bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

arXiv:2007.13505v1 [cs.LG] 16 Jul 2020

Modern Hopfield Networks and Attention for Immune Repertoire Classification

Michael Widrich[*] **Bernhard Schäfl**[*] **Milena Pavlović**[‡,§] **Hubert Ramsauer**[*]

Lukas Gruber[*] **Markus Holzleitner**[*] **Johannes Brandstetter**[*] **Geir Kjetil Sandve**[§]

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

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

Abstract

A central mechanism in machine learning is to identify, store, and recognize patterns. How to learn, access, and retrieve such patterns is crucial in Hopfield networks and the more recent transformer architectures. We show that the attention mechanism of transformer architectures is actually the update rule of modern Hopfield networks that can store exponentially many patterns. We exploit this high storage capacity of modern Hopfield networks to solve a challenging multiple instance learning (MIL) problem in computational biology: immune repertoire classification. Accurate and interpretable machine learning methods solving this problem could pave the way towards new vaccines and therapies, which is currently a very relevant research topic intensified by the COVID-19 crisis. Immune repertoire classification based on the vast number of immunosequences of an individual is a MIL problem with an unprecedentedly massive number of instances, two orders of magnitude larger than currently considered problems, and with an extremely low witness rate. In this work, we present our novel method DeepRC that integrates transformer-like attention, or equivalently modern Hopfield networks, into deep learning architectures for massive MIL such as immune repertoire classification. We demonstrate that DeepRC outperforms all other methods with respect to predictive performance on large-scale experiments, including simulated and real-world virus infection data, and enables the extraction of sequence motifs that are connected to a given disease class. Source code and datasets: *https://github.com/ml-jku/DeepRC*

Introduction

Transformer architectures (Vaswani et al., 2017) and their attention mechanisms are currently used in many applications, such as natural language processing (NLP), imaging, and also in multiple instance learning (MIL) problems (Lee et al., 2019). In MIL, a set or bag of objects is labelled rather than objects themselves as in standard supervised learning tasks (Dietterich et al., 1997). Examples for MIL problems are medical images, in which each sub-region of the image represents an instance, video

Preprint. Under review.

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

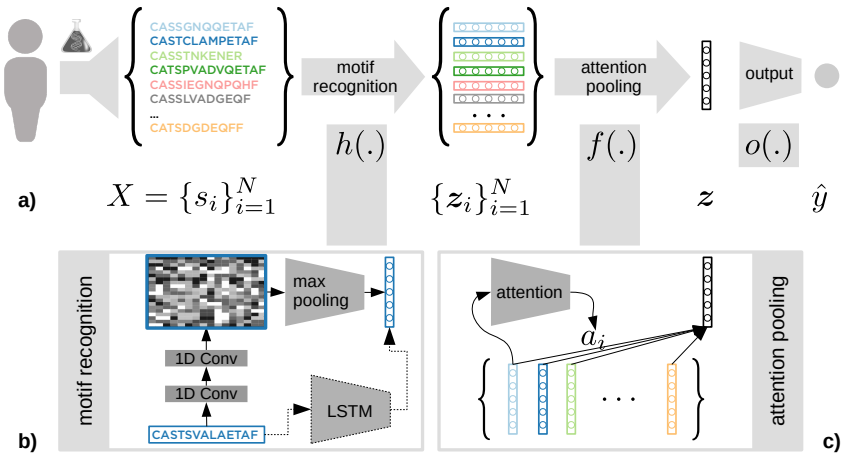

Figure 1: Schematic representation of the DeepRC approach. **a)** An immune repertoire X is represented by large bags of immune receptor sequences (colored). A neural network (NN) h serves to recognize patterns in each of the sequences s_i and maps them to sequence-representations $\boldsymbol{z}_i$. 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 recurrent neural networks (RNNs), such as LSTMs (Hochreiter et al., 2007), or transformer networks (Vaswani et al., 2017) 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 weights are determined by an update rule of modern Hopfield networks that allows to retrieve exponentially many patterns.

classification, in which each frame is an instance, text classification, where words or sentences are instances of a text, point sets, where each point is an instance of a 3D object, and remote sensing data, where each sensor is an instance (Carbonneau et al., 2018; Uriot, 2019). 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) and transformer-like attention mechanisms for sets of points and images (Lee et al., 2019).

However, in MIL problems considered by machine learning methods up to now, the number of instances per bag is in the range of hundreds or few thousands (Carbonneau et al., 2018; Lee et al., 2019) (see also Tab. A2). At the same time the witness rate (WR), the rate of discriminating instances per bag, is already considered low at $1\% - 5\%$. We will tackle the problem of *immune repertoire classification* with hundreds of thousands of instances per bag without instance-level labels and with extremely low witness rates down to 0.01% using an attention mechanism. We show that the attention mechanism of transformers is the update rule of modern Hopfield networks (Krotov & Hopfield, 2016, 2018; Demircigil et al., 2017) that are generalized to continuous states in contrast to classical Hopfield networks (Hopfield, 1982). A detailed derivation and analysis of modern Hopfield networks is given in our companion paper (Ramsauer et al., 2020). These novel continuous state Hopfield networks allow to store and retrieve exponentially (in the dimension of the space) many patterns (see next Section). Thus, modern Hopfield networks with their update rule, which are used as an attention mechanism in the transformer, enable immune repertoire classification in computational biology.

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 receptors, represented as amino acid sequences. Usually, the presence of only a small fraction of particular receptors determines the immune status with respect to a particular disease (Christophersen et al., 2014; Emerson et al., 2017). This is because the immune system has already acquired a resistance if one or few particular immune receptors that can bind to the disease agent are present. Therefore, classification of immune repertoires bears a high difficulty since each immune repertoire can contain millions of sequences as instances with only a few indicating the class. Further properties of the data that complicate

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

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 determine whether binding to a 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 an extremely low witness rate and large numbers of instances, which represents a challenge for currently available machine learning methods. Furthermore, the methods should ideally be interpretable, since the extraction of class-associated sequence motifs is desired to gain crucial biological insights.

The acquisition of human immune repertoires has been enabled by immunosequencing technology (Georgiou et al., 2014; Brown et al., 2019) which allows to obtain the immune receptor sequences and immune repertoires of individuals. 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 and the individual's immune status. 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 working 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 insights 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).

Immunosequencing data have 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 available machine learning methods for immune receptor data has been focusing 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 T-cell receptor (TCR) sequences (specifically, binding of TCR sequences to peptide-MHC complexes) and demonstrated that motifs can be extracted from these models. Similarly, Konishi et al. (2019) use CNNs, gradient boosting, and other machine learning techniques on B-cell receptor (BCR) sequences to distinguish tumor tissue from normal tissue. However, the methods presented so far predict a particular class, the epitope, based on a single input sequence.

Immune repertoire classification has been considered as a MIL problem in the following publications. 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 prediction 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 Table 1).

Our proposed method, DeepRC, also uses a MIL approach but considers sequences rather than k-mers as instances within an input object and a transformer-like attention mechanism. DeepRC sets out to avoid the above-mentioned constraints of current methods by (a) applying transformer-like attention-pooling instead of max-pooling and learning a classifier on the repertoire rather than on the sequence-representation, (b) pooling learned representations rather than predictions, and (c) using less rigid feature extractors, such as 1D convolutions or LSTMs. *In this work, we contribute the following:* We demonstrate that continuous generalizations of binary modern Hopfield-networks (Krotov &

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

Hopfield, 2016, 2018; Demircigil et al., 2017) have an update rule that is known as the attention mechanisms in the transformer. We show that these modern Hopfield networks have exponential storage capacity, which allows them to extract patterns among a large set of instances (next Section). Based on this result, we propose DeepRC, a novel deep MIL method based on modern Hopfield networks for large bags of complex sequences, as they occur in immune repertoire classification (Section "Deep Repertoire Classification). We evaluate the predictive performance of DeepRC and other machine learning approaches for the classification of immune repertoires in a large comparative study (Section "Experimental Results")

Exponential storage capacity of continuous state modern Hopfield networks with transformer attention as update rule

In this section, we show that modern Hopfield networks have exponential storage capacity, which will later allow us to approach massive multiple-instance learning problems, such as immune repertoire classification. See our companion paper (Ramsauer et al., 2020) for a detailed derivation and analysis of modern Hopfield networks.

We assume patterns $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^d$ that are stacked as columns to the matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ and a query pattern $\boldsymbol{\xi}$ that also represents the current state. The largest norm of a pattern is $M = \max_i \|\boldsymbol{x}_i\|$. The *separation* Δ_i of a pattern $\boldsymbol{x}_i$ is defined as its minimal dot product difference to any of the other patterns: $\Delta_i = \min_{j, j \neq i} \left(\boldsymbol{x}_i^T \boldsymbol{x}_i - \boldsymbol{x}_i^T \boldsymbol{x}_j \right)$. A pattern is *well-separated* from the data if $\Delta_i \geq \frac{2}{\beta N} + \frac{1}{\beta} \log \left(2(N-1) N \beta M^2 \right)$. We consider a modern Hopfield network with current state $\boldsymbol{\xi}$ and the energy function $\mathrm{E} = -\beta^{-1} \log \left(\sum_{i=1}^N \exp(\beta \boldsymbol{x}_i^T \boldsymbol{\xi}) \right) + \beta^{-1} \log N + \frac{1}{2} \boldsymbol{\xi}^T \boldsymbol{\xi} + \frac{1}{2} M^2$. For energy E and state $\boldsymbol{\xi}$, the update rule

$$\boldsymbol{\xi}^{\text{new}} = f(\boldsymbol{\xi}; \boldsymbol{X}, \beta) = \boldsymbol{X}\, \boldsymbol{p} = \boldsymbol{X} \operatorname{softmax}(\beta \boldsymbol{X}^T \boldsymbol{\xi}) \tag{1}$$

is proven to converge globally to stationary points of the energy E, which are local minima or saddle points (see (Ramsauer et al., 2020), appendix, Theorem A2). *Surprisingly, the update rule Eq.* (1) *is also the formula of the well-known transformer attention mechanism.*

