DeepTCR: a deep learning framework for revealing structural concepts within TCR Repertoire

John-William Sidhom[1,2,3], H. Benjamin Larman[1,2,4],

Drew M. Pardoll[1,2], Alexander S. Baras[1,2,4]

[1] Bloomberg~Kimmel Institute for Cancer Immunotherapy, Sidney Kimmel Comprehensive Cancer Center, Johns Hopkins University School of Medicine, Baltimore, MD, USA

[2] The Sidney Kimmel Comprehensive Cancer Center, Johns Hopkins University School of Medicine, Baltimore, MD, USA

[3] Department of Biomedical Engineering, Johns Hopkins University School of Medicine, Baltimore, MD, USA

[4] Department of Pathology, Johns Hopkins University School of Medicine, Baltimore, MD, USA

Deep learning algorithms have been utilized to achieve excellent performance in pattern-recognition tasks, such as in image and vocal recognition[1,2]. The ability to learn complex patterns in data has tremendous implications in the genomics world, where sequence motifs become learned 'features' that can be used to predict functionality, guiding our understanding of disease and basic biology[3–6]. T-cell receptor (TCR) sequencing assesses the diversity of the adaptive immune system, and while prior conventional biological sequence analysis tools have been insightful, they can miss signals in the data due to their rigidity[7–9]. We present DeepTCR, a broad collection of unsupervised and supervised deep learning methods able to uncover structure in highly complex and large TCR sequencing data. We demonstrate its utility across multiple basic science and clinical examples, including learning antigen-specific motifs and understanding immunotherapy-related shaping of repertoire. We further extract meaningful motifs from the trained network as a means of explaining the sequence concepts that have been learned to accomplish a given task. Our results show the flexibility and capacity for deep neural networks to handle the complexity of high-dimensional genomics data for both descriptive and predictive purposes.

Next-Generation Sequencing (NGS) has allowed a comprehensive description and understanding of the complexity encoded at the genomic level in a wide variety of organisms. The applications of NGS have grown rapidly as this technology has become a molecular microscope for understanding the genomic basis for the fundamental functions of the cell[10]. In parallel to this explosion of NGS applications, in the machine learning world, deep learning has seen a similar expansion of applications as computational resources have grown, large advances in algorithms and programming libraries have distributed these capabilities to many scientific communities, and in particular, big data has transcended all facets of daily life. As a result of these two technological revolutions, there exists many opportunities to apply deep learning in genomics as the data generated from NGS is very large and highly complex.

T-cell receptor sequencing (TCRSeq) is an application of NGS that has allowed scientists across many disciplines to characterize the diversity of the immune system[11–13] (Supplementary Fig. 1). By selectively amplifying and sequencing the highly diverse CDR3 region of the β-chain of T-cells, scientists have been able to study the diverse repertoire the immune system generates to probe both foreign and native potential antigens. With this new sequencing technology, there has arisen a need to develop analytical tools to parse and draw meaningful concepts from the data. In recent work, investigators have applied conventional sequence analytics, where either targeted motif searches or sequence alignment algorithms have been applied to begin parsing the structural data within TCRSeq[7–9]. However, since many of these approaches were initially conceived to analyze longer biological strings for the purpose of identifying evolutionary changes at the DNA or protein level, problems can arise when applying them to TCRSeq data in which the strings being compared are quite short and the end regions are highly conserved. Finally, while these methods

are considered unsupervised machine learning approaches, there has been little in the way of using supervised approaches to guide the learning process.

We present DeepTCR, a package of both unsupervised and supervised deep learning methods for analysis of TCRSeq at both the sequence and sample level in order to learn concepts in the data that may be used for both descriptive and predictive purposes. In order to demonstrate the utility of these algorithms, we collected three previously published datasets including samples sorted by antigen-specificity (*Glanville_2017* & *Sidhom_2017),* and samples taken from cohorts of tumor-bearing mice treated with various immunotherapies *(Rudqvist_2017)*[7,8,14] (Supplementary. Fig. 2,3).

The first class of algorithms we developed are unsupervised deep learning methods that learn the underlying distribution of the sequence data in high-dimensional space for the purpose of 1) clustering TCR sequences that likely recognize the same antigen and 2) for the first time quantifying similarity between whole repertoires based on their structural composition. We implement both a variational autoencoder (VAE) and generative adversarial network (GAN) to perform dimensionality reductions and data re-representations at a sequence level, using convolutional layers in order to learn motifs that describe the distribution of data. We first implemented the VAE as autoencoders have been previously described as a common dimensionality reduction/data re-representation technique[15,16]. When implemented with trainable convolutional layers, they can become powerful as a data re-representation technique for images, allowing downstream analysis such as clustering of similar images. Our implementation of a variational autoencoder starts by taking a TCR sequence that is embedded in a fixed-length vector with zero right padding (Fig. 1A). We then use a trainable embedding layer, as described in *Sidhom et. al*, to learn meaning of the amino acids, moving them from a discrete to continuous numerical

space[6]. This is followed by convolutional layers, ultimately reducing the sequence to a latent space that is described as a multi-dimensional unit gaussian distribution. The sequence is then reconstructed from the latent space through the use of deconvolutional layers and the transposition of the trainable embedding layer that was used at the beginning of the network. The network is then optimized with a gradient-descent based algorithm minimizing a reconstruction loss and variational loss, which acts as a mode of regularization. Since this algorithm is primarily trained to minimize the reconstruction loss, the concept of sequence length is learned within the network. However, since TCR sequences are variable length sequences that describe a structural part of the TCR, they can contain length-independent motifs that are required for antigen-specificity, as has been previously demonstrated by *Glanville et. al*[7].

