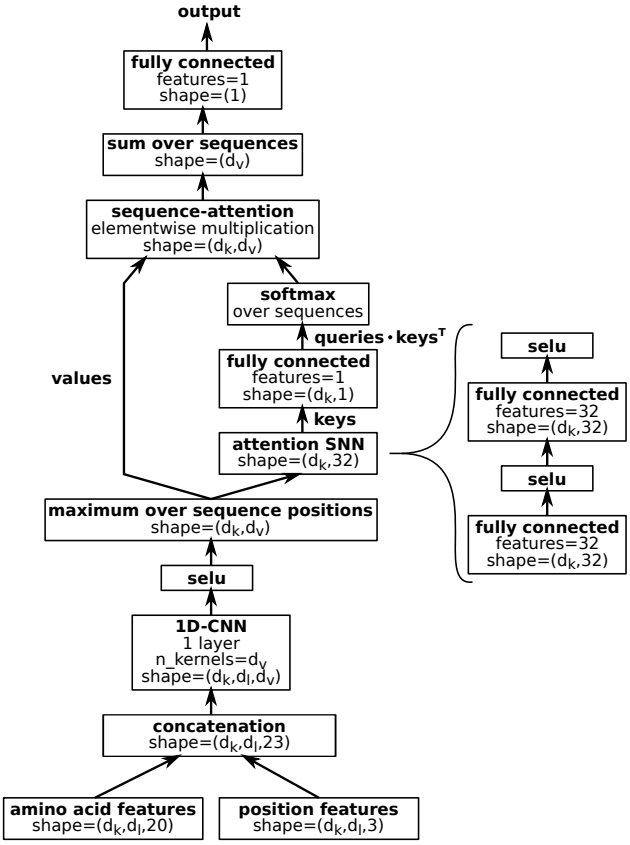


Figure 2. Schematic representation of the DeepRC architecture. d_l indicates the sequence length.

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 reduces the sequence-representations $\boldsymbol{z}_k$ to 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 1.

Regularization. We apply random and attention-based subsampling of repertoire sequences to reduce over-fitting and increase computational efficiency. During training, each repertoire is subsampled to $10,000$ input sequences, which are randomly drawn from the respective repertoire. This can also be interpreted as random drop-out (Hinton et al., 2012) on the input sequences or attention weights. During training and evaluation, the attention weights computed by the attention network are furthermore 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. Additionally, one might employ further regularization techniques, which we did not investigate further due to the high demands to computation time. 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, i.e. healthy, class. The last regularization technique assumes that the sequences in positive, i.e. diseased, class repertoires carry a signal, such as an AA motif corresponding to an immune response, whereas the sequences in negative repertoires do not. Hence, the sequences can be shuffled randomly between negative class repertoires without obscuring the signal in the positive class repertoires.

Interpretability. 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 in Appendix A5.

Hyperparameters. Table 1 provides an overview of the hyperparameter search, which was conducted as a grid-search for each of the datasets in a nested 5-fold cross-validation (CV) procedure, as described in section 5.2.

5. Experiments

5.1. 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 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 1. **DeepRC hyperparameter search space.** 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 10^5 updates. *: Experiments for $\{64; 128; 256\}$ kernels were omitted for datasets with motif implantation probabilities below 1% in category "simulated immunosequencing data".

learning methods. To this end, we either use simulated or experimentally-observed immune receptor sequences and we implant signals, specifically, sequence motifs (Akbar et al., 2019; Weber et al., 2020), at different frequencies into sequences of repertoires of the positive class. It has been shown previously that interaction of immune receptors with antigen occurs 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 and frequency of the implanted signal. 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.

Simulated immunosequencing data. In the first category, we aim at investigating the impact of the signal frequency and signal complexity on the performance of the different methods. To this end, we created 18 datasets, which each contain a large number of repertoires with a large number of random AA sequences in each repertoire. We then implanted signals in the AA sequences of the positive class repertoires, where the 18 datasets differ in frequency and complexity of the implanted signals. In detail, the AAs in the sequences were sampled randomly independent of their position in the sequence, while the frequencies of AAs, distribution of sequence lengths, and distribution of numbers of sequences per repertoires 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 a 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 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 AA, following the results of the experimental analysis of antigen-binding motifs in antibodies and T-cell receptor sequences (Akbar et al., 2019). We varied the characteristics of the implanted motifs for each of the 18 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. (b) The number of wild-card positions in the motif. A wild-card position contains a random AA character which is randomly sampled for each sequence. Wild-card positions are located in the center of the implanted motif. (c) The number of deletion positions in the implanted motif. A deletion position has 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. See Table 2 for an overview of the properties of the implanted motifs in the 18 datasets.

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 immunosequences 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 sequence, 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 A4), 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. Details on LSTM training and repertoire generation are given in Appendix A4.

After generation, each repertoire was assigned to either the positive or negative class, with 500 repertoires per class. We implanted motifs of length 4 and 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 (see Table 2). 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 2. Properties of simulated repertoires, variations of motifs, and motif frequencies 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".

Real-world data with implanted signals. In the third category, we implanted signals into experimentally obtained immunosequences, 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 3, 5, or 10 in the sequence 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.

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.

5.2. Methods compared

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.

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.