To see this more clearly, we simultaneously update several queries $\boldsymbol{\xi}_i$. Furthermore the queries $\boldsymbol{\xi}_i$ and the patterns $\boldsymbol{x}_i$ are linear mappings of vectors $\boldsymbol{y}_i$ into the space $\mathbb{R}^d$. For matrix notation, we set $\boldsymbol{x}_i = \boldsymbol{W}_K^T \boldsymbol{y}_i$, $\boldsymbol{\xi}_i = \boldsymbol{W}_Q^T \boldsymbol{y}_i$ and multiply the result of our update rule with $\boldsymbol{W}_V$. Using $\boldsymbol{Y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N)^T$, we define the matrices $\boldsymbol{X}^T = \boldsymbol{K} = \boldsymbol{Y} \boldsymbol{W}_K$, $\boldsymbol{Q} = \boldsymbol{Y} \boldsymbol{W}_Q$, and $\boldsymbol{V} = \boldsymbol{Y} \boldsymbol{W}_K \boldsymbol{W}_V = \boldsymbol{X}^T \boldsymbol{W}_V$, where $\boldsymbol{W}_K \in \mathbb{R}^{d_y \times d_k}$, $\boldsymbol{W}_Q \in \mathbb{R}^{d_y \times d_k}$, $\boldsymbol{W}_V \in \mathbb{R}^{d_k \times d_v}$, $\boldsymbol{K} \in \mathbb{R}^{N \times d_k}$, $\boldsymbol{Q} \in \mathbb{R}^{N \times d_k}$, $\boldsymbol{V} \in \mathbb{R}^{N \times d_v}$, and the patterns are now mapped to the Hopfield space with dimension $d = d_k$. We set $\beta = 1/\sqrt{d_k}$ and change $\operatorname{softmax}$ to a row vector. The update rule Eq. (1) multiplied by $\boldsymbol{W}_V$ performed for all queries simultaneously becomes in row vector notation:

$$\operatorname{att}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}; \beta) = \operatorname{softmax}\left(\beta\, \boldsymbol{Q}\, \boldsymbol{K}^T \right) \boldsymbol{V} = \operatorname{softmax}\left(\left(1/\sqrt{d_k} \right) \boldsymbol{Q}\, \boldsymbol{K}^T \right) \boldsymbol{V} \,. \tag{2}$$

This formula is the transformer attention.

If the patterns $\boldsymbol{x}_i$ are well separated, the iterate Eq. (1) converges to a fixed point close to a pattern to which the initial $\boldsymbol{\xi}$ is similar. If the patterns are not well separated the iterate Eq.(1) converges to a fixed point close to the arithmetic mean of the patterns. If some patterns are similar to each other but well separated from all other vectors, then a *metastable state* between the similar patterns exists. Iterates that start near a metastable state converge to this metastable state. For details see Ramsauer et al. (2020), appendix, Sect. A2. Typically, the update converges after one update step (see Ramsauer et al. (2020), appendix, Theorem A8) and has an exponentially small retrieval error (see Ramsauer et al. (2020), appendix, Theorem A9).

Our main concern for application to immune repertoire classification is the number of patterns that can be stored and retrieved by the modern Hopfield network, equivalently to the transformer attention head. The storage capacity of an attention mechanism is critical for massive MIL problems. We first define what we mean by storing and retrieving patterns from the modern Hopfield network.

Definition 1 (Pattern Stored and Retrieved)**.** *We assume that around every pattern $\boldsymbol{x}_i$ a sphere S_i is given. We say $\boldsymbol{x}_i$* is stored *if there is a single fixed point $\boldsymbol{x}_i^* \in \mathrm{S}_i$ to which all points $\boldsymbol{\xi} \in \mathrm{S}_i$ converge,*

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

and $\mathrm{S}_i \cap \mathrm{S}_j = \emptyset$ *for* $i \neq j$*. We say* $\boldsymbol{x}_i$ is retrieved *if the iteration Eq.* (1) *converged to the single fixed point* $\boldsymbol{x}_i^* \in \mathrm{S}_i$*.*

For randomly chosen patterns, the number of patterns that can be stored is exponential in the dimension d of the space of the patterns ($\boldsymbol{x}_i \in \mathbb{R}^d$).

Theorem 1. *We assume a failure probability* $0 < p \leqslant 1$ *and randomly chosen patterns on the sphere with radius* $M = K\sqrt{d-1}$*. We define* $a := \frac{2}{d-1}\left(1 + \ln(2\,\beta\,K^2\,p\,(d-1))\right)$*,* $b := \frac{2\,K^2\,\beta}{5}$*, and* $c = \frac{b}{W_0(\exp(a + \ln(b))}$*, where* W_0 *is the upper branch of the Lambert* W *function and ensure* $c \geq \left(\frac{2}{\sqrt{p}}\right)^{\frac{4}{d-1}}$*. Then with probability* $1-p$*, the number of random patterns that can be stored is*

$$N \geq \sqrt{p}\, c^{\frac{d-1}{4}} \,. \tag{3}$$

Examples are $c \geq 3.1546$ *for* $\beta = 1$*,* $K = 3$*,* $d = 20$ *and* $p = 0.001$ *(*$a + \ln(b) > 1.27$*) and* $c \geq 1.3718$ *for* $\beta = 1$ $K = 1$*,* $d = 75$*, and* $p = 0.001$ *(*$a + \ln(b) < -0.94$*).*

See Ramsauer et al. (2020), appendix, Theorem A5 for a proof. We have established that a modern Hopfield network or a transformer attention mechanism can store and retrieve exponentially many patterns. This allows us to approach MIL with massive numbers of instances from which we have to retrieve a few with an attention mechanism.

Deep Repertoire Classification

Problem setting and notation. We consider a MIL problem, in which an input object X is a *bag* of N instances $X = \{s_1, \ldots, s_N\}$. The instances do not have dependencies nor orderings between them and N can be different for every object. We assume that each instance s_i is associated with a label $y_i \in \{0, 1\}$, assuming a binary classification task, to which we do not have access. We only have access to a label $Y = \max_i y_i$ 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 and that the relation between instance-label and bag-label can be more complex (Foulds & Frank, 2010).

A model $\hat{y} = g(X)$ should be (a) invariant to permutations of the instances and (b) able to cope with the fact that N 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 $\boldsymbol{z}_i$ of each instance by $h : \mathcal{S} \mapsto \mathbb{R}^{d_v}$ and then encode 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, the pooling of representations rather than scoring functions, is currently best performing (Wang et al., 2018).

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_i from a space $s_i \in \mathcal{S}$. These sequences act as the instances in the MIL problem. Although immune repertoire classification can readily be formulated as a MIL problem, it is yet unclear how well machine learning methods solve the above-described problem with a large number of instances $N \gg 10,000$ and with instances s_i being complex sequences. Next we describe currently used pooling functions for MIL 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: $\boldsymbol{z}_i = h_{\boldsymbol{\theta}}(s_i)$ and then a pooling function $\boldsymbol{z} = f(\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\})$ supplies a representation $\boldsymbol{z}$ of the input object $X = \{s_1, \ldots, s_N\}$. The following pooling functions are typically used: *average-pooling*: $\boldsymbol{z} = \frac{1}{N}\sum_{i=1}^N \boldsymbol{z}_i$, *max-pooling*: $\boldsymbol{z} = \sum_{m=1}^{d_v} \boldsymbol{e}_m(\max_{i,1\leqslant i\leqslant N}\{z_{im}\})$, where $\boldsymbol{e}_m$ is the standard basis vector for dimension m and *attention-pooling*: $\boldsymbol{z} = \sum_{i=1}^N a_i \boldsymbol{z}_i$, where a_i are non-negative ($a_i \geq 0$), sum to one ($\sum_{i=1}^N a_i = 1$), and are determined by an attention mechanism. These pooling functions are invariant to permutations

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

of $\{1, \ldots, N\}$ and are differentiable. Therefore, they are suited as building blocks for Deep Learning architectures. We employ attention-pooling in our DeepRC model as detailed in the following.

Modern Hopfield networks viewed as transformer-like attention mechanisms. The modern Hopfield networks, as introduced above,have a storage capacity that is exponential in the dimension of the vector space and converge after just one update (see (Ramsauer et al., 2020), appendix).Additionally, the update rule of modern Hopfield networks is known as key-value attention mechanism, which has been highly successful through the transformer (Vaswani et al., 2017) and BERT (Devlin et al., 2019) models in natural language processing. Therefore using modern Hopfield networks with the key-value-attention mechanism as update rule is the natural choice for our task. In particular, modern Hopfield networks are theoretically justified for storing and retrieving the large number of vectors (sequence patterns) that appear in the immune repertoire classification task.

Instead of using the terminology of modern Hopfield networks, we explain our DeepRC architecture in terms of key-value-attention (the update rule of the modern Hopfield network), since it is well known in the deep learning community. The attention mechanism assumes a space of dimension d_k in which keys and queries are compared. A set of N key vectors are combined to the matrix $\boldsymbol{K}$. A set of d_q query vectors are combined to the matrix $\boldsymbol{Q}$. Similarities between queries and keys are computed by inner products, therefore queries can search for similar keys that are stored. Another set of N value vectors are combined to the matrix $\boldsymbol{V}$. 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 $\mathrm{att}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}; \beta)$ maps queries $\boldsymbol{Q}$, keys $\boldsymbol{K}$, and values $\boldsymbol{V}$ to d_v-dimensional outputs: $\mathrm{att}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}; \beta) = \mathrm{softmax}(\beta \boldsymbol{Q}\boldsymbol{K}^T)\boldsymbol{V}$ (see also Eq. (2)). While this attention mechanism has originally been developed for sequence tasks (Vaswani et al., 2017), it can be readily transferred to sets (Lee et al., 2019; Ye et al., 2018). This type of attention mechanism will be employed in DeepRC.