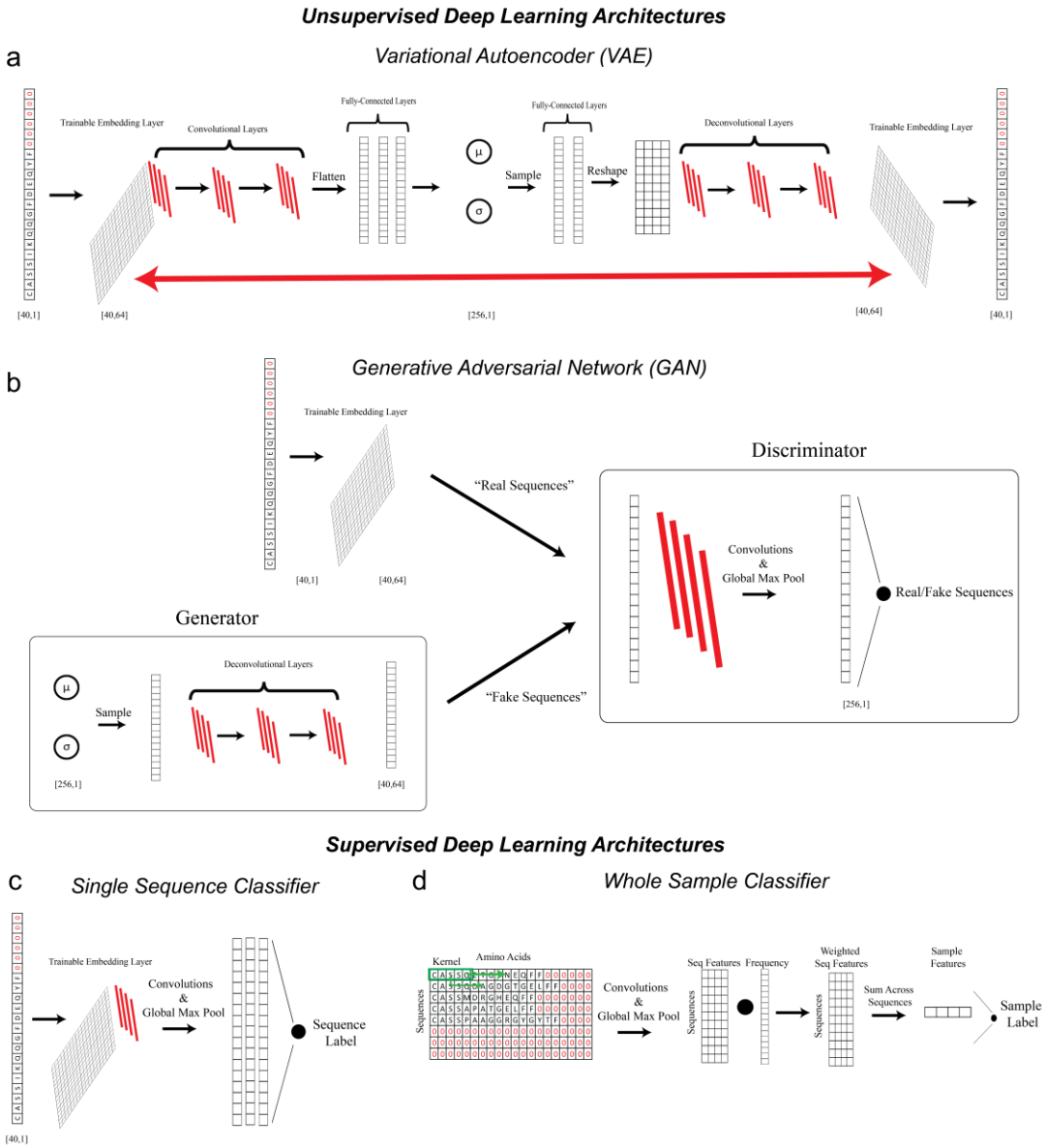


Figure 1. Deep Learning Architectures. (a) The variational autoencoder (VAE) is designed to take as a discrete input the amino acid sequence of the TCR sequence with a right zero-padding scheme. A trainable embedding layer is used to transform the sequence from discrete to continuous numerical domain. Convolutional and fully connected layers transform the sequence into a latent representation that is parametrized by a multi-dimensional unit gaussian. Reconstruction of the sequence occurs via fully connected and deconvolutional layers followed by the transposition of the same trainable embedding layer used at the beginning of the network. **(b)** The generative adversarial network (GAN) consists of the generator and discriminator, separate networks trained with separate objective functions. The generator samples from a multi-dimensional unit gaussian to create a 'fake' TCR sequence. The discriminator learns to distinguish 'real' from 'fake' sequences through one layer of convolutions with a global max pooling operation to provide translational invariance to the network. Of note, the generator's output is the continuous and not discrete representation of the TCR sequence. The latent space used for downstream analysis is the penultimate layer of the discriminator, here described as having dimensions of [256,1]. **(c)** The single sequence classifier follows a conventional convolutional neural network architecture consisting of one convolutional layer with global max pooling and three fully connected layers to a final classification layer. **(d)** The whole sample classifier utilizes a kernel that scans in a horizontal fashion across all sequences in the file resulting in a sequences-by-features tensor. This is then multiplied by the frequency vector for each sequence to derive weighted sequence features. These are then summed across the sequence space to compute sample level features that are fed into a classification layer.

In order to implement an unsupervised deep learning method that could learn features in a length independent fashion, we utilized a GAN architecture (Fig. 1B)[17]. This model consists of two networks, the generator and discriminator that train in an adversarial manner, optimizing separate objective functions (Supplementary Fig. 4). The generator attempts to model the distribution of sequencing data through a generative process where a latent vector is randomly sampled from a multi-dimensional unit gaussian and deconvolutional layers are used to create a TCR sequence. The discriminator is a network that is trained to distinguish between sequences from the 'real' data and sequences from the 'fake' generated data. Aside from being used to model biological sequence data, our implementation of the GAN differs from previously described architectures as it uses a discriminator that has only one convolutional layer with a global max pooling operation to achieve translational invariance to relevant motifs as described by *Sidhom et. al* [6]. In this manner, this network is designed to model the underlying sequence distribution in a length independent manner.

In order to assess how well these unsupervised methods could learn relevant features of TCR sequences, we used a previously published dataset of 2067 sequences for 7 specificities used to train GLIPH, a state-of-the-art method for clustering TCR sequences[7]. We note that both unsupervised deep learning methods are able to cluster sequences of the same specificity (Fig. 2A) and at the whole sample level can make meaningful comparisons between antigen-specific repertoires (Fig 2B). When assessing the specificity of these methods to cluster sequences in groups specific to a given antigen, the VAE demonstrates comparable performance to GLIPH while only requiring the CDR3 β-chain sequence (Fig 2C), demonstrating that 94.48% of clustered TCRs (14% of all sequences clustered) were correctly grouped with other sequences of common specificity. Furthermore, both the VAE and GAN maintain a high clustering accuracy while clustering more sequences. Finally, to assess the characteristics of the clusters formed at various

thresholds, we examined the number of clusters and the variance of the lengths of sequences the clusters contained (Fig 2D). While the VAE and GAN comparably cluster sequences of common specificity and create the same number of clusters doing so, the GAN clusters sequences of different lengths far more than the VAE.