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_k 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 $\boldsymbol{u}_k = h_{\text{kmer}}(s_k)$, which is analogous to $\boldsymbol{z}_k$. Specifically, $u_{km} = (h_{\text{kmer}}(s_k))_m = \#\{p_m \in s_k\}$, where $\#\{p_m \in s_k\}$ denotes how often the k-mer pattern p_m occurs in sequence s_k. Afterwards, average-pooling is applied to obtain $\boldsymbol{u} = 1/d_k \sum_{k=1}^{d_k} \boldsymbol{u}_k$, 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:

$$\begin{aligned} k(X^{(n)}, X^{(l)}) &= k_{\text{MinMax}}(\boldsymbol{u}^{(n)}, \boldsymbol{u}^{(l)}) \\ &= \frac{\sum_{m=1}^{d_v} \min(u_m^{(n)}, u_m^{(l)})}{\sum_{m=1}^{d_v} \max(u_m^{(n)}, u_m^{(l)})}, \end{aligned} \tag{6}$$

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 Appendix Table A1.

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):

$$\begin{aligned} d_{\text{MinMax}}(\boldsymbol{u}^{(n)}, \boldsymbol{u}^{(l)}) &= 1 - k_{\text{MinMax}}(\boldsymbol{u}^{(n)}, \boldsymbol{u}^{(l)}), \\ d_{\text{Jaccard}}(\boldsymbol{u}^{(n)}, \boldsymbol{u}^{(l)}) &= 1 - k_{\text{Jaccard}}(\boldsymbol{u}^{(n)}, \boldsymbol{u}^{(l)}). \end{aligned} \tag{7}$$

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 Appendix Table A2.

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 2 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 Appendix Table A3.

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 Appendix Table A4.

5.3. Experimental Results

In this section, we report and analyze the predictive power of DeepRC and the compared methods on the datasets described in Section 5.1. The AUC is used as main metric for the predictive power to focus on the ranking of the repertoires instead of the classification boundary.

Hyperparameter selection. We used a nested 5-fold cross validation (CV) procedure to estimate the performance of each of the methods. 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. For the hyperparameter search of DeepRC for the category "simulated immunosequencing data", we only conducted a large-scale hyperparameter search on the datasets with motif implantation probabilities below 1% due to computational constraints, as described in Table 1. This process is repeated for all 5 folds of the 5-fold CV and the average score on the 5 test sets constitutes the final score of a method. The results are reported in Table 3.

Results. In each of the four categories, "real-world data", "real-world data with implanted signals", "LSTM-generated data", and "simulated immunosequencing data", DeepRC outperforms all competing methods with respect to average AUC (see Table 3, Appendix Tables A5, A7, and A6). In all

	Real-world	Real-World data with implanted signals				LSTM-generated data					Simulated
	CMV	OM 1%	OM 0.1%	MM 1%	MM 0.1%	10%	1%	0.5%	0.1%	0.05%	avg.
DeepRC	**0.831** ± 0.002	**1.00** ± 0.00	**0.98**± 0.01	**1.00**± 0.00	**0.94**±0.01	**1.00**± 0.00	**1.00**± 0.00	**1.00**± 0.00	**1.00**± 0.00	**1.00**± 0.00	**0.846**± 0.223
SVM (MM)	0.825 ± 0.022	**1.00** ± 0.00	0.58± 0.02	**1.00**± 0.00	0.53±0.02	**1.00**± 0.00	**1.00**± 0.00	**1.00**± 0.00	**1.00**± 0.00	0.99± 0.01	0.827± 0.210
SVM (J)	0.546 ± 0.021	0.99 ± 0.00	0.53± 0.02	**1.00**± 0.00	0.57±0.02	0.98± 0.04	**1.00**± 0.00	**1.00**± 0.00	0.90± 0.04	0.77± 0.07	0.550± 0.080
KNN (MM)	0.679 ± 0.076	0.74 ± 0.24	0.49± 0.03	0.67± 0.18	0.50±0.02	0.70± 0.27	0.72± 0.26	0.73± 0.26	0.54± 0.16	0.52± 0.15	0.634± 0.129
KNN (J)	0.534 ± 0.039	0.65 ± 0.16	0.48± 0.03	0.70± 0.20	0.51±0.03	0.70± 0.29	0.61± 0.24	0.52± 0.16	0.55± 0.19	0.54± 0.19	0.501± 0.007
Log. Regr.	0.607 ± 0.058	**1.00** ± 0.00	0.54± 0.04	0.99± 0.00	0.51±0.04	**1.00**± 0.00	**1.00**± 0.00	0.93± 0.15	0.60± 0.19	0.43± 0.16	0.826± 0.211
Log. MIL (KMER)	0.582 ± 0.065	0.54 ± 0.07	0.51± 0.03	0.99± 0.00	0.62±0.15	**1.00**± 0.00	0.72± 0.11	0.64± 0.14	0.57± 0.15	0.53± 0.13	0.665± 0.224
Log. MIL (TCRB)	0.515 ± 0.073	0.50 ± 0.03	0.50± 0.02	0.99± 0.00	0.78±0.03	0.54± 0.09	0.57± 0.16	0.47± 0.09	0.51± 0.07	0.50± 0.12	0.501± 0.016
Known motif b.	–	1.00 ± 0.00	0.70± 0.03	0.99± 0.00	0.62±0.04	1.00± 0.00	1.00± 0.00	1.00± 0.00	1.00± 0.00	1.00± 0.00	0.890± 0.168
Known motif c.	–	0.92 ± 0.00	0.56± 0.03	0.65± 0.03	0.52±0.03	1.00± 0.00	1.00± 0.00	0.99± 0.01	0.72± 0.09	0.63± 0.09	0.738± 0.202

Table 3. Results in terms of AUC of the method comparisons on all datasets. The reported errors are standard deviations across 5 cross-validation folds (except for the column "Simulated"). **Real-world CMV:** Average performance over 5 cross-validation folds in terms of AUC. **Real-world data with implanted signals:** Average performance over 5 cross-validation folds in terms of AUC for each of the four datasets. In each dataset, a signal was implanted with different frequency of 1% or 0.1%, and either a single motif ("OM") or multiple motifs ("MM") were implanted. **LSTM-generated data:** Average performance over 5 cross-validation folds in terms of AUC for each of the 5 datasets. In each dataset, a signal was implanted with different frequency of 10%, 1%, 0.5%, 0.1%, and 0.05%, respectively. **Simulated:** Here we report the mean over 18 simulated datasets with implanted signals (see Table A5 for details). The error reported is the standard deviation of the AUC values across datasets that have varying difficulties. More detailed results are provided in Appendix A3.

categories, the runner-up method is the SVM with MinMax kernel.