The DeepRC method. 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 $\boldsymbol{z}_i$ to denote the *sequence-representation* of the i-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: Sect. A2; DeepRC variations: Sect. A10.)

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 $\boldsymbol{z}_i = h_1(s_i)$, which acts as the values $\boldsymbol{V} = \boldsymbol{Z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N)$ 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 $\boldsymbol{K} \in \mathbb{R}^{N \times d_k}$ 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_k-dimensional query 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 (Eq. (2)) of DeepRC is implemented as follows:

$$\boldsymbol{z} = \mathrm{att}(\boldsymbol{\xi}^T, \boldsymbol{K}, \boldsymbol{Z}; \frac{1}{\sqrt{d_k}}) = \mathrm{softmax}\left(\frac{\boldsymbol{\xi}^T \boldsymbol{K}^T}{\sqrt{d_k}}\right) \boldsymbol{Z}, \tag{4}$$

where $\boldsymbol{Z} \in \mathbb{R}^{N \times d_v}$ 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 $\boldsymbol{z}_i$. The attention mechanism can readily be extended to multiple queries, however, computational demand could constrain this depending on the application and dataset. Theorem 1 demonstrates that this mechanism is able to retrieve a single pattern out of several hundreds of thousands.

Attention-pooling and interpretability. Each input object, i.e. repertoire, consists of a large number N 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

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

mechanism adapted to sets is realized in DeepRC which supplies a_i – the importance of the sequence s_i. This importance value is an interpretable quantity, which is highly desired for the immunological problem at hand. Thus, DeepRC allows for two forms of interpretability methods. (a) A trained DeepRC model can compute attention weights a_i, which directly indicate the importance of a sequence. (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). See Sect. A8 for details.

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 predicts a single target, the class label y, e.g. the immune status of an immune repertoire, using one output value. 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 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 of 4 and dropout of input sequences.

Implementation. To reduce computational time, the attention network first computes the attention weights a_i for each sequence s_i in a repertoire. Subsequently, the top 10% of sequences with the highest a_i per repertoire are used to compute the weight updates and prediction. Furthermore, computation of $\boldsymbol{z}_i$ is performed in 16-bit, others in 32-bit precision to ensure numerical stability in the softmax. See Sect. A2 for details.

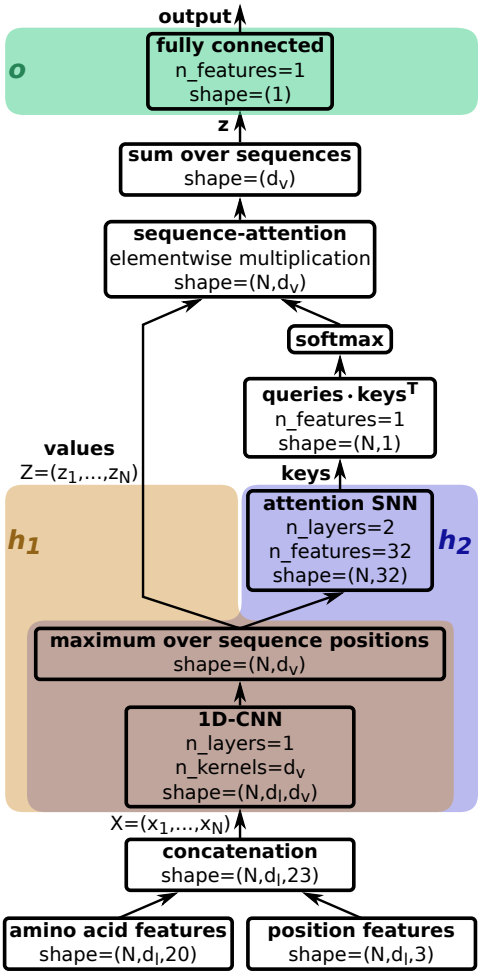

Figure 2: DeepRC architecture as used in Table 1 with sub-networks h_1, h_2, and o. d_l indicates the sequence length.

Experimental Results

In this section, we report and analyze the predictive power of DeepRC and the compared methods on several immunosequencing datasets. The ROC-AUC is used as the main metric for the predictive power.

Methods compared. We compared previous methods for immune repertoire classification, (Ostmeyer et al., 2019) ("Log. MIL (KMER)", "Log. MIL (TCRB)") and a burden test (Emerson et al., 2017), as well as the baseline methods Logistic Regression ("Log. Regr."), k-nearest neighbour ("KNN"), and Support Vector Machines ("SVM") with kernels designed for sets, such as the Jaccard kernel ("J") and the MinMax ("MM") kernel (Ralaivola et al., 2005). For the simulated data, we also added baseline methods that search for the implanted motif either in binary or continuous fashion ("Known motif b.", "Known motif c.") assuming that this motif was known (for details, see Sect. A4).

Datasets. We aimed at constructing immune repertoire classification scenarios with varying degree of difficulties and realism in order to compare and analyze the suggested machine learning methods. To this end, we either use simulated or experimentally-observed immune receptor sequences and we implant signals, specifically, sequence motifs or sets thereof (Akbar et al., 2019; Weber et al., 2020),

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

at different frequencies into sequences of repertoires of the positive class. These frequencies represent the *witness rates* and range from 0.01% to 10%. Overall, we compiled four categories of datasets: (a) simulated immunosequencing data with implanted signals, (b) LSTM-generated immunosequencing data with implanted signals, (c) real-world immunosequencing data with implanted signals, and (d) real-world immunosequencing data with known immune status, the *CMV dataset* (Emerson et al., 2017). The average number of instances per bag, which is the number of sequences per immune repertoire, is ≈300,000 except for category (c), in which we consider the scenario of low-coverage data with only 10,000 sequences per repertoire. The number of repertoires per dataset ranges from 785 to 5,000. In total, all datasets comprise ≈30 billion sequences or instances. This represents the largest comparative study on immune repertoire classification (see Sect. A3).

Hyperparameter selection. We used a nested 5-fold cross validation (CV) procedure to estimate the performance of each of the methods. All methods could adjust their most important hyperparameters on a validation set in the inner loop of the procedure. See Sect. A5 for details.

	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** ± 0.022	**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
Burden test	0.699 ± 0.041	**1.00** ± 0.00	0.64± 0.05	**1.00**± 0.00	0.89±0.02	**1.00**± 0.00	**1.00**± 0.00	**1.00**± 0.00	**1.00**± 0.00	0.79± 0.28	0.549± 0.074
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 (TCRβ)	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 1: Results in terms of AUC of the competing methods on all datasets. The reported errors are standard deviations across 5 cross-validation (CV) folds (except for the column "Simulated"). **Real-world CMV:** Average performance over 5 CV folds on the *CMV dataset* (Emerson et al., 2017). **Real-world data with implanted signals:** Average performance over 5 CV folds for each of the four datasets. A signal was implanted with a frequency (=witness rate) of 1% or 0.1%. Either a single motif ("OM") or multiple motifs ("MM") were implanted. **LSTM-generated data:** Average performance over 5 CV folds for each of the 5 datasets. In each dataset, a signal was implanted with a frequency of 10%, 1%, 0.5%, 0.1%, or 0.05%, respectively. **Simulated:** Here we report the mean over 18 simulated datasets with implanted signals and varying difficulties (see Tab. A9 for details). The error reported is the standard deviation of the AUC values across the 18 datasets.

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. Across categories, the runner-up methods are either the SVM for MIL problems with MinMax kernel or the burden test (see Table 1 and Sect. A6).

Results on simulated immunosequencing data. In this setting the complexity of the implanted signal is in focus and varies throughout 18 simulated datasets (see Sect. A3). Some datasets are challenging for the methods because the implanted motif is hidden by noise and others because only a small fraction of sequences carries the motif, and hence have a low witness rate. 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 1.000 in several datasets to an AUC of 0.532 in dataset '17' (see Sect. A6). DeepRC outperforms all other methods with an average AUC of 0.846 ± 0.223, followed by the SVM with MinMax kernel with an average AUC of 0.827 ± 0.210 (see Sect. A6). The predictive performance of all methods suffers if the signal occurs only in an extremely small fraction of sequences. In datasets, in which only 0.01% of the sequences carry the motif, all AUC values are below 0.550. *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 Sect. A6 and A7). The second best method, SVM with MinMax kernel, has a similar predictive

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

performance to DeepRC on all datasets but the other competing methods have a lower predictive performance on datasets with low frequency of the signal (0.05%). *Results on real-world data with implanted motifs.* In this dataset category, we used real immunosequences 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 burden test with an average AUC of 0.883 ± 0.170. Notably, all methods except for DeepRC have difficulties with noisy motifs at a frequency of 0.1% (see Tab. A11). *Results on real-world data.* On the real-world dataset, in which the immune status of persons affected by the cytomegalovirus has to be predicted, the competing methods yield predictive AUCs between 0.515 and 0.825 (see Table 1). We note that this dataset is not the exact dataset that was used in Emerson et al. (2017). It differs in pre-processing and also comprises a different set of samples and a smaller training set due to the nested 5-fold cross-validation procedure, which leads to a more challenging dataset. 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 the burden test with an AUC of 0.699 ± 0.041. The top-ranked sequences by DeepRC significantly correspond to those detected by Emerson et al. (2017), which we tested by a Mann-Whitney 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 of $1.3 \cdot 10^{-93}$). The sequence attention values are displayed in Tab. A14.