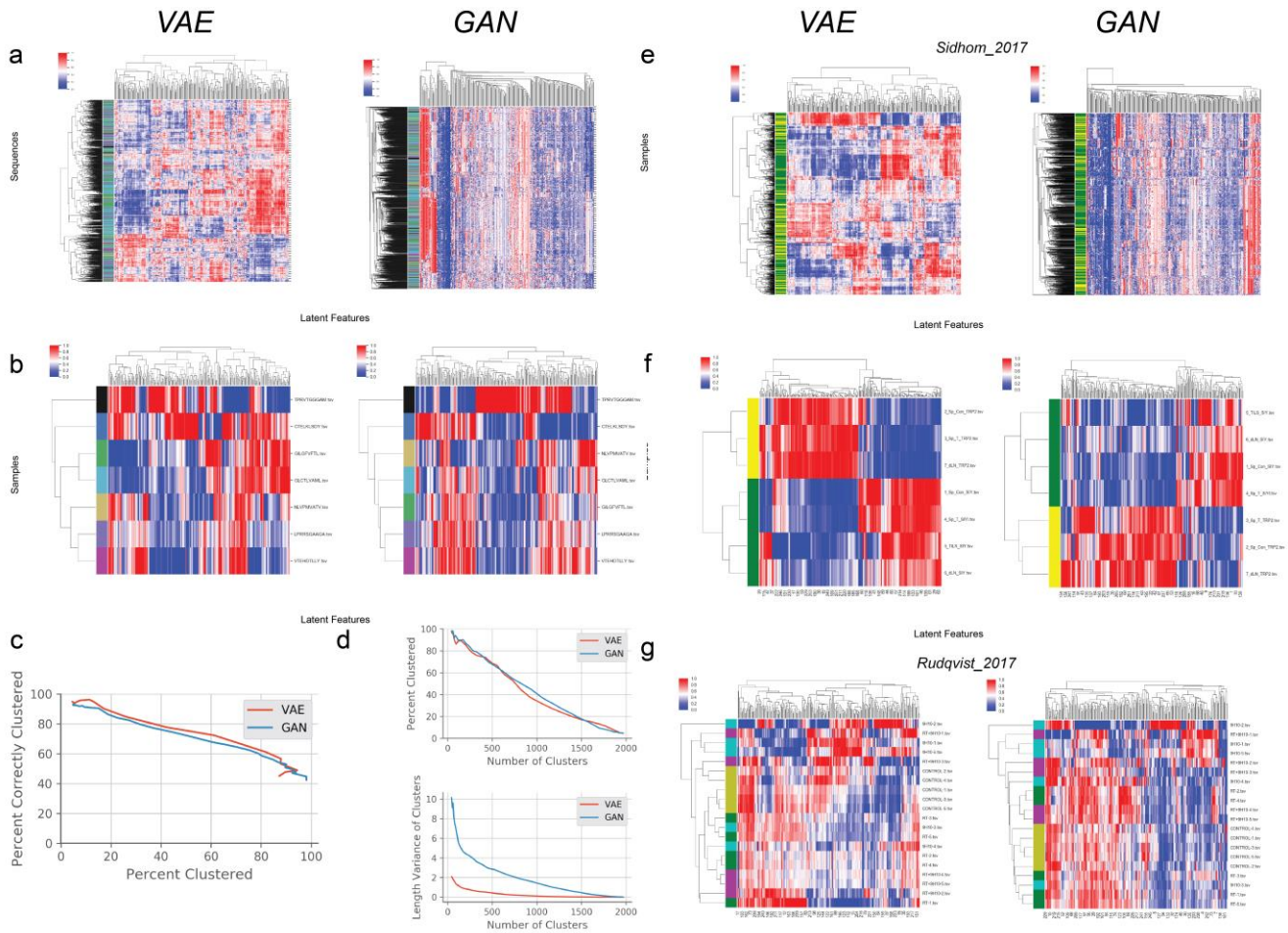


Figure 2. Unsupervised Learning Examples. (a) Heatmaps of sequence-by-features for 7 tetramer sorted populations of antigen-specific T-cells taken from *Glanville_2017* dataset. **(b)** Heatmaps of samples-by-features for 7 tetramer sorted populations of antigen-specific T-cells taken from *Glanville_2017* dataset. **(c)** Clustering specificity of VAE and GAN across various clustering thresholds following hierarchical clustering of sequences by their learned features. **(d)** Number of clusters vs percent of sequences clustered at various clustering thresholds for both VAE and GAN *(top).* Number of clusters vs length variance of sequences within clusters for both VAE and GAN *(bottom)* **(e)** Heatmaps of sequences-by-features for antigen-specific sequences taken from sorted SIY and TRP2 specific T-cells created by VAE and GAN. (SIY = green, TRP2 = yellow) **(f)** Heatmaps of samples-by-features for antigen-specific samples takes from sorted SIY and TRP2 specific T-cells created by VAE and GAN. (SIY = green, TRP2 = yellow) **(g)** Heatmaps of samples-by-features for the tumor-infiltrating lymphocyte (TIL) samples taken from various immunotherapies (Control = yellow, 9H10 = Cyan, RT = Green, RT + 9H10 = Magenta). Heatmaps for sequences-by-features in Supplementary Fig. 6.

When applying these two separate approaches on the *Sidhom_2017* dataset, we note the VAE and GAN are able to comparably cluster antigen-specific sequences as well as antigen-specific samples (Fig. 2 E & F). However, when clustering sequences from their latent representations, we noted that the variances in the length of sequences in a given cluster were much smaller from the VAE as opposed to the GAN (Supplementary Fig. 5) as we would expect, since the VAE learns length dependent features while the GAN does not. When applying these two types of unsupervised approaches to the *Rudqvist_2017* dataset, we note that both methods identify that the control mice have highly conserved structural profiles, as was described in the initial publication (Fig 2G). Our experience using these unsupervised approaches demonstrates they can be useful not only to cluster TCR sequences of high homology but also to compare repertoires at a wholistic level, allowing a method for the first time to quantify similarity between repertoires based on their overall structural composition.