Results on simulated immunosequencing data. As mentioned in section 5.1, in this setting the complexity of the implanted signal is in focus and varies throughout 18 simulated datasets. Some datasets are difficult to classify because the implanted motif is hidden by noise and others are challenging to classify because only a small fraction of sequences carry the motif. These difficulties become evident by the method called "known motif binary", which assumes the implanted motif is known. The performance of this method ranges from a perfect AUC of 1.00 in several datasets to an AUC of 0.532 in dataset '17' (see Appendix Table A5).

DeepRC outperforms all other methods with an average AUC of 0.864 ± 0.223, followed by the SVM with MinMax kernel with an average AUC of 0.827 ± 0.210 (see Appendix Table A5). The predictive performance of all methods suffers if the signal occurs only in an extremely small fraction of sequences, which becomes evident from datasets '8', '11', '14', and '17'. In these datasets, only 0.01% of the sequences carry the motif and all AUC values are below 0.55.

Results on LSTM-generated data. On the LSTM-generated data, in which we implanted noisy motifs with frequencies of 10%, 1%, 0.5%, 0.1%, and 0.05%, DeepRC yields almost perfect predictive performance with an average AUC of 1.000 ± 0.001 (see Table A6). The second best method, SVM with MinMax kernel, has a similar predictive performance to DeepRC on all 5 datasets but the other competing methods have a lower predictive performance at the datasets with low frequency of the signal (0.05%).

Results on Real-world data with implanted motifs. In this dataset category, we used real immuno-sequences and implanted single or multiple noisy motifs. Again, DeepRC outperforms all other methods with an average AUC of 0.980 ± 0.029, with the second best method being the SVM with MinMax kernel an average AUC of 0.777 ± 0.258. Notably, all methods except for DeepRC have difficulties with noisy motifs at a frequency of 0.1% (see columns "One 0.1%" and "Multi 0.1%" in Table A7).

Results on Real-world data. On the real-world dataset, in which the immune status of persons affected by the cytomegalovirus has to be detected, the competing methods yield predictive AUCs between 0.515 and 0.831. We note that this dataset is not the exact dataset that was used in Emerson et al. (2017). It differs in several ways in preprocessing steps and also comprises a different set of repertoires, which leads to a more challenging dataset. We are currently working on an immune receptor machine learning platform (Pavlović et al., 2020) that aims to enable

direct replication and comparability with previous studies. The best performing method is DeepRC with an AUC of 0.831 ± 0.002, followed by the SVM with MinMax kernel (AUC 0.825 ± 0.022) and KNN with the same kernel with an AUC of 0.678 ± 0.076. The top-ranked sequences by DeepRC significantly correspond to those detected by Emerson et al. (2017), which we tested by a U-test with the null hypothesis that the attention values of the sequences detected by Emerson et al. (2017) would be equal to the attention values of the remaining sequences (p-value $1.3 \cdot 10^{-93}$). The sequences and their attention values are displayed in Appendix Table A10.

6. Conclusion

We have demonstrated how attention-based deep multiple instance learning can be adapted to the task of classifying the immune status of immune repertoires. For immune state classification, methods have to identify the discriminating sequences among a large set of sequences in an immune repertoire, specifically, even motifs within those sequences have to be identified. We have shown that a 1D-convolutional network combined with a Transformer-like attention mechanism with a fixed query can solve this difficult task across a range of different experimental conditions. In large-scale experiments, we have compared a set of machine learning methods and previously suggested methods with DeepRC and found that DeepRC yields the best predictive quality with respect to AUC. Furthermore, DeepRC can be interpreted with suitable methods, by which we extract motifs from the sequences (Appendix A5).

The current methods are mostly limited by computational complexity, since both hyperparameter and model selection is computationally demanding. For hyperparameter selection, a substantial set of architectures have to be searched to identify a proficient one. For model selection, a single repertoire requires the propagatation of many thousands of sequences through a neural network and keep those quantities in GPU memory for the attention mechanism. 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.

We envision that with the increasing availability of large immunosequencing datasets (Kovaltsuk et al., 2018; Corrie et al., 2018; Christley et al., 2018; Zhang et al., 2020), further fine-tuning of ground-truth benchmarking immune receptor datasets (Weber et al., 2020; Olson et al., 2019; Marcou et al., 2018), increased GPU memory, and increased computing power, it will be possible to identify discriminating immune receptor motifs for many diseases, potentially even for the current SARS-CoV-2 (CoViD-19) pandemic. Such results would pave the way towards antibody and TCR-driven immunotherapies and immunodiagnostics as well as rational vaccine design.

Availability

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`.

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*, pp. 759498, 2019.

Alipanahi, B., Delong, A., Weirauch, M. T., and Frey, B. J. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831–838, 2015.

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.