Conclusion. We have demonstrated how modern Hopfield networks and attention mechanisms enable successful classification of the immune status of immune repertoires. For this task, methods have to identify the discriminating sequences amongst a large set of sequences in an immune repertoire. Specifically, even motifs within those sequences have to be identified. We have shown that DeepRC, a modern Hopfield network and an attention mechanism with a fixed query, can solve this difficult task despite the massive number of instances. DeepRC furthermore outperforms the compared methods across a range of different experimental conditions.

Broader Impact

Impact on machine learning and related scientific fields. We envision that with (a) the increasing availability of large immunosequencing datasets (Kovaltsuk et al., 2018; Corrie et al., 2018; Christley et al., 2018; Zhang et al., 2020; Rosenfeld et al., 2018; Shugay et al., 2018), (b) further fine-tuning of ground-truth benchmarking immune receptor datasets (Weber et al., 2020; Olson et al., 2019; Marcou et al., 2018), (c) accounting for repertoire-impacting factors such as age, sex, ethnicity, and environment (potential confounding factors), and (d) 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 (Raybould et al., 2020; Minervina et al., 2020; Galson et al., 2020). Such results would greatly benefit ongoing research on antibody and TCR-driven immunotherapies and immunodiagnostics as well as rational vaccine design (Brown et al., 2019).

In the course of this development, the experimental verification and interpretation of machine-learning-identified motifs could receive additional focus, as for most of the sequences within a repertoire the corresponding antigen is unknown. Nevertheless, recent technological breakthroughs in high-throughput antigen-labeled immunosequencing are beginning to generate large-scale antigen-labeled single-immune-receptor-sequence data thus resolving this longstanding problem (Setliff et al., 2019).

From a machine learning perspective, the successful application of DeepRC on immune repertoires with their large number of instances per bag might encourage the application of modern Hopfield networks and attention mechanisms on new, previously unsolved or unconsidered, datasets and problems.

Impact on society. If the approach proves itself successful, it could lead to faster testing of individuals for their immune status w.r.t. a range of diseases based on blood samples. This might motivate changes in the pipeline of diagnostics and tracking of diseases, e.g. automated testing of the immune status in regular intervals. It would furthermore make the collection and screening of blood samples for larger databases more attractive. In consequence, the improved testing of immune statuses might identify individuals that do not have a working immune response towards certain diseases to government or insurance companies, which could then push for targeted immunisation of the individual. Similarly to compulsory vaccination, such testing for the immune status could be made

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

compulsory by governments, possibly violating privacy or personal self-determination in exchange for increased over-all health of a population.

Ultimately, if the approach proves itself successful, the insights gained from the screening of individuals that have successfully developed resistances against specific diseases could lead to faster targeted immunisation, once a certain number of individuals with resistances can be found. This might strongly decrease the harm done by e.g. pandemics and lead to a change in the societal perception of such diseases.

Consequences of failures of the method. As common with methods in machine learning, potential danger lies in the possibility that users rely too much on our new approach and use it without reflecting on the outcomes. However, the full pipeline in which our method would be used includes wet lab tests after its application, to verify and investigate the results, gain insights, and possibly derive treatments. Failures of the proposed method would lead to unsuccessful wet lab validation and negative wet lab tests. Since the proposed algorithm does not directly suggest treatment or therapy, human beings are not directly at risk of being treated with a harmful therapy. Substantial wet lab and in-vitro testing and would indicate wrong decisions by the system.

Leveraging of biases in the data and potential discrimination. As for almost all machine learning methods, confounding factors, such as age or sex, could be used for classification. This, might lead to biases in predictions or uneven predictive performance across subgroups. As a result, failures in the wet lab would occur (see paragraph above). Moreover, insights into the relevance of the confounding factors could be gained, leading to possible therapies or counter-measures concerning said factors.

Furthermore, the amount of data available with respec to relevant confounding factors could lead to better or worse performance of our method. E.g. a dataset consisting mostly of data from individuals within a specific age group might yield better performance for that age group, possibly resulting in better or exclusive treatment methods for that specific group. Here again, the application of DeepRC would be followed by in-vitro testing and development of a treatment, where all target groups for the treatment have to be considered accordingly.

Availability

All datasets and code is available at https://github.com/ml-jku/DeepRC. The CMV dataset is publicly available at https://clients.adaptivebiotech.com/pub/Emerson-2017-NatGen.

Acknowledgments

The ELLIS Unit Linz, the LIT AI Lab and the Institute for Machine Learning are supported by the Land Oberösterreich, LIT grants DeepToxGen (LIT-2017-3-YOU-003), and AI-SNN (LIT-2018-6-YOU-214), the Medical Cognitive Computing Center (MC3), Janssen Pharmaceutica, UCB Biopharma, Merck Group, Audi.JKU Deep Learning Center, Audi Electronic Venture GmbH, TGW, Primal, Silicon Austria Labs (SAL), FILL, EnliteAI, Google Brain, ZF Friedrichshafen AG, Robert Bosch GmbH, TÜV Austria, DCS, and the NVIDIA Corporation. Victor Greiff (VG) and Geir Kjetil Sandve (GKS) are supported by The Helmsley Charitable Trust (#2019PG-T1D011, to VG), UiO World-Leading Research Community (to VG), UiO:LifeSciences Convergence Environment Immunolingo (to VG and GKS), EU Horizon 2020 iReceptorplus (#825821, to VG) and Stiftelsen Kristian Gerhard Jebsen (K.G. Jebsen Coeliac Disease Research Centre, to GKS).

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

Appendix

In the following, the appendix to the paper "Modern Hopfield Networks and Attention for Immune Repertoire Classification" is presented. Here we provide details on DeepRC, the compared methods, and the experimental setup and results. Furthermore, the generation of the immune repertoire classification data using an LSTM network, the interpretation of DeepRC and the extraction of found motifs, and the ablation study using different variants of DeepRC are described.

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

Contents of Appendix

1 Immune Repertoire Classification . . . 13
- A1 Introduction . . . 13
- A2 DeepRC implementation details . . . 13
- A3 Datasets . . . 15
  - A3.1 Simulated immunosequencing data . . . 15
  - A3.2 LSTM-generated data . . . 16
  - A3.3 Real-world data with implanted signals . . . 16
  - A3.4 Real-world data: CMV dataset . . . 17
  - A3.5 Comparison to other MIL datasets . . . 17
- A4 Compared methods . . . 19
  - A4.1 Known motif . . . 19
  - A4.2 Support Vector Machine (SVM) . . . 19
  - A4.3 K-Nearest Neighbor (KNN) . . . 19
  - A4.4 Logistic regression . . . 20
  - A4.5 Burden test . . . 20
  - A4.6 Logistic MIL (Ostmeyer et al) . . . 20
- A5 Hyperparameter selection . . . 21
- A6 Results . . . 23
- A7 Repertoire generation via LSTM . . . 26
- A8 Interpreting DeepRC . . . 28
- A9 Attention values for previously associated CMV sequences . . . 31
- A10 DeepRC variations and ablation study . . . 32

List of figures

- A1 Position encoding . . . 14
- A2 Distribution of AAs and k-mers . . . 27
- A3 Interpretation of the DeepRC classifier . . . 29
- A4 Visualization of the contributions of AA . . . 30

List of tables

- A1 Properties of simulated repertoires, variations of motifs, and motif frequencies . . . 17
- A2 Overview of MIL datasets and their characteristics . . . 18
- A3 DeepRC hyperparameter search space . . . 21
- A5 Hyperparameter search of the KNN baseline . . . 21
- A6 Hyperparameter search of the logistic regression . . . 22
- A7 Hyperparameter search of the burden test . . . 22
- A8 Hyperparameter search of the logistic MIL baseline . . . 22
- A9 AUC estimates for all 18 datasets in "simulated immunosequencing data" . . . 23
- A10 AUC estimates for all 5 datasets in “LSTM-generated data” . . . 24
- A11 AUC estimates for all 4 datasets in “real-world data with implanted signals” . . . 25
- A12 Results on the CMV dataset given by AUC, F1 score, balanced accuracy, and accuracy 25
- A13 Visualization of extracted motifs . . . 28
- A14 TCRβ sequences re-discovered by DeepRC . . . 31
- A15 Hyperparameter search space for DeepRC variations . . . 32
- A16 Hyperparameter search space for DeepRC variations with LSTM embedding . . . 33
- A17 Impact of hyperparameters on DeepRC with LSTM . . . 33
- A18 Impact of hyperparameters on DeepRC with 1D CNN . . . 34

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

Immune Repertoire Classification

Michael Widrich Bernhard Schäfl Milena Pavlović Geir Kjetil Sandve Sepp Hochreiter Victor Greiff Günter Klambauer

A1 Introduction

In Section A2 we provide details on the architecture of DeepRC, in Section A3 we present details on the datasets, in Section A4 we explain the methods that we compared, in Section A5 we elaborate on the hyperparameters and their selection process. Then, in Section A6 we present detailed results for each dataset category in tabular form, in Section A7 we provide information on the LSTM model that was used to generate antibody sequences, in Section A8 we show how DeepRC can be interpreted, in Section A9 we show the correspondence of previously identified TCR sequences for CMV immune status with attention values by DeepRC, and finally we present variations and an ablation study of DeepRC in Section A10.

A2 DeepRC implementation details

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

1D CNN for motif recognition. In the following, we describe how DeepRC identifies patterns in the individual sequences and reduces each sequence in the input object to a fixed-size feature vector. DeepRC employs 1D convolution layers to extract patterns, where trainable weight kernels are convolved over the sequence positions. In principle, also recurrent neural networks (RNNs) or transformer networks could be used instead of 1D CNNs, however, (a) the computational complexity of the network must be low to be able to process millions of sequences for a single update. Additionally, (b) the learned network should be able to provide insights in the recognized patterns in form of motifs. Both properties (a) and (b) are fulfilled by 1D convolution operations that are used by DeepRC.