As noted in these datasets, there are often labels associated to TCRSeq, which can either be applied at the sequence or sample level. To accommodate labels at the single sequence level, we designed a simple convolutional neural network that learns sequence specific motifs in a length independent fashion (Fig 1C) to correctly classify sequences by their labels. The second, and arguably the more interesting architecture, is a supervised multi-instance deep learning algorithm that is able to learn meaningful concepts that may lie within large samples of many sequences, either being obscured by the noise of many irrelevant sequences or are weakly predictive at the single sequence level (Fig 1D). This whole sample multi-instance classifier uses convolutional kernels that scan the entire file, learning features for each sequence. These features are then weighted by the frequency of the sequences. Finally, these features are summed across the sequences to give a weighted average of a feature/motif within a sample. We first applied the single

sequence classifier to the *Glanville_2017* dataset and noted there was a weak predictive signature that could differentiate the sequences with better performance for antigens with more TCR sequences (Fig 3A). However, when creating samples *in-silico* that used a given number of unique sequences per sample, we found an increase in predictive performance at the whole sample level as more sequences were used in each sample, demonstrating the ability of a 'weak learner' to become more predictive when provided with more evidence in aggregate (Fig 3B). When applied to the *Sidhom_2017* dataset, we note again that while the sequence level classifier is able to achieve reasonable performance, the whole sample classifier does far better as it is able to use an entire sample of sequences to make a prediction (Fig 3C). This point is further demonstrated in the *Rudqvist_2017* dataset as these samples are from tumor-infiltrating lymphocytes (TIL) where much of the signal comes from background repertoire, making it difficult for a sequence-level classifier to work. However, in the whole sample classifier, we see improved performance with particular improvement in the RT and Control groups as they have profound structural signatures (Fig 3D). Ultimately, the nature of how the T-cell receptor binds its cognate peptide-MHC makes prediction at the single sequence level difficult; however, when multiple instances of a concept are present, the whole sample classifier can leverage this data in order to make more accurate predictions. Finally, given that usually only the β-chain is sequenced, we acknowledge that this presents a considerable limitation in ultimately predicting antigen-specificity as both chains are important for recognition. Thus, we would expect improved performance across all described algorithms when provided both sequencing data for the α and β chains.

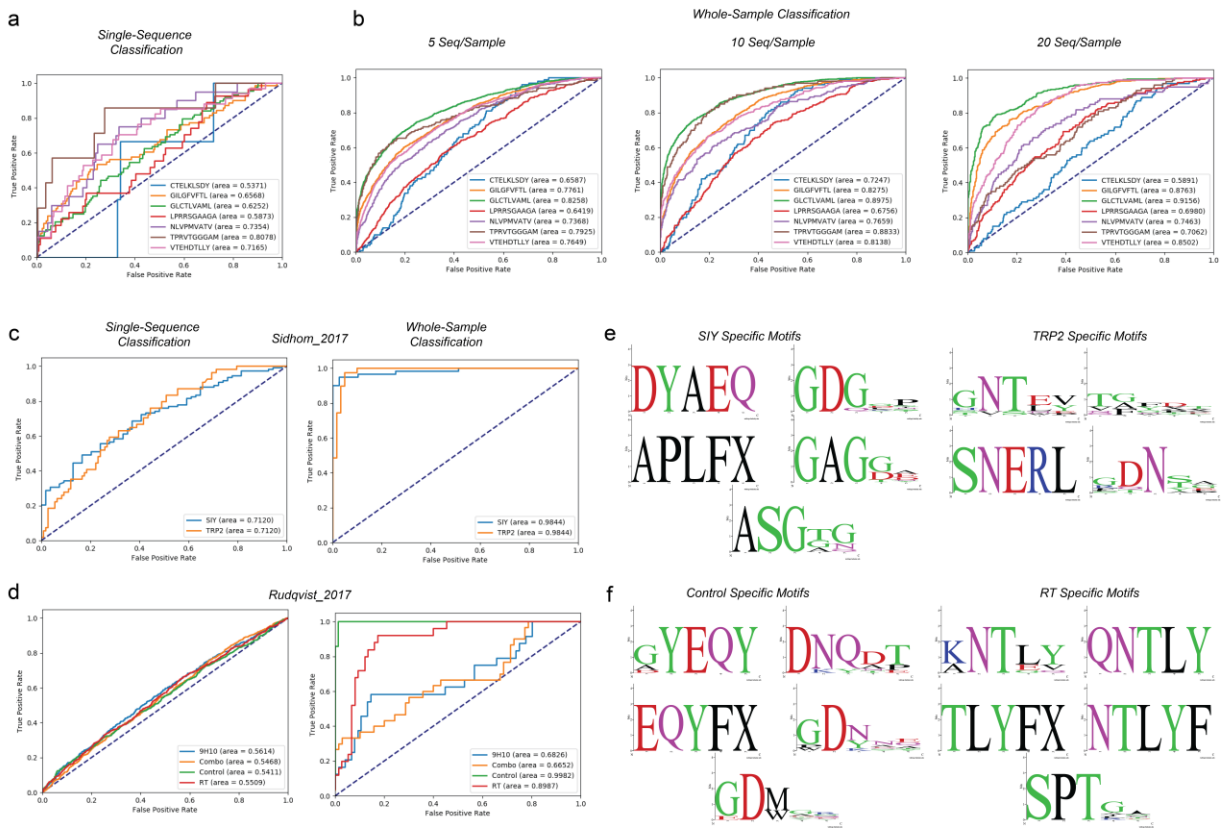


Figure 3. Supervised Learning Examples. (a) Receiver Operating Characteristic (ROC) curve for single-sequence classifier on sequences taken from 7 T-cell specificities from *Glanville_2017* dataset. **(b)** All unique sequence from *Glanville_2017* dataset were allocated randomly without replacement into *in-silico* samples so no sequences were shared among any samples. Samples were created with either 5,10, or 20 sequences/file and whole-sample classifier was used to assess predictive power at the whole sample level. **(c,d)** Receiver Operating Characteristic (ROC) curves for single sequence and whole sample classifier for *Sidhom_2017* and *Rudqvist_2017* datasets. **(e,f)** Representative motifs learned by whole sample classifier for cohorts that had highly predictive structural signatures.