Bashford-Rogers, R., Bergamaschi, L., McKinney, E., Pombal, D., Mescia, F., Lee, J., Thomas, D., Flint, S., Kellam, P., Jayne, D., et al. Analysis of the b cell receptor repertoire in six immune-mediated diseases. *Nature*, 574 (7776):122–126, 2019.

Brown, A. J., Snapkov, I., Akbar, R., Pavlović, M., Miho, E., Sandve, G. K., and Greiff, V. Augmenting adaptive immunity: progress and challenges in the quantitative engineering and analysis of adaptive immune receptor repertoires. *Molecular Systems Design & Engineering*, 4 (4):701–736, 2019.

Christley, S., Scarborough, W., Salinas, E., Rounds, W. H., Toby, I. T., Fonner, J. M., Levin, M. K., Kim, M., Mock, S. A., Jordan, C., et al. Vdjserver: a cloud-based analysis portal and data commons for immune repertoire sequences and rearrangements. *Frontiers in immunology*, 9: 976, 2018.

Christophersen, A., Ráki, M., Bergseng, E., Lundin, K. E., Jahnsen, J., Sollid, L. M., and Qiao, S.-W. Tetramer-

visualized gluten-specific cd4+ t cells in blood as a potential diagnostic marker for coeliac disease without oral gluten challenge. *United European gastroenterology journal*, 2(4):268–278, 2014.

Corrie, B. D., Marthandan, N., Zimonja, B., Jaglale, J., Zhou, Y., Barr, E., Knoetze, N., Breden, F. M., Christley, S., Scott, J. K., et al. ireceptor: A platform for querying and analyzing antibody/b-cell and t-cell receptor repertoire data across federated repositories. *Immunological reviews*, 284(1):24–41, 2018.

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

Dash, P., Fiore-Gartland, A. J., Hertz, T., Wang, G. C., Sharma, S., Souquette, A., Crawford, J. C., Clemens, E. B., Nguyen, T. H., Kedzierska, K., et al. Quantifiable predictive features define epitope-specific t cell receptor repertoires. *Nature*, 547(7661):89–93, 2017.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. *ArXiv*, 2018.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1423.

Dietterich, T. G., Lathrop, R. H., and Lozano-Pérez, T. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1-2):31–71, 1997.

Elhanati, Y., Sethna, Z., Callan Jr, C. G., Mora, T., and Walczak, A. M. Predicting the spectrum of tcr repertoire sharing with a data-driven model of recombination. *Immunological reviews*, 284(1):167–179, 2018.

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.

Fischer, D. S., Wu, Y., Schubert, B., and Theis, F. J. Predicting antigen-specificity of single t-cells based on tcr cdr3 regions. *BioRxiv*, pp. 734053, 2019.

Georgiou, G., Ippolito, G. C., Beausang, J., Busse, C. E., Wardemann, H., and Quake, S. R. The promise and challenge of high-throughput sequencing of the antibody repertoire. *Nature biotechnology*, 32(2):158, 2014.

Gielis, S., Moris, P., Bittremieux, W., De Neuter, N., Ogunjimi, B., Laukens, K., and Meysman, P. Tcrex: detection of enriched t cell epitope specificity in full t cell receptor sequence repertoires. *bioRxiv*, pp. 373472, 2019.

Glanville, J., Huang, H., Nau, A., Hatton, O., Wagar, L. E., Rubelt, F., Ji, X., Han, A., Krams, S. M., Pettus, C., et al. Identifying specificity groups in the t cell receptor repertoire. *Nature*, 547(7661):94–98, 2017.

Graves, A. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Greiff, V., Bhat, P., Cook, S. C., Menzel, U., Kang, W., and Reddy, S. T. A bioinformatic framework for immune repertoire diversity profiling enables detection of immunological status. *Genome medicine*, 7(1):49, 2015.

Greiff, V., Weber, C. R., Palme, J., Bodenhofer, U., Miho, E., Menzel, U., and Reddy, S. T. Learning the high-dimensional immunogenomic features that predict public and private antibody repertoires. *The Journal of Immunology*, 199(8):2985–2997, 2017.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

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

Hochreiter, S., Heusel, M., and Obermayer, K. Fast model-based protein homology detection without alignment. *Bioinformatics*, 23(14):1728–1736, 2007.

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.

Jurtz, V. I., Jessen, L. E., Bentzen, A. K., Jespersen, M. C., Mahajan, S., Vita, R., Jensen, K. K., Marcatili, P., Hadrup, S. R., Peters, B., et al. Nettcr: sequence-based prediction of tcr binding to peptide-mhc complexes using convolutional neural networks. *bioRxiv*, pp. 433706, 2018.

Kelley, D. R., Snoek, J., and Rinn, J. L. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome research*, 26(7): 990–999, 2016.

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 preprint arXiv:1911.06616*, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 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.

Konishi, H., Komura, D., Katoh, H., Atsumi, S., Koda, H., Yamamoto, A., Seto, Y., Fukayama, M., Yamaguchi, R., Imoto, S., et al. Capturing the differences between humoral immunity in the normal and tumor environments from repertoire-seq of b-cell receptors using supervised machine learning. *BMC bioinformatics*, 20(1): 1–11, 2019.

Kovaltsuk, A., Leem, J., Kelm, S., Snowden, J., Deane, C. M., and Krawczyk, K. Observed antibody space: A resource for data mining next-generation sequencing of antibody repertoires. *The Journal of Immunology*, 201(8): 2502–2509, 2018.

Kraus, O. Z., Ba, J. L., and Frey, B. J. Classifying and segmenting microscopy images with deep multiple instance learning. *Bioinformatics*, 32(12):i52–i59, 2016.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pp. 3744–3753, 2019.