We use one 1D CNN layer (Hu et al., 2014) with SELU activation functions (Klambauer et al., 2017) to identify the relevant patterns in the input sequences with a computationally light-weight

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

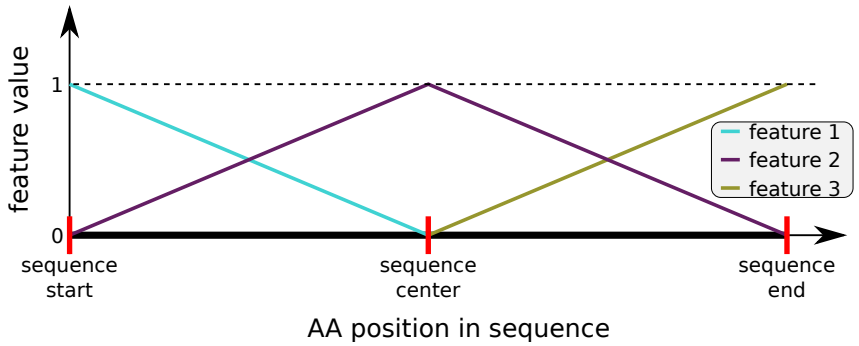

Figure A1: We use 3 input features with values in range $[0, 1]$ to encode the relative position of each AA in a sequence with respect to the sequence. "feature 1" encodes if an AA is close to the sequence start, "feature 2" to the sequence center, and "feature 3" to the sequence end. For every position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

operation. The larger the kernel size, the more surrounding sequence positions are taken into account, which influences the length of the motifs that can be extracted. We therefore adjust the kernel size during hyperparameter search. In prior works (Ostmeyer et al., 2019), a k-mer size of 4 yielded good predictive performance, which could indicate that a kernel size in the range of 4 may be a proficient choice. For d_v trainable kernels, this produces a feature vector of length d_v at each sequence position. Subsequently, global max-pooling over all sequence positions of a sequence reduces the sequence-representations $\boldsymbol{z}_i$ to vectors of the fixed length d_v. Given the challenging size of the input data per repertoire, the computation of the CNN activations and weight updates is performed using 16-bit floating point values. A list of hyperparameters evaluated for DeepRC is given in Table A3. A comparison of RNN-based and CNN-based sequence embedding for motif recognition in a smaller experimental setting is given in Sec. A10.

Regularization. We apply random and attention-based subsampling of repertoire sequences to reduce over-fitting and decrease computational effort. 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 only partly investigated further in a smaller experimental setting in Sec. A10 due to high computational demands. Such regularization techniques include $l1$ and $l2$ weight decay, noise in the form of random AA permutations in the input sequences, noise on the attention weights, or random shuffling of sequences between repertoires that belong to the negative class. The last regularization technique assumes that the sequences in positive-class repertoires carry a signal, such as an AA motif corresponding to an immune response, whereas the sequences in negative-class repertoires do not. Hence, the sequences can be shuffled randomly between negative class repertoires without obscuring the signal in the positive class repertoires.

Hyperparameters. For the hyperparameter search of DeepRC for the category "simulated immunosequencing data", we only conducted a full hyperparameter search on the more difficult datasets with motif implantation probabilities below 1%, as described in Table A3. This process was repeated for all 5 folds of the 5-fold cross-validation (CV) and the average score on the 5 test sets constitutes the final score of a method.

Table A3 provides an overview of the hyperparameter search, which was conducted as a grid search for each of the datasets in a nested 5-fold CV procedure, as described in Section A4.

Computation time and optimization. We took measures on the implementation level to address the high computational demands, especially GPU memory consumption, in order to make the large number of experiments feasible. We train the DeepRC model with a small batch size of 4 samples and

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

perform computation of inference and updates of the 1D CNN using 16-bit floating point values. The rest of the network is trained using 32-bit floating point values. The Adam parameter for numerical stability was therefore increased from the default value of $\epsilon = 10^{-8}$ to $\epsilon = 10^{-4}$. Training was performed on various GPU types, mainly `NVIDIA RTX 2080 Ti`. Computation times were highly dependent on the number of sequences in the repertoires and the number and sizes of CNN kernels. A single update on an `NVIDIA 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 (≥ 16 GB) into account, it is already possible to train DeepRC, possibly with multi-head attention and a larger network architecture, on larger datasets (see Sec. A10). Our network implementation is based on `PyTorch 1.3.1` (Paszke et al., 2019).

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

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

A3 Datasets

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

A3.1 Simulated immunosequencing data

In the first category, we aim at investigating the impact of the signal frequency, i.e. the WR, and the signal complexity on the performance of the different methods. To this end, we created 18 datasets, whereas each dataset contains a large number of repertoires with a large number of random AA sequences per repertoire. We then implanted signals in the AA sequences of the positive class

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

repertoires, where the 18 datasets differ in frequency and complexity of the implanted signals. In detail, the AAs were sampled randomly independent of their respective position in the sequence, while the frequencies of AAs, distribution of sequence lengths, and distribution of the number of sequences per repertoire, i.e. the number of instances per bag, are following the respective distributions observed in the real-world *CMV dataset* (Emerson et al., 2017). For this, we first sampled the number of sequences for a repertoire from a Gaussian $\mathcal{N}(\mu = 316k, \sigma = 132k)$ distribution and rounded to the nearest positive integer. We re-sampled if the size was below $5k$. We then generated random sequences of AAs with a length of $\mathcal{N}(\mu = 14.5, \sigma = 1.8)$, again rounded to the nearest positive integers. Each simulated repertoire was then randomly assigned to either the positive or negative class, with $2,500$ repertoires per class. In the repertoires assigned to the positive class, we implanted motifs with an average length of 4 AAs, following the results of the experimental analysis of antigen-binding motifs in antibodies and T-cell receptor sequences by (Akbar et al., 2019). We varied the characteristics of the implanted motifs for each of the 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, which is the witness rate. (b) The number of wildcard positions in the motif. A wildcard position contains a random AA, which is randomly sampled for each sequence. Wildcard positions are located in the center of the implanted motif. (c) The number of deletion positions in the implanted motif. A deletion position has a probability of 0.5 of being removed from the motif. Deletion positions are located in the center of the implanted motifs.

In this way, we generated 18 different datasets of variable difficulty containing in total roughly 28.7 billion sequences. See Table A1 for an overview of the properties of the implanted motifs in the 18 datasets.

A3.2 LSTM-generated data

In the second dataset category, we investigate the impact of the signal frequency and complexity in combination with more plausible immune receptor sequences by taking into account the positional AA distributions and other sequence properties. To this end, we trained an LSTM (Hochreiter & Schmidhuber, 1997) in a standard next character prediction (Graves, 2013) setting to create AA sequences with properties similar to experimentally observed immune receptor sequences.

In the first step, the LSTM model was trained on all immuno-sequences in the *CMV dataset* (Emerson et al., 2017) that contain valid information about sequence abundance and have a known CMV label. Such an LSTM model is able to capture various properties of the sequences, including position-dependent probability distributions and combinations, relationships, and order of AAs. We then used the trained LSTM model to generate $1,000$ repertoires in an autoregressive fashion, starting with a start sequence that was randomly sampled from the trained-on dataset. Based on a visual inspection of the frequencies of 4-mers (see Section A7), the similarity of LSTM generated sequences and real sequences was deemed sufficient for the purpose of generating the AA sequences for the datasets in this category. Further details on LSTM training and repertoire generation are given in Section A7.

After generation, each repertoire was assigned to either the positive or negative class, with 500 repertoires per class. We implanted motifs of length 4 with varying properties in the center of the sequences of the positive class to obtain 5 different datasets. Each sequence in the positive repertoires has a probability ρ to carry the motif, which was varied throughout 5 datasets and corresponds to the WR (see Table A1). 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.

A3.3 Real-world data with implanted signals

In the third category, we implanted signals into experimentally obtained immuno-sequences, where we considered 4 dataset variations. Each dataset consists of 750 repertoires for each of the two classes, where each repertoire consists of $10k$ sequences. In this way, we aim to simulate datasets with a *low sequencing coverage*, which means that only relatively few sequences per repertoire are available. The sequences were randomly sampled from healthy (CMV negative) individuals from the *CMV dataset* (see below paragraph for explanation). Two signal types were considered: (a) **One signal with one motif.** The AA motif `LDR` was implanted in a certain fraction of sequences. The pattern is randomly altered at one of the three positions with probabilities 0.2, 0.6, and 0.2, respectively. (b) **One signal with multiple motifs.** One of the three possible motifs `LDR`, `CAS`, and `GL-N` was

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

	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 A1: Properties of simulated repertoires, variations of motifs, and motif frequencies, i.e. the witness rate, for the datasets in categories "simulated immunosequencing data", "LSTM-generated data", and "real-world data with implanted signals". Noise types for $*$ are explained in paragraph "real-world data with implanted signals".

implanted with equal probability. Again, the motifs were randomly altered before implantation. The AA motif `LDR` changed as described above. The AA motif `CAS` was altered at the second position with probability 0.6 and with probability 0.3 at the first position. The pattern `GL-N`, where `-` denotes a gap location, is randomly altered at the first position with probability 0.6 and the gap has a length of 0, 1, or 2 AAs with equal probability.

Additionally, the datasets differ in the values for ρ, the average ratio of sequences carrying a signal, which were chosen as 1% or 0.1%. The motifs were implanted at positions 107, 109, and 114 according to the IMGT numbering scheme for immune receptor sequences (Lefranc et al., 2003) with probabilities 0.3, 0.35 and 0.2, respectively. With the remaining 0.15 chance, the motif is implanted at any other sequence position. This means that the motif occurrence in the simulated sequences is biased towards the middle of the sequence.