While there is inherent value in predictive models as they can be used as biomarkers for disease, there has been much effort in improving the 'explainability' of deep learning models for the purpose of understanding what the network learned. In the context of TCRSeq, being able to extract knowledge from the network can inform relevant motifs for antigen-specific recognition. Therefore, we established a method by which we could query for differentially used motifs at the cohort level from the trained network allowing us to identify relevant cohort-specific motifs (Fig 3 E & F). By extracting the index along with the value of the feature following its convolution and

global max pooling operation, we can identify where in the sequence a kernel is being maximally activated and use this to derive the motif being learned. These supervised methods demonstrate how predictive models can also be used to generate descriptive results that can inform our understanding of the mechanisms at play.

NGS has become one of the largest sources of big data in the biological sciences, and deep learning is a promising modality for analyzing big data where features need to be learned. In this work, we present DeepTCR, a collection of unsupervised and supervised deep learning approaches to characterize TCRSeq data for both descriptive and predictive purposes. Our unsupervised approaches use newly developed techniques from the deep learning community including the variational autoencoder and generative adversarial network to relate and compare repertoires at the sequence and sample level. We further develop supervised methods in applications where labels can greatly help the learning process, such as when there is buried signal in a large sample of sequences. While DeepTCR has been developed for analyzing TCRSeq data, the concepts introduced within the designed architectures could translate and be applied to a variety of other NGS-based technologies. These types of technologies could yield an entire new area of biomarker discovery as well as improving our own understanding of the complex concepts within the genomic code.

1. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521,** 436 (2015).

2. Krizhevsky, A., Sutskever, I. & in neural, H. G. Imagenet classification with deep convolutional neural networks. (2012).

3. Alipanahi, B., Delong, A., Weirauch, M. T. & Frey, B. J. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nature Biotechnology* **33,** 831–838 (2015).

4. Zeng, H., Edwards, Liu, G. & Bioinformatics, G. D. Convolutional neural network architectures for predicting DNA–protein binding. (2016).

5. Han, Y. & bioinformatics, K. D. Deep convolutional neural networks for pan-specific peptide-MHC class I binding prediction. (2017).

6. Sidhom, J.-W., Pardoll, D. & Baras, A. AI-MHC: an allele-integrated deep learning framework for improving Class I & Class II HLA-binding predictions. *bioRxiv* 318881 (2018). doi:10.1101/318881

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

8. Sidhom, J.-W. *et al.* ImmunoMap: A Bioinformatics Tool for T-Cell Repertoire Analysis. *Cancer Immunology Research* **6,** canimm.0114.2017 (2017).

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

10. Buermans, H. P. J. & den Dunnen, J. T. Next generation sequencing technology: Advances and applications. *Biochimica et Biophysica Acta (BBA) - Molecular Basis of Disease* **1842,** 1932–1941 (2014).

11. ... *et al.* Tumor and microenvironment evolution during immunotherapy with nivolumab. (2017).

12. Gerlinger, M., Quezada, S. & of …, P. K. Ultra-deep T cell receptor sequencing reveals the complexity and intratumour heterogeneity of T cell clones in renal cell carcinomas. (2013).

13. Wang, G., Dash, P. & translational …, M. J. T cell receptor αβ diversity inversely correlates with pathogen-specific antibody levels in human cytomegalovirus infection. (2012).

14. Rudqvist, N., Pilones, K. & Immunol …, L. C. Radiotherapy and CTLA-4 blockade shape

the TCR repertoire of tumor-infiltrating T cells. (2018).

15. Pu, Y., Gan, Z., Henao, R., Yuan, X. & in neural …, L. C. Variational autoencoder for deep learning of images, labels and captions. (2016).

16. Doersch, C. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908* (2016).

17. Goodfellow, I., Pouget-Abadie, J. & in neural …, M. M. Generative adversarial nets. (2014).

Methods

Data Curation

TCR sequencing files were collected as raw tsv formatted files (Supplementary Fig. 1) from the various sources cited within the manuscript. Sequencing files were parsed to take the amino acid sequence of the CDR3 after removing unproductive sequences. Clones with different nucleotide sequences but the same amino acid sequence were aggregated together under one amino acid sequence and their reads were summed to determine their relative abundance. Within the parsing code, we additionally specified to ignore sequences that used non-IUPAC letters (*,X,O) and removed sequences that were greater than 40 amino acids in length. For the purpose of the algorithm, the maximum length can be altered but we chose 40 as we did not expect any real sequences to be longer than this length.

TCRSeq Quantitative Metrics

Basic TCRSeq analyses were initially done to characterize all samples presented in the manuscript. In order to characterize the distribution of sequences by frequency, we computed the clonality for all samples and characterized the distribution of sequences by their lengths (Supplementary Fig 3). The code used to generate these plots and do this analysis is attached in supplementary material.

Data Transformations

In order to allow a neural network to train from sequence data, we converted the amino acids to numbers between 0-19 representing the 20 possible amino acids. These were then one-hot encoded as to provide a categorical and discrete representation of the amino acids in numerical space. This process was applied prior to all networks being trained.

Training VAE

In order to train the VAE, following creation of the computational graph as described in the manuscript and main figure, we applied an Adam Optimizer (learning rate = 0.001) to minimize a reconstruction loss and a variational loss. The reconstruction loss is the cross-entropy loss between the reconstructed sequence (S) and the one-hot encoded tensor of the input sequence (L) across the *i*th position in the sequence (1). The variational loss is the Kullback–Leibler (KL) divergence between the distributions of the latent variables and a unit gaussian (2).

$$R_{Loss} = -\sum_{i} L_i \log(S_i) \quad (1)$$

$$V_{loss} = D_{KL}(N(\mu(X), \sigma(X)||N(0, I)) \quad (2)$$

The variational loss serves as a regularizer to the network as it prevents overfitting of the network and direct memorization of sequence to latent space and allows for meaningful downstream clustering of the sequences in their latent representation. The variational autoencoder was trained until the reconstruction accuracy over the penultimate 10 iterations was greater than 80%. Features for all sequences were then extracted from the latent space and used to create either heatmaps of sequences by features or a weighted average of the features by the frequency of the sequence was used to construct heatmap of samples by features.