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

Liu, X., Zhang, W., Zhao, M., Fu, L., Liu, L., Wu, J., Luo, S., Wang, L., Wang, Z., Lin, L., et al. T cell receptor β repertoires as novel diagnostic markers for systemic lupus erythematosus and rheumatoid arthritis. *Annals of the rheumatic diseases*, 78(8):1070–1078, 2019.

Marcou, Q., Mora, T., and Walczak, A. M. High-throughput immune repertoire analysis with igor. *Nature communications*, 9(1):1–10, 2018.

Maron, O. and Lozano-Pérez, T. A framework for multiple-instance learning. In *Advances in neural information processing systems*, pp. 570–576, 1998.

Miho, E., Yermanos, A., Weber, C. R., Berger, C. T., Reddy, S. T., and Greiff, V. Computational strategies for dissecting the high-dimensional complexity of adaptive immune repertoires. *Frontiers in immunology*, 9:224, 2018.

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.

Mora, T. and Walczak, A. M. How many different clonotypes do immune repertoires contain? *Current Opinion in Systems Biology*, 18:104 – 110, 2019. ISSN 2452-3100. doi: https://doi.org/10.1016/j.coisb.2019.10.001. URL http://www.sciencedirect.com/science/article/pii/S2452310019300289.

Moris, P., De Pauw, J., Postovskaya, A., Ogunjimi, B., Laukens, K., and Meysman, P. Treating biomolecular interaction as an image classification problem–a case study on t-cell receptor-epitope recognition prediction. *bioRxiv*, 2019.

Olson, B. J., Moghimi, P., Schramm, C., Obraztsova, A., Ralph, D. K., Vander Heiden, J. A., Shugay, M., Shepherd, A. J., Lees, W. D., Matsen, I., et al. sumrep: a summary statistic framework for immune receptor repertoire comparison and model validation. *Frontiers in immunology*, 10:2533, 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.

Pavlović, M., Scheffer, L., Chernigovskaya, M., Widrich, M., Klambauer, G., Greiff, V., and Sandve, G. K. An open-source platform for novice to expert-operated interpretable machine learning analysis of immune receptor data. In preparation, 2020.

Pawlowski, N., Bhooshan, S., Ballas, N., Ciompi, F., Glocker, B., and Drozdzal, M. Needles in haystacks: On classifying tiny objects in large images. *arXiv preprint arXiv:1908.06037*, 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.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

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

Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Gruber, L., Holzleitner, M., Klambauer, G., Brandstetter, J., and Hochreiter, S. Hopfield networks is all you need, 2020. To appear.

Shugay, M., Bagaev, D. V., Turchaninova, M. A., Bolotin, D. A., Britanova, O. V., Putintseva, E. V., Pogorelyy, M. V., Nazarov, V. I., Zvyagin, I. V., Kirgizova, V. I., et al. Vdjtools: unifying post-analysis of t cell receptor repertoires. *PLoS computational biology*, 11(11), 2015.

Sidhom, J.-W., Larman, H. B., Ross-MacDonald, P., Wind-Rotolo, M., Pardoll, D. M., and Baras, A. S. Deeptcr: a deep learning framework for understanding t-cell receptor sequence signatures within complex t-cell repertoires. *bioRxiv*, 2019. doi: 10.1101/464107. URL https://www.biorxiv.org/content/early/2019/12/23/464107.

Springer, I., Besser, H., Tickotsky-Moskovitz, N., Dvorkin, S., and Louzoun, Y. Prediction of specific tcr-peptide binding from large dictionaries of tcr-peptide pairs. *bioRxiv*, 2020. doi: 10.1101/650861. URL https://www.biorxiv.org/content/early/2020/01/19/650861.

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. org, 2017.

Tomita, N., Abdollahi, B., Wei, J., Ren, B., Suriawinata, A., and Hassanpour, S. Attention-based deep neural networks for detection of cancerous and precancerous esophagus tissue on histopathological slides. *JAMA network open*, 2 (11):e1914645–e1914645, 2019.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. Curran Associates, Inc., 2017a.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *ArXiv*, 2017b.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017c.

Wang, X., Yan, Y., Tang, P., Bai, X., and Liu, W. Revisiting multiple instance neural networks. *Pattern Recognition*, 74:15–24, 2018.

Wardemann, H. and Busse, C. E. Novel approaches to analyze immunoglobulin repertoires. *Trends in immunology*, 38(7):471–482, 2017.

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. ISSN 1367-4803. doi: 10.1093/bioinformatics/btaa158. URL https://doi.org/10.1093/bioinformatics/btaa158. btaa158.

Wucherpfennig, K. W., Allen, P. M., Celada, F., Cohen, I. R., De Boer, R., Garcia, K. C., Goldstein, B., Greenspan, R., Hafler, D., Hodgkin, P., et al. Polyspecificity of t cell and b cell receptor recognition. In *Seminars in immunology*, volume 19, pp. 216–224. Elsevier, 2007.

Yaari, G. and Kleinstein, S. H. Practical guidelines for b-cell receptor repertoire sequencing analysis. *Genome medicine*, 7(1):121, 2015.

Ye, H.-J., Hu, H., Zhan, D.-C., and Sha, F. Learning embedding adaptation for few-shot learning. *arXiv preprint arXiv:1812.03664*, 2018.

Zeng, H., Edwards, M. D., Liu, G., and Gifford, D. K. Convolutional neural network architectures for predicting dna–protein binding. *Bioinformatics*, 32(12):i121–i127, 2016.