A3.4 Real-world data: CMV dataset

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

A3.5 Comparison to other MIL datasets

We give a non-exhaustive overview of previously considered MIL datasets and problems in Table A2. To our knowledge the datasets considered in this work pose the most challenging MIL problems with respect to the number of instances per bag (column 5).

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

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

Table A2: MIL datasets with their numbers of bags and numbers of instances. “total number of instances” refers to the total number of instances in the dataset. The simulated and real-world immunosequencing datasets considered in this work contain a by orders of magnitudes larger number of instances per bag than MIL datasets that were considered by machine learning methods up to now.

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

A4 Compared methods

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

A4.1 Known motif

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

A4.2 Support Vector Machine (SVM)

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

$$\begin{aligned} k(X^{(n)}, X^{(l)}) &= k_{\text{MinMax}}(\boldsymbol{u}^{(n)}, \boldsymbol{u}^{(l)}) \\ &= \frac{\sum_{m=1}^{d_u} \min(u_m^{(n)}, u_m^{(l)})}{\sum_{m=1}^{d_u} \max(u_m^{(n)}, u_m^{(l)})}, \end{aligned} \tag{A1}$$

where $u_m^{(n)}$ is the m-th element of the vector $u^{(n)}$. The *Jaccard kernel* (Levandowsky & Winter, 1971) is identical to the MinMax kernel except that it operates on binary $\boldsymbol{u}^{(n)}$. We used a standard C-SVM, as introduced by Cortes & Vapnik (1995). The corresponding hyperparameter C is optimized by random search. The settings of the full hyperparameter search as well as the respective value ranges are given in Table A4a.

A4.3 K-Nearest Neighbor (KNN)

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

$$\begin{aligned} d_{\text{MinMax}}(\boldsymbol{u}^{(n)}, \boldsymbol{u}^{(l)}) &= 1 - k_{\text{MinMax}}(\boldsymbol{u}^{(n)}, \boldsymbol{u}^{(l)}), \\ d_{\text{Jaccard}}(\boldsymbol{u}^{(n)}, \boldsymbol{u}^{(l)}) &= 1 - k_{\text{Jaccard}}(\boldsymbol{u}^{(n)}, \boldsymbol{u}^{(l)}). \end{aligned} \tag{A2}$$

The amount of neighbors is treated as the hyperparameter and optimized by an exhaustive grid search. The settings of the full hyperparameter search as well as the respective value ranges are given in Table A5.

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

A4.4 Logistic regression

We implemented logistic regression on the *k-mer representation* $\boldsymbol{u}$ of an immune repertoire. The model is trained by gradient descent using the Adam optimizer (Kingma & Ba, 2014). The learning rate is treated as the hyperparameter and optimized by grid search. Furthermore, we explored two regularization settings using combinations of $l1$ and $l2$ weight decay. The settings of the full hyperparameter search as well as the respective value ranges are given in Table A6.

A4.5 Burden test

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

A4.6 Logistic MIL (Ostmeyer et al)

The logistic multiple instance learning (MIL) approach for immune repertoire classification (Ostmeyer et al., 2019) applies a logistic regression model to each k-mer representation in a bag. The resulting scores are then summarized by max-pooling to obtain a prediction for the bag. Each amino acid of each k-mer is represented by 5 features, the so-called Atchley factors (Atchley et al., 2005). As k-mers of length 4 are used, this gives a total of $4 \times 5 = 20$ features. One additional feature per 4-mer is added, which represents the relative frequency of this 4-mer with respect to its containing bag, resulting in 21 features per 4-mer. Two options for the relative frequency feature exist, which are (a) whether the frequency of the 4-mer ("4MER") or (b) the frequency of the sequence in which the 4-mer appeared ("TCRβ") is used. We optimized the learning rate, batch size, and early stopping parameter on the validation set. The settings of the full hyperparameter search as well as the respective value ranges are given in Table A8.

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

A5 Hyperparameter selection

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

DeepRC. The set of hyperparameters of DeepRC is shown in Table A3. These hyperparameter combinations are adjusted via a grid search procedure.

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

Table A3: **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 $\geq 1\%$ in the category "simulated immunosequencing data".

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

SVM. The corresponding hyperparameter C of the SVM is optimized by randomly drawing 10^3 values in the range of $[-6; 6]$ according to a uniform distribution. These values act as the exponents of a power of 10 and are applied for each of the two kernel types (see Table A4a).

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

(a)

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

KNN. The amount of neighbors is treated as the hyperparameter and optimized by grid search operating in the discrete range of $[1; \max\{N, 10^3\}]$ with a step size of 1. The corresponding tight upper bound is automatically defined by the total amount of samples $N \in \mathbb{N}_{>0}$ in the training set, capped at 10^3 (see Table A5).

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

Table A5: 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. The hyperparameter optimization strategy that was used was grid search across hyperparameters given in Table A6.

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

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 A6: Settings used in the hyperparameter search of the logistic regression baseline approach.

Burden test. The burden test selects two hyperparameters: the number of features in the burden set and the type of features, see Table A7.

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

Table A7: Settings used in the hyperparameter search of the burden test 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 A8).

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

Table A8: 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.

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

A6 Results

In this section, we report the detailed results on all four categories of datasets (a) simulated immunosequencing data (Table A9) (b) LSTM-generated data (Table A10), (c) real-world data with implanted signals (Table A11), and (d) real-world data on the *CMV dataset* (Table A12), as discussed in the main paper.

ID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	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 (TCRβ)	0.544	0.505	0.493	0.487	0.476	0.500	0.520	0.495	0.510	0.492	0.506	0.503	0.509	0.505	0.500	0.475	0.489	0.500	0.501
	± 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
Burden test	0.770	0.523	0.510	0.666	0.510	0.509	0.652	0.508	0.505	0.583	0.508	0.509	0.564	0.508	0.507	0.536	0.508	0.504	0.549
	± 0.013	± 0.013	± 0.014	± 0.011	± 0.009	± 0.007	± 0.008	± 0.011	± 0.012	± 0.012	± 0.007	± 0.014	± 0.017	± 0.010	± 0.020	± 0.012	± 0.016	± 0.016	± 0.074
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 A9: 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. Wildcard characters in motifs are indicated by Z, characters with 50% probability of being removed by ^{d}.

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

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 (TCRβ)	0.541 ± 0.086	0.566 ± 0.162	0.468 ± 0.086	0.505 ± 0.067	0.500 ± 0.121	0.516 ± 0.038
Burden test	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	0.999 ± 0.003	0.792 ± 0.280	0.958 ± 0.093
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 A10: 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 A3, paragraph "LSTM-generated data", are indicated by ^{r}.

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

	OM 1%	OM 0.1%	MM 1%	MM 0.1%	Avg.
DeepRC	**1.000** $\pm$ 0.000	**0.984** $\pm$ 0.008	0.999 $\pm$ 0.001	**0.938** $\pm$ 0.009	**0.980** $\pm$ 0.029
SVM (MinMax)	**1.000** $\pm$ 0.000	0.578 $\pm$ 0.020	**1.000** $\pm$ 0.000	0.531 $\pm$ 0.019	0.777 $\pm$ 0.258
SVM (Jaccard)	0.988 $\pm$ 0.003	0.527 $\pm$ 0.016	**1.000** $\pm$ 0.000	0.574 $\pm$ 0.019	0.772 $\pm$ 0.257
KNN (MinMax)	0.744 $\pm$ 0.237	0.486 $\pm$ 0.031	0.674 $\pm$ 0.182	0.500 $\pm$ 0.022	0.601 $\pm$ 0.128
KNN (Jaccard)	0.652 $\pm$ 0.155	0.484 $\pm$ 0.025	0.695 $\pm$ 0.200	0.508 $\pm$ 0.025	0.585 $\pm$ 0.104
Logistic regression	**1.000** $\pm$ 0.000	0.544 $\pm$ 0.035	0.991 $\pm$ 0.003	0.512 $\pm$ 0.035	0.762 $\pm$ 0.270
Logistic MIL (KMER)	0.541 $\pm$ 0.074	0.506 $\pm$ 0.034	0.994 $\pm$ 0.004	0.620 $\pm$ 0.153	0.665 $\pm$ 0.224
Logistic MIL (TCRβ)	0.503 $\pm$ 0.032	0.501 $\pm$ 0.016	0.992 $\pm$ 0.003	0.782 $\pm$ 0.030	0.695 $\pm$ 0.238
Burden test	**1.000** $\pm$ 0.000	0.640 $\pm$ 0.048	**1.000** $\pm$ 0.000	0.891 $\pm$ 0.016	0.883 $\pm$ 0.170
Known motif b.	1.000 $\pm$ 0.000	0.704 $\pm$ 0.028	0.994 $\pm$ 0.003	0.620 $\pm$ 0.038	0.830 $\pm$ 0.196
Known motif c.	0.920 $\pm$ 0.004	0.562 $\pm$ 0.028	0.647 $\pm$ 0.030	0.515 $\pm$ 0.031	0.661 $\pm$ 0.181

Table A11: 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. **OM 1%:** In this dataset, a single motif with a frequency of 1% was implanted. **OM 0.1%:** In this dataset, a single motif with a frequency of 0.1% was implanted. **MM 1%:** In this dataset, multiple motifs with a frequency of 1% were implanted. **MM 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 A3, paragraph "Real-world data with implanted signals.".