Training GAN

In order to train the VAE, following creation of the computational graph as described in the manuscript and main figure, we applied an RMSProp Optimizer (learning rate = 0.0002) to simultaneously minimize the discriminator (3) and generator (4) loss. The training of a GAN can

be thought of an abstraction of the minimax algorithm where these two networks train in an adversarial fashion until the networks reach a Nash's equilibrium.

$$d_{loss} = -\frac{1}{2}E_{X\sim p_{data}}\log[D(x)] + -\frac{1}{2}E_Z\left[\log\left(1 - D\big(G(z)\big)\right)\right] \quad (3)$$

$$g_{loss} = -\frac{1}{2}E_Z\left[\log D\big(G(z)\big)\right] \quad (4)$$

Since this was the first example we could find of a GAN being used in biological sequence analysis, there were several modifications to the traditional GAN architecture used for image analysis to allow our network to train in a meaningful fashion. The first of these modifications was the input into the discriminator. When a generator is conventionally trained, it outputs an image with a given x by y dimensionality with 3 RGB dimensions that are continuous. However, in our applications, biological sequences are represented in a discrete space and this presented hurdles in getting the generator to create discrete representations. Therefore, the network was trained to output continuous sequence representations that were already embedded in a continuous domain. In a sense, the generator inputs its data in the middle of the discriminator, after the real data has already been embedded in a trainable embedding layer. The second point of alteration to the traditional GAN comes from the need for the discriminator to be a dimensionality reduction operation as oppose to the generator creating real sequences. In order to learn length invariant features, our discriminator has only one convolutional layer where the kernel is global max pooled across the length of the sequence. This operation creates our latent representation which is immediately fed into the final neuron for classification. The nature of this operation results in the generator creating sequences which are a conglomeration of motifs found in the original data as there is no feedback to the generator about length of the sequences in the original distribution of data. Finally, given the simplicity of the discriminator, we found the network was highly

susceptible to mode collapse, a type of failure where the generator outputs only sequence because it successfully fools the discriminator every time. In order to enforce a wide variety of generated motifs and sequences, we applied a feature matching algorithm where we add an additional loss to the generator (5).

$$g_{loss_2} = \frac{1}{n_{features}} \sum \left\| \mu\big(F(x)\big) - \mu\big(F\big(G(z)\big)\big) \right\| \quad (5)$$

This loss is the absolute difference between the average feature values for a batch of real data and fake data. This loss acts as a regularizer to encourage the generator to create diverse batches of sequences, capturing the entire distribution of the data. While this technique worked fairly well, we found the network could occasionally still suffer from mode collapse and further work is needed in the area of using GAN's for short sequences.

Finally, the network was trained in alternating fashion between the generator optimizer and discriminator optimizer over each iteration of the network. Training was halted when the average discriminator loss over the penultimate 10 iterations fell below 1.0 and the generator loss did not fall at least 1% in the penultimate 30 iterations (Supplementary Fig 4). We noted this type of early stopping criterion resulted in the generator initially fooling the discriminator quite easily until the discriminator learned the appropriate features to distinguish real from fake data. At this point, the generator loss would grow and eventually be unable to create sequences capable of fooling the discriminator and the training process was stopped at this point.

Training Single Sequence Classifier

In order to train the single sequence classifier, we followed a traditional conventional neural network architecture where a single translationally invariant convolutional layer was applied to

the sequence followed by three fully connected layers to a final classification layer. The network was trained using an Adam Optimizer (learning rate = 0.001) to minimize the cross-entropy loss between the softmaxed logits and the one-hot encoded representation of the discrete categorical outputs of the network. Training was conducted by using 75% of the data for the training set, and 25% for validation and testing. The validation group of sequences was used to implement an early stopping algorithm.

Training Whole Sample Classifier

Designing an architecture for whole sample multi-instance classification presented unique challenges that were specific to the way TCRSeq data is generated. Not only are the length of individual sequences variable length but the length of the individual files can vary in length as well in terms of number of unique sequences. Since neural networks required fixed-size inputs, this required not only a padding scheme for the sequences but also a padding scheme for the files. When a given dataset was imported, we applied a right zero padding scheme to each of the sequences but then we padded all zero sequences until every file had the same number of sequences. When this tensor is fed into the network, convolutional layers with dimensionality of [1, kernel] are then used to scan across the entire file of sequences. This results in feature values for each sequence in file. Additionally, since TCRSeq is a count-based NGS technology, there are quantitative measurements for each sequence that can be represented as a frequency of the entire file. This frequency is then used to weight the features. At this point, the network takes a sum of the features across all the sequences for a given file, computing a weighted average of all learned features over the entire sample. This vector of weighted average features is then fed directly into the classification layer. The network is trained with an Adam Optimizer (learning rate = 0.001) to minimize the cross-entropy loss between the softmaxed logits and the one-hot encoded

representation of the discrete categorical outputs of the network. Training splits and early stopping algorithms are the same as described above for the single sequence classifier except for in the cases of the *Sidhom_2017* and *Rudqvist_2017* datasets as the number of samples (7 & 20) were too small to create proper sized train/validation/test sets. Therefore, we used a leave-one-out training strategy where we trained on all but one sample until the training loss plateaued and then predicted on the one-out. Due to the generally small nature of these cohorts, in either traditional train/valid/test or leave-one-out splits, we employed monte carlo cross-validation, randomly selecting samples for train/test and iterating a number of times to approximate the predictive signature in the dataset.

Motif Identification

Neural networks are often treated as 'black boxes' where their value is largely in their predictive performance and not in understanding how the neural network is accomplishing its task. However, in the area of the biological sciences, there is not only the desire to create predictive tools but use these tools to inform our own understanding of the mechanisms at play. This area of research is often termed as improving the 'explainability of neural networks. In biological sequence analytics such as DeepTCR, investigators want to be able to extract the features/motifs the neural network learned to accomplish its task. For the supervised learning architectures, we were able to identify motifs the network had learned by extracting the indices of where the kernels were activated following the global max pooling layer. The result of this operation is the network not only extracts the maximum value of a kernel over the length of the sequence but also deduces its position within the sequence. This can be then used to not only pick up which features are activated on a given sequence but where in the sequence this activation occurs, allowing us to identify the motifs that

any given neuron in the net is learning. Sequence logos were created with https://weblogo.berkeley.edu/logo.cgi.

Code and Data availability

DeepTCR was written using Google's TensorFlow™ deep learning library and is available as a python package. Source code, comprehensive documentation, and use-case tutorials along with all data used in this manuscript can be found at https://github.com/sidhomj/DeepTCR. DeepTCR can either be installed directly from Github or from PyPI at https://pypi.org/project/DeepTCR/.