Zhang, W., Wang, L., Liu, K., Wei, X., Yang, K., Du, W., Wang, S., Guo, N., Ma, C., Luo, L., et al. Pird: Pan immune repertoire database. *Bioinformatics*, 36(3):897–903, 2020.

Zhou, J. and Troyanskaya, O. G. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature methods*, 12(10):931–934, 2015.

Appendix

A1. Details on hyperparameter selection

Known motif. This method has no 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 Appendix A1).

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

Table A1. 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 Appendix A2).

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

Table A2. 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. Hyperparameter optimization strategy: grid search.

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

Table A3. Settings used in the hyperparameter search of the Logistic Regression baseline approach.

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 A4).

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

Table A4. 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.

A2. DeepRC implementation details

In this section we provide more detailed information on the implementation of DeepRC.

Positional features. We use 3 features to encode the relative position of each AA, as illustrated in Figure A1. The inputs are provided to the DeepRC network as 16 bit floating point values. Sequences of different lengths were zero-padded to the maximum sequence length per batch at the sequence ends.

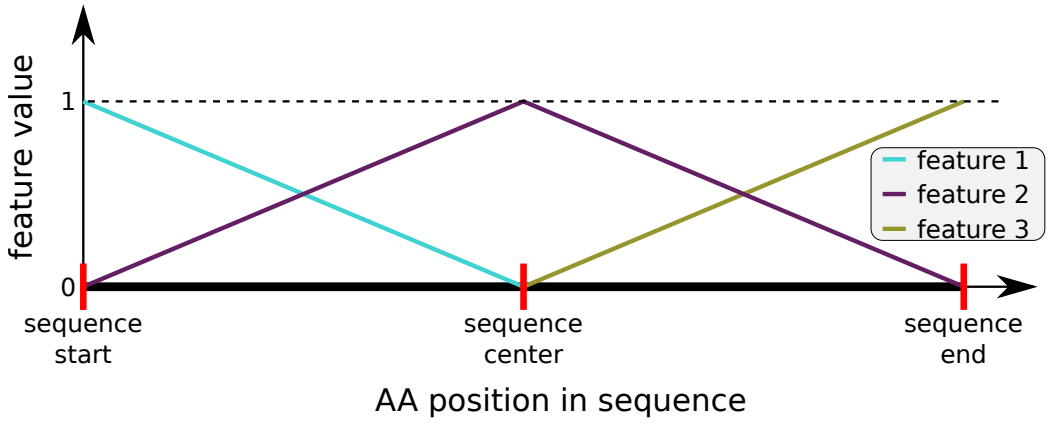


Figure A1. We use 3 input features with values in range $[0, 1]$ to encode the relative position of each AA in a sequence with respect to the sequence. "feature 1" encodes if an AA is close to the sequence start, "feature 2" to the sequence center, and "feature 3" to the sequence end. For every position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

Computation time and optimization. As mentioned in section 4, we took measures to meet the high demands of computation time and GPU memory consumption in our implementation, 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 set to an increased value of $\epsilon = 10^{-4}$. Training was performed on various GPU types, mainly `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 update on a `RTX 2080 Ti` GPU took approximately 0.0129 to 0.0135 seconds, while requiring approximately 8 to 11 GB GPU memory. Taking these optimizations and GPUs with larger memory ($\geq$ 16 GB) into account, it would already be feasible to train DeepRC, possibly with multi-head attention and a larger network architecture, on larger datasets. 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 input via concatenation to the feature vectors that result from the maximum over the sequence positions of the 1D-CNN output. 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.

A3. Detailed results

ID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	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%	–
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	–
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	**0.864**
	± 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	± 0.223
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	0.827
	± 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	± 0.210
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	0.550
	± 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	± 0.080
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	0.634
	± 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	± 0.129
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**	0.501
	± 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	± 0.007
Logistic Regression	**1.000**	**1.000**	**0.786**	**1.000**	**1.000**	**0.607**	**1.000**	0.997	0.527	**1.000**	0.992	0.526	**1.000**	0.719	0.505	**1.000**	0.694	0.510	0.826
	± 0.000	± 0.000	± 0.009	± 0.000	± 0.000	± 0.025	± 0.000	± 0.002	± 0.018	± 0.000	± 0.004	± 0.019	± 0.000	± 0.019	± 0.015	± 0.001	± 0.021	± 0.017	± 0.211
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	0.665
	± 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	± 0.224
Logistic MIL (TCRB)	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	0.501
	± 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	± 0.016
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	0.890
	± 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	± 0.168
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	0.738
	± 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	± 0.202

Table A5. AUC estimates based on 5-fold CV for all 18 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. Wild-card characters in motifs are indicated by Z, characters with 50% probability of being removed by ^{d}.