	AUC	F1 score	Balanced accuracy	Accuracy
DeepRC	**0.831** $\pm$ 0.002	**0.726** $\pm$ 0.050	**0.741** $\pm$ 0.043	0.727 $\pm$ 0.049
SVM (MinMax)	0.825 $\pm$ 0.022	0.680 $\pm$ 0.056	0.734 $\pm$ 0.037	**0.742** $\pm$ 0.031
SVM (Jaccard)	0.546 $\pm$ 0.021	0.272 $\pm$ 0.184	0.523 $\pm$ 0.026	0.542 $\pm$ 0.032
KNN (MinMax)	0.679 $\pm$ 0.076	0.000 $\pm$ 0.000	0.500 $\pm$ 0.000	0.545 $\pm$ 0.044
KNN (Jaccard)	0.534 $\pm$ 0.039	0.073 $\pm$ 0.101	0.508 $\pm$ 0.012	0.551 $\pm$ 0.042
Logistic regression	0.607 $\pm$ 0.058	0.244 $\pm$ 0.206	0.552 $\pm$ 0.049	0.590 $\pm$ 0.019
Logistic MIL (KMER)	0.582 $\pm$ 0.065	0.118 $\pm$ 0.264	0.503 $\pm$ 0.007	0.515 $\pm$ 0.058
Logistic MIL (TCRβ)	0.515 $\pm$ 0.073	0.000 $\pm$ 0.000	0.496 $\pm$ 0.008	0.541 $\pm$ 0.039
Burden test	0.699 $\pm$ 0.041	-	-	-

Table A12: 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.

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

A7 Repertoire generation via LSTM

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

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

Finally, we randomly assigned 500 of the generated repertoires to the positive (diseased) and 500 to the negative (healthy) class. We then implanted motifs in the positive class repertoires as described in Section A3.2.

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

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

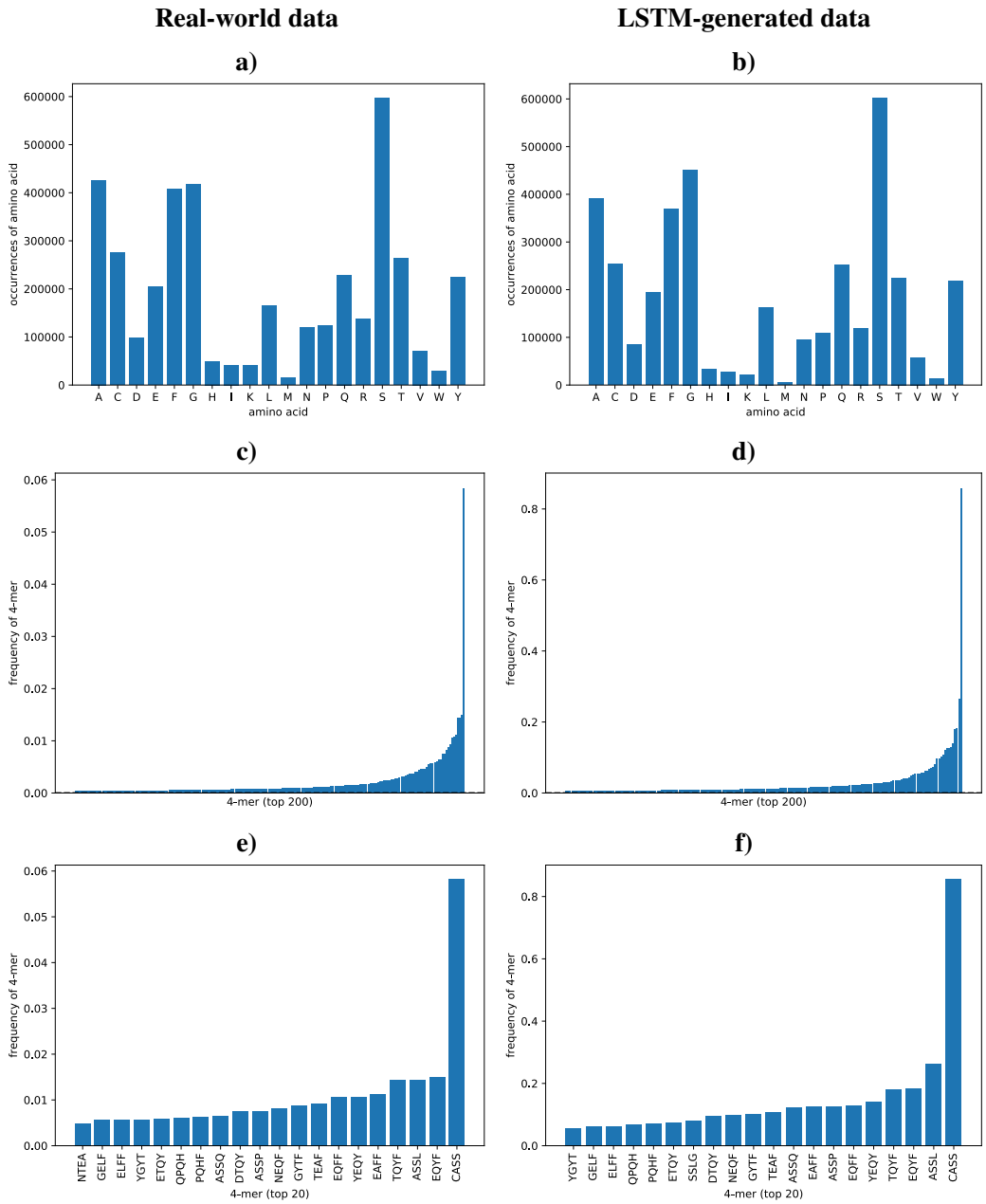

Figure A2: Distribution of AAs and k-mers in real-world *CMV dataset* and LSTM-generated data. **Left:** Histograms of real-world data. **Right:** Histograms of LSTM-generated data. **a)** Frequency of AAs in sequences of the *CMV dataset*. **b)** Frequency of AAs in sequences of the LSTM-generated datasets.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

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

A8 Interpreting DeepRC

DeepRC allows for two forms of interpretability methods. (a) Due to its attention-based design, a trained model can be used to compute the attention weights of a sequence, which directly indicates its importance. (b) DeepRC furthermore allows for the usage of contribution analysis methods, such as Integrated Gradients (IG) (Sundararajan et al., 2017) or Layer-Wise Relevance Propagation (Montavon et al., 2018; Arras et al., 2019; Montavon et al., 2019; Preuer et al., 2019). We apply IG to identify the input patterns that are relevant for the classification. To identify AA patterns with high contributions in the input sequences, we apply IG to the AAs in the input sequences. Additionally, we apply IG to the kernels of the 1D CNN, which allows us to identify AA motifs with high contributions. In detail, we compute the IG contributions for the AAs and positional features in the kernels for every repertoire in the validation and test set, so as to exclude potential artifacts caused by over-fitting. Averaging the IG values over these repertoires then results in concise AA motifs. We include qualitative visual analyses of the IG method on different datasets below.

Here, we provide examples for the interpretation of trained DeepRC models using Integrated Gradients (IG) (Sundararajan et al., 2017) as contribution analysis method. The following illustrations were created using 50 IG steps, which we found sufficient to achieve stable IG results.

A visual analysis of DeepRC models on the simulated datasets, as illustrated in Tab. A13 and Fig. A3, shows that the implanted motifs can be successfully extracted from the trained model and are straightforward to interpret. In the real-world *CMV dataset*, DeepRC finds complex patterns with high variability in the center regions of the immuno-sequences, as illustrated in figure A4.

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

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

Table A13: Visualization of motifs extracted from trained DeepRC models for datasets from categories "simulated immunosequencing data", "LSTM-generated data", and "real-world data with implanted signals". Motif extraction was performed using Integrated Gradients on the 1D CNN kernels over the validation set and test set repertoires of one CV fold. Wildcard characters are indicated by Z, random noise on characters by ^{r}, characters with 50% probability of being removed by ^{d}, and gap locations of random lengths of $\{0; 1; 2\}$ by -. Larger characters in the extracted motifs indicate higher contribution, with blue indicating positive contribution and red indicating negative contribution towards the prediction of the diseased class. Contributions to positional encoding are indicated by $<$ (beginning of sequence), $\wedge$ (center of sequence), and $>$ (end of sequence). Only kernels with relatively high contributions are shown, i.e. with contributions roughly greater than the average contribution of all kernels.

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

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 $\mathtt{SZ}^{\mathtt{d}}\mathtt{Z}^{\mathtt{d}}\mathtt{N}$ 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 $\{\mathtt{L}^{\mathtt{r}}\mathtt{D}^{\mathtt{r}}\mathtt{R}^{\mathtt{r}}; \mathtt{C}^{\mathtt{r}}\mathtt{A}^{\mathtt{r}}\mathtt{S}; \mathtt{G}^{\mathtt{r}}\mathtt{L}\text{-}\mathtt{N}\}$ 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. Wildcard characters in implanted motifs are indicated by Z, characters with 50% probability of being removed by $^{\mathtt{d}}$, 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.