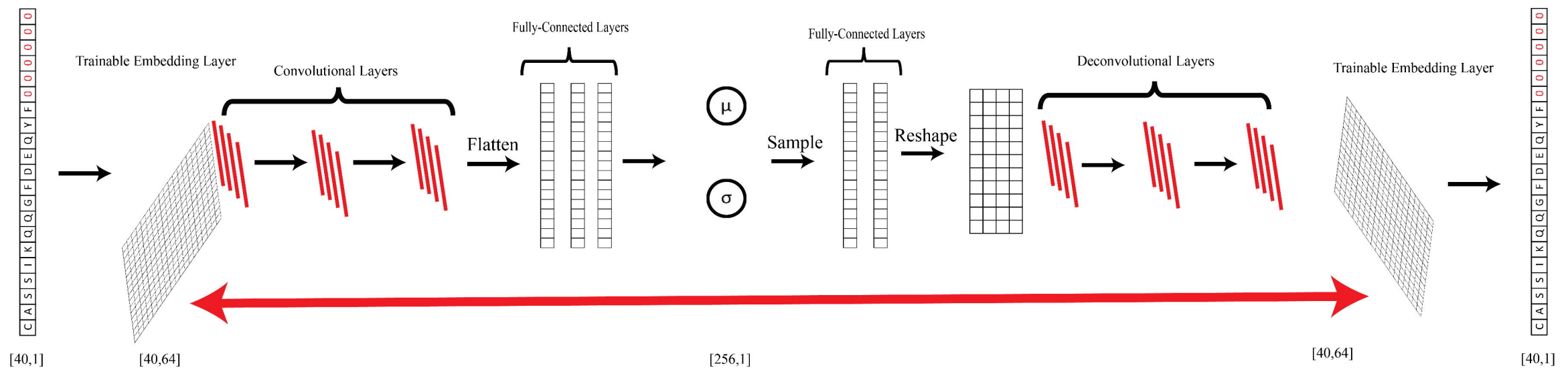
Acknowledgements

The authors thank the MARC/SU2C Foundation for providing financial support for the work of developing algorithmic pipelines presented in this manuscript. We also would like to thank James R. White for editorial assistance and reviewing the DeepTCR codebase.

Unsupervised Deep Learning Architectures

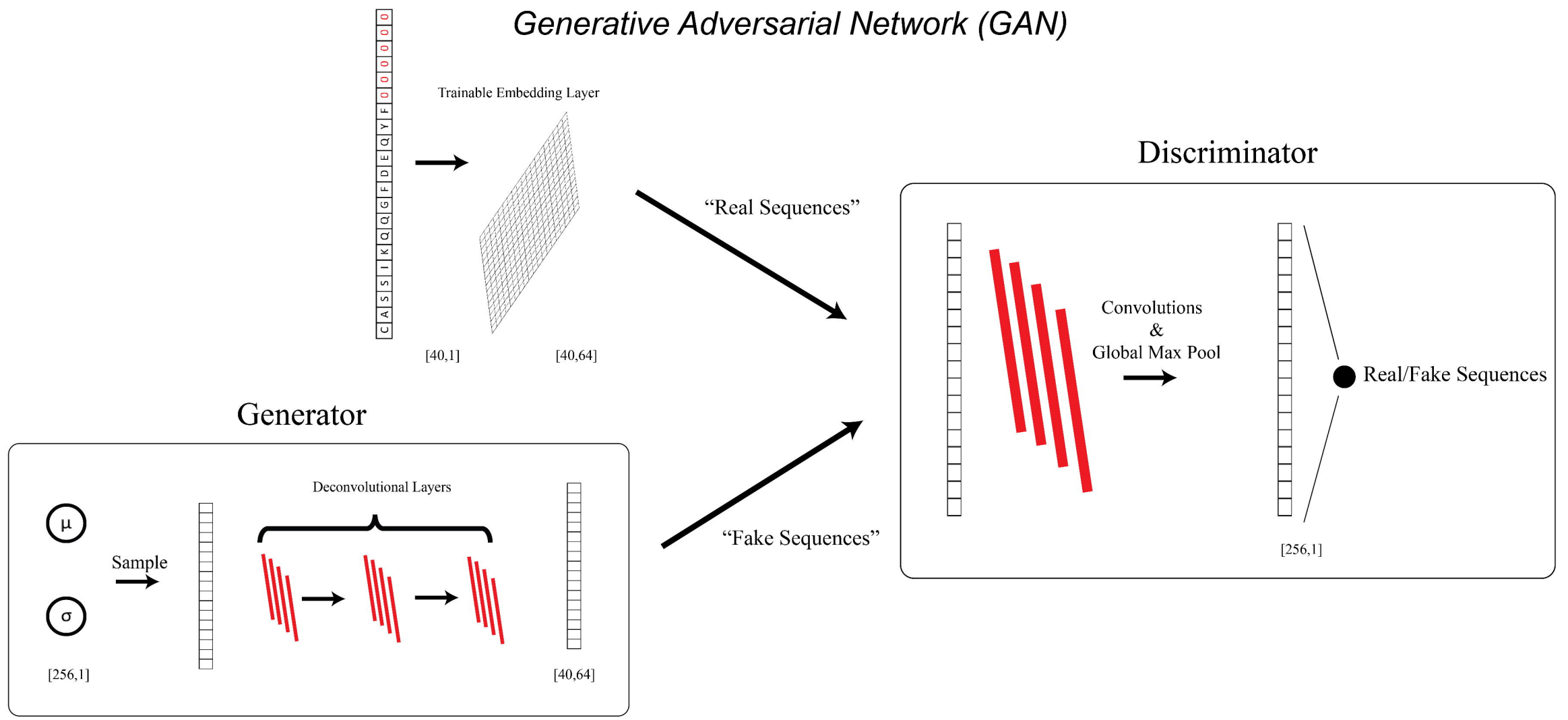
a

Variational Autoencoder (VAE)



b

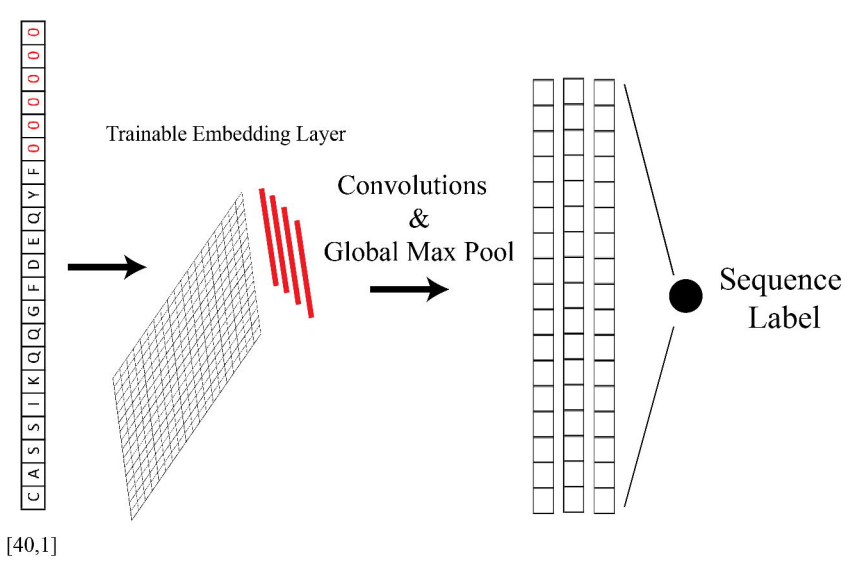
Generative Adversarial Network (GAN)