ID	0	1	2	3	4	avg.
motif freq. ρ	10%	1%	0.5%	0.1%	0.05%	–
implanted motif	$G^rS^rA^rF^r$	$G^rS^rA^rF^r$	$G^rS^rA^rF^r$	$G^rS^rA^rF^r$	$G^rS^rA^rF^r$	–
DeepRC	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**0.998** ± 0.002	**1.000** ± 0.001
SVM (MinMax)	**1.000** ± 0.000	**1.000** ± 0.000	0.999 ± 0.001	0.999 ± 0.002	0.985 ± 0.014	0.997 ± 0.007
SVM (Jaccard)	0.981 ± 0.041	**1.000** ± 0.000	**1.000** ± 0.000	0.904 ± 0.036	0.768 ± 0.068	0.931 ± 0.099
KNN (MinMax)	0.699 ± 0.272	0.717 ± 0.263	0.732 ± 0.263	0.536 ± 0.156	0.516 ± 0.153	0.640 ± 0.105
KNN (Jaccard)	0.698 ± 0.285	0.606 ± 0.237	0.523 ± 0.164	0.550 ± 0.186	0.539 ± 0.194	0.583 ± 0.071
Logistic Regression	**1.000** ± 0.000	**1.000** ± 0.000	0.934 ± 0.147	0.604 ± 0.193	0.427 ± 0.156	0.793 ± 0.262
Logistic MIL (KMER)	0.997 ± 0.004	0.718 ± 0.112	0.637 ± 0.144	0.571 ± 0.146	0.528 ± 0.129	0.690 ± 0.186
Logistic MIL (TCRB)	0.541 ± 0.086	0.566 ± 0.162	0.468 ± 0.086	0.505 ± 0.067	0.500 ± 0.121	0.516 ± 0.038
Known motif b.	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.999 ± 0.003	0.999 ± 0.003	1.000 ± 0.001
Known motif c.	1.000 ± 0.000	1.000 ± 0.000	0.989 ± 0.011	0.722 ± 0.085	0.626 ± 0.094	0.867 ± 0.180

Table A6. 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 5.1, paragraph ""LSTM-generated data", are indicated by ^{r}.

	One Motif 1%	One 0.1%	Multi 1%	Multi 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	0.544 ± 0.035	0.991 ± 0.003	0.512 ± 0.035	0.762 ± 0.270
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 (TCRB)	0.503 ± 0.032	0.501 ± 0.016	0.992 ± 0.003	0.782 ± 0.030	0.695 ± 0.238
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 A7. 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. **One Motif 1%:** In this dataset, a single motif with a frequency of 1% was implanted. **One 0.1%:** In this dataset, a single motif with a frequency of 0.1% was implanted. **Multi 1%:** In this dataset, multiple motifs with a frequency of 1% were implanted. **Multi 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 5.1, paragraph "Real-world data with implanted signals.".

	AUC	F1 score	balanced accuracy	accuracy
DeepRC	**0.831** ± 0.002	**0.728** ± 0.041	**0.741** ± 0.043	0.727 ± 0.049
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	0.607 ± 0.058	0.244 ± 0.206	0.552 ± 0.049	0.590 ± 0.019
Logistic MIL (KMER)	0.582 ± 0.065	0.118 ± 0.264	0.503 ± 0.007	0.515 ± 0.058
Logistic MIL (TCRB)	0.515 ± 0.073	0.000 ± 0.000	0.496 ± 0.008	0.541 ± 0.039

Table A8. 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.

A4. Next-character LSTM

We trained a classic next-character LSTM model (Graves, 2013) based on the implementation `https://github.com/spro/practical-pytorch` using `PyTorch 1.3.1` (Paszke et al., 2019). For this, we appled 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 batchsize of 100 character chunks, and a character chunk lenght of 200. As input we used the immuno-sequences in the `CDR3` column of the *CMV dataset*, where we repeated sequences according to the counts of the sequences 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 5, paragraph "LSTM-generated data.".

As illustrated in the comparison of histograms given in Table A4, the generated immuno-sequences exhibit a very similar distribution of 4-mers and AAs compared to the original *CMV dataset*.

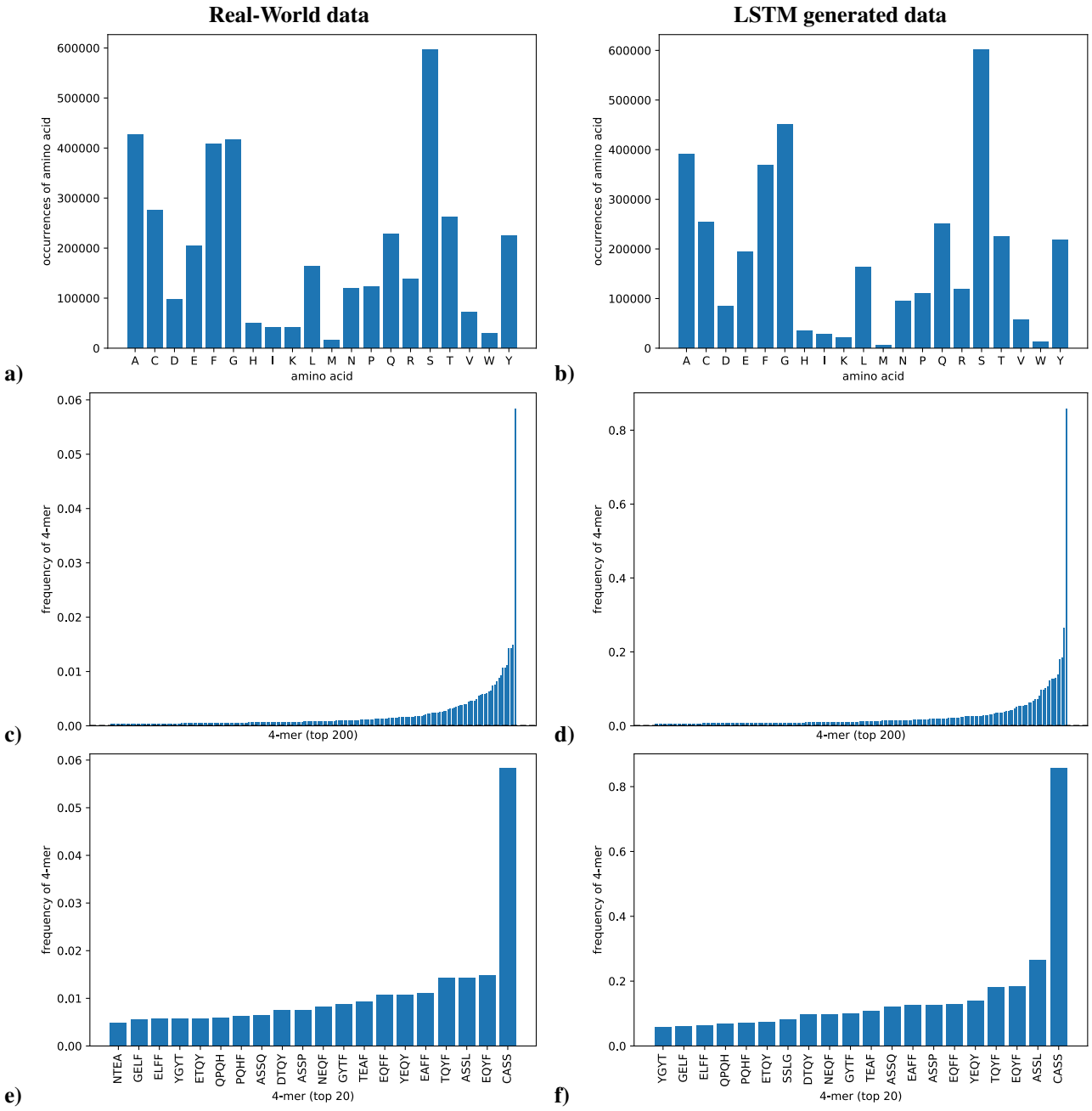
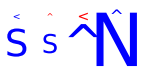