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

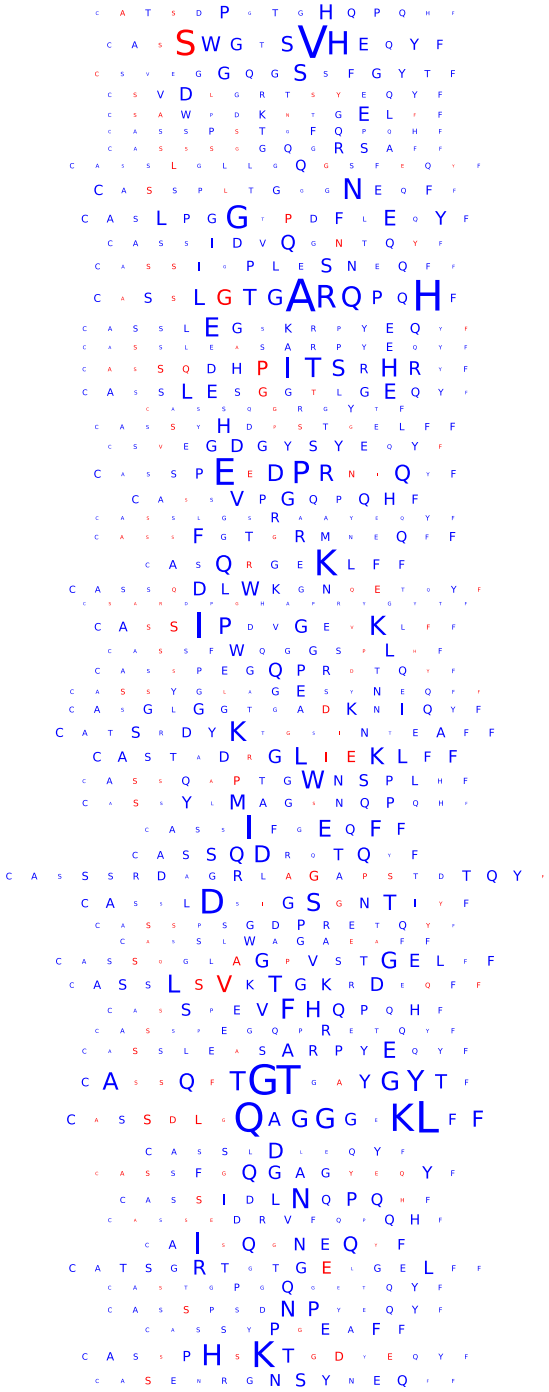

Figure A4: Visualization of the contributions of characters within a sequence via IG. Each sequence was selected from a different repertoire and showed the highest contribution in its repertoire. The model was trained on *CMV dataset*, using a kernel size of 9, 32 kernels and 137 repertoires for early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

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

A9 Attention values for previously associated CMV sequences

Table A14 lists sequences of the *CMV dataset* that were previously associated with CMV immune status and their assigned high attention values by DeepRC.

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 A14: 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.

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

A10 DeepRC variations and ablation study

In this section we investigate the impact of different variations of DeepRC on the performance on the *CMV dataset*. We consider both a CNN-based sequence embedding, as used in the main paper, and an LSTM-based sequence embedding. In both cases we vary the number of attention heads and the β parameter for the softmax function the attention mechanism (see Eq. 2 in main paper). For the CNN-based sequence embedding we also vary the number of CNN kernels and the kernel sizes used in the 1D CNN. For the LSTM-based sequence embedding we use one one-directional LSTM layer, of which the output values at the last sequence position (without padding) are taken as embedding of the sequence. Here we vary the number of LSTM blocks in the LSTM layer. To counter over-fitting due to the increased complexity of these DeepRC variations, we added a $l2$ weight penalty to the training loss. The factor with which the $l2$ weight penalty contributes to the training loss is varied over 3 orders of magnitudes, where suitable value ranges were manually determined on one of the training folds beforehand.

To reduce the computational effort, we do not consider all numbers of kernels that were considered in the main paper. Furthermore, we only compute the AUC scores on 3 of the 5 cross-validation folds. The hyperparameters, which were used in a grid search procedure, are listed in Tab. A15 for the CNN-based sequence embedding and Tab. A16 for the LSTM-based sequence embedding.

Results. We show performance in terms of AUC score with single hyperparameters set to fixed values so as to investigate their influence in Tab. A18 for the CNN-based sequence embedding and Tab. A17 for the LSTM-based sequence embedding. We note that due to restricted computational resources this study was conducted with fewer different numbers of CNN kernels, with the AUC estimated from only 3 of the 5 cross-validation folds, which leads to a slight decrease of performance in comparison to the full hyperparameter search and cross-validation procedure used in the main paper. As can be seen in Tab. A18 and A17, the LSTM-based sequence embedding generalizes slightly better than the CNN-based sequence embedding. The performance of DeepRC, however, remains rather robust w.r.t. the different hyperparameter settings.

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

Table A15: Hyperparameter search space for DeepRC variations with CNN-based sequence embedding. Every $5 \cdot 10^3$ updates, the current model was evaluated against the validation fold. The early stopping hyperparameter was determined by selecting the model with the best loss on the validation fold after $2 \cdot 10^5$ updates.

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

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

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

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

Table A17: Impact of hyperparameters on DeepRC with LSTM for sequence encoding. Mean ("mean") and standard deviation ("std") for the area under the ROC curve over the first 3 folds of a 5-fold nested cross-validation for different sub-sets of hyperparameters ("sub-set") are shown. The following sub-sets were considered: "full": Full grid search over hyperparameters; "beta=*": Grid search over hyperparameters with reduction to specific value $*$ of beta value of attention softmax; "heads=*": Grid search over hyperparameters with reduction to specific number $*$ of attention heads; "lstms=*": Grid search over hyperparameters with reduction to specific number $*$ of LSTM blocks for sequence embedding.

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

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

Table A18: Impact of hyperparameters on DeepRC with 1D CNN for sequence encoding. Mean ("mean") and standard deviation ("std") for the area under the ROC curve over the first 3 folds of a 5-fold nested cross-validation for different sub-sets of hyperparameters ("sub-set") are shown. The following sub-sets were considered: "full": Full grid search over hyperparameters; "beta=*": Grid search over hyperparameters with reduction to specific value $*$ of beta value of attention softmax; "heads=*": Grid search over hyperparameters with reduction to specific number $*$ of attention heads; "ksize=*": Grid search over hyperparameters with reduction to specific kernel size $*$ of 1D CNN kernels for sequence embedding; "kernels=*": Grid search over hyperparameters with reduction to specific number $*$ of 1D CNN kernels for sequence embedding.

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

Bibliography

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

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.

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

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.

Carbonneau, M.-A., Cheplygina, V., Granger, E., and Gagnon, G. Multiple instance learning: a survey of problem characteristics and applications. *Pattern Recognition*, 77:329–353, 2018.

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.

Demircigil, M., Heusel, J., Löwe, M., Upgang, S., and Vermet, F. On a model of associative memory with huge storage capacity. *Journal of Statistical Physics*, 168(2):288–299, 2017.

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

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.

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*, 2019.

Foulds, J. and Frank, E. A review of multi-instance learning assumptions. *The Knowledge Engineering Review*, 25(1):1–25, 2010.

Galson, J. D., Schaetzle, S., Bashford-Rogers, R. J. M., Raybould, M. I. J., Kovaltsuk, A., Kilpatrick, G. J., Minter, R., Finch, D. K., Dias, J., James, L., Thomas, G., Lee, W.-Y. J., Betley, J., Cavlan, O., Leech, A., Deane, C. M., Seoane, J., Caldas, C., Pennington, D., Pfeffer, P., and Osbourn, J. Deep sequencing of B cell receptor repertoires from COVID-19 patients reveals strong convergent immune signatures. *bioRxiv*, 2020.

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

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*, 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*, 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*, 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.

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

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*, 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*, 1911.06616, 2019.

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

Krotov, D. and Hopfield, J. J. Dense associative memory for pattern recognition. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, pp. 1172–1180. Curran Associates, Inc., 2016.

Krotov, D. and Hopfield, J. J. Dense associative memory is robust to adversarial inputs. *Neural Computation*, 30(12):3151–3167, 2018.

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.

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

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

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

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

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

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.

Minervina, A. A., Komech, E. A., Titov, A., Koraichi, M. B., Rosati, E., Mamedov, I. Z., Franke, A., Efimov, G. A., Chudakov, D. M., Mora, T., Walczak, A. M., Lebedev, Y. B., and Pogorelyy, M. V. Longitudinal high-throughput TCR repertoire profiling reveals the dynamics of T cell memory formation after mild COVID-19 infection. *bioRxiv*, 2020.

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.

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.

Pawlowski, N., Bhooshan, S., Ballas, N., Ciompi, F., Glocker, B., and Drozdzal, M. Needles in haystacks: on classifying tiny objects in large images. *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., Widrich, M., Gruber, L., Holzleitner, M., Pavlović, M., Sandve, G. K., Greiff, V., Kreil, D., Kopp, M., Klambauer, G., Brandstetter, J., and Hochreiter, S. Hopfield networks is all you need. *ArXiv*, 2020.

Raybould, M. I. J., Kovaltsuk, A., Marks, C., and Deane, C. M. CoV-AbDab: the coronavirus antibody database. *bioRxiv*, 2020.

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

Rosenfeld, A. M., Meng, W., Luning Prak, E. T., and Hershberg, U. Immunedb, a novel tool for the analysis, storage, and dissemination of immune repertoire sequencing data. *Frontiers in Immunology*, 9:2107, 2018.

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

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

Setliff, I., Shiakolas, A. R., Pilewski, K. A., Murji, A. A., Mapengo, R. E., Janowska, K., Richardson, S., Oosthuysen, C., Raju, N., Ronsard, L., et al. High-throughput mapping of B cell receptor sequences to antigen specificity. *Cell*, 179(7):1636–1646, 2019.

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.

Shugay, M., Bagaev, D. V., Zvyagin, I. V., Vroomans, R. M., Crawford, J. C., Dolton, G., Komech, E. A., Sycheva, A. L., Koneva, A. E., Egorov, E. S., et al. VDJdb: a curated database of T-cell receptor sequences with known antigen specificity. *Nucleic Acids Research*, 46(D1):D419–D427, 2018.

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.

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.

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

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), 2019.

Uriot, T. Learning with sets in multiple instance regression applied to remote sensing. *ArXiv*, 1903.07745, 2019.

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, 2017.

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.

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

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

bioRxiv preprint doi: https://doi.org/10.1101/2020.04.12.038158; this version posted August 17, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. All rights reserved. No reuse allowed without permission.

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*, 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.

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

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