Supervised Deep Learning Architectures

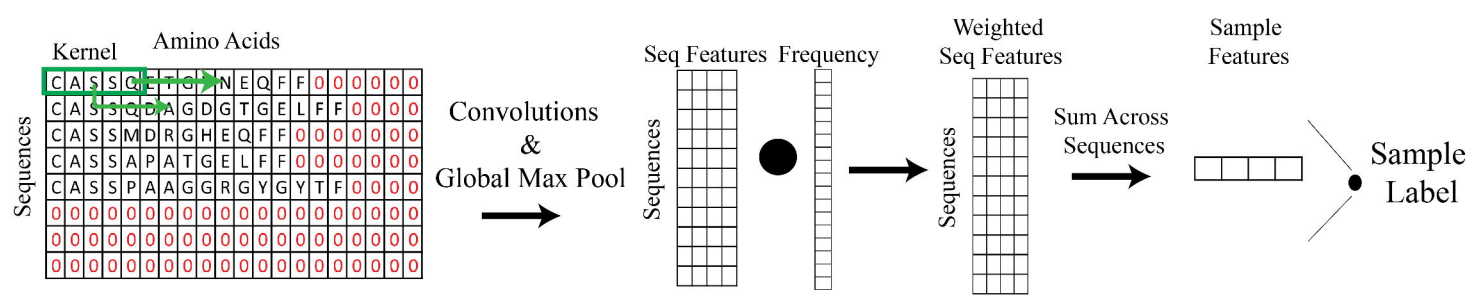
c

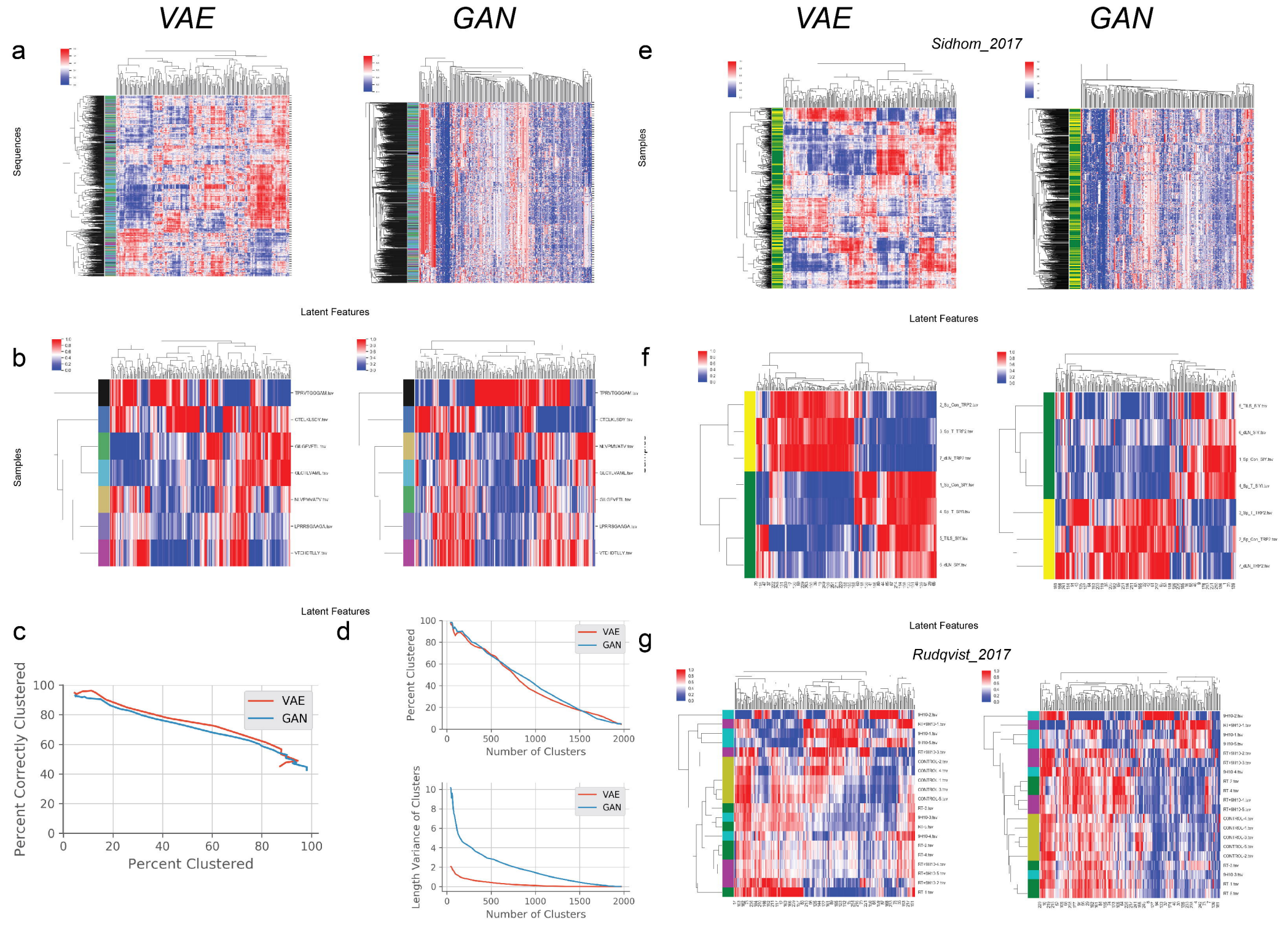
Single Sequence Classifier