Figure A2. Distribution of AAs and k-mers in real-world *CMV dataset* and LSTM-generated data. **Left:** Histograms of real-world data. **Right:** Histograms of LSTM-generated data. **a)** Frequency of AAs in sequences of the *CMV dataset*. **b)** Frequency of AAs in sequences of the LSTM-generated datasets. **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.

A5. Interpreting DeepRC

In this section, we provide examples for the interpretation of trained DeepRC models using Integrated Gradients (IG) (Sundararajan et al., 2017) as contribution analysis method. Application of IG was performed as described in section 4, paragraph "Interpretability". 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. A9 and Fig. A3, shows that the implanted motifs can be successfully extracted from the trained model and are straight-forward 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 A9. 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. Wild-card 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 SZ^{d}Z^{d}N and motif implantation probability 0.1%. The DeepRC model reacts to the S and N at the 5th and 8th 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 {L^{r}D^{r}R^{r}; C^{r}A^{r}S; G^{r}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 5th and 7th sequence position (c), thereby identifying the implanted motif in the sequences. Wild-card characters in implanted motifs are indicated by Z, characters with 50% probability of being removed by ^{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.

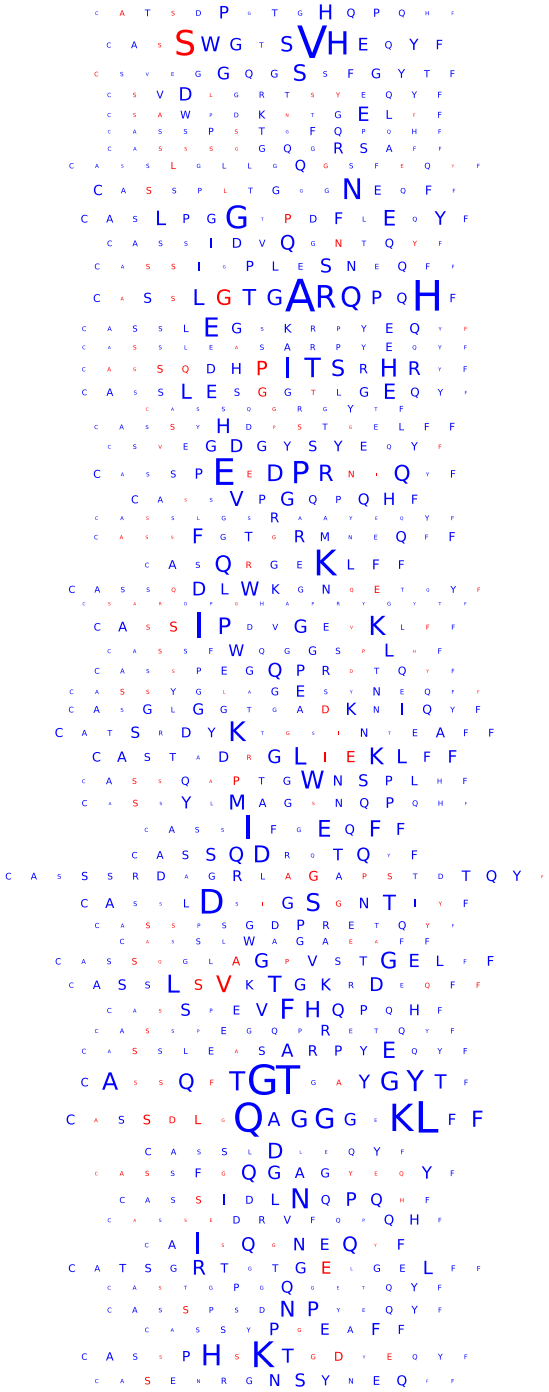


Figure A4. Visualization of the contributions of characters within a sequence via IG. Each sequence was selected from a different repertoire and showed the highest contribution in its repertoire. Model was trained on *CMV dataset*, using a kernel size of 9, 32 kernels and 137 repertoires for early stopping. 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.

A6. 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 A10. 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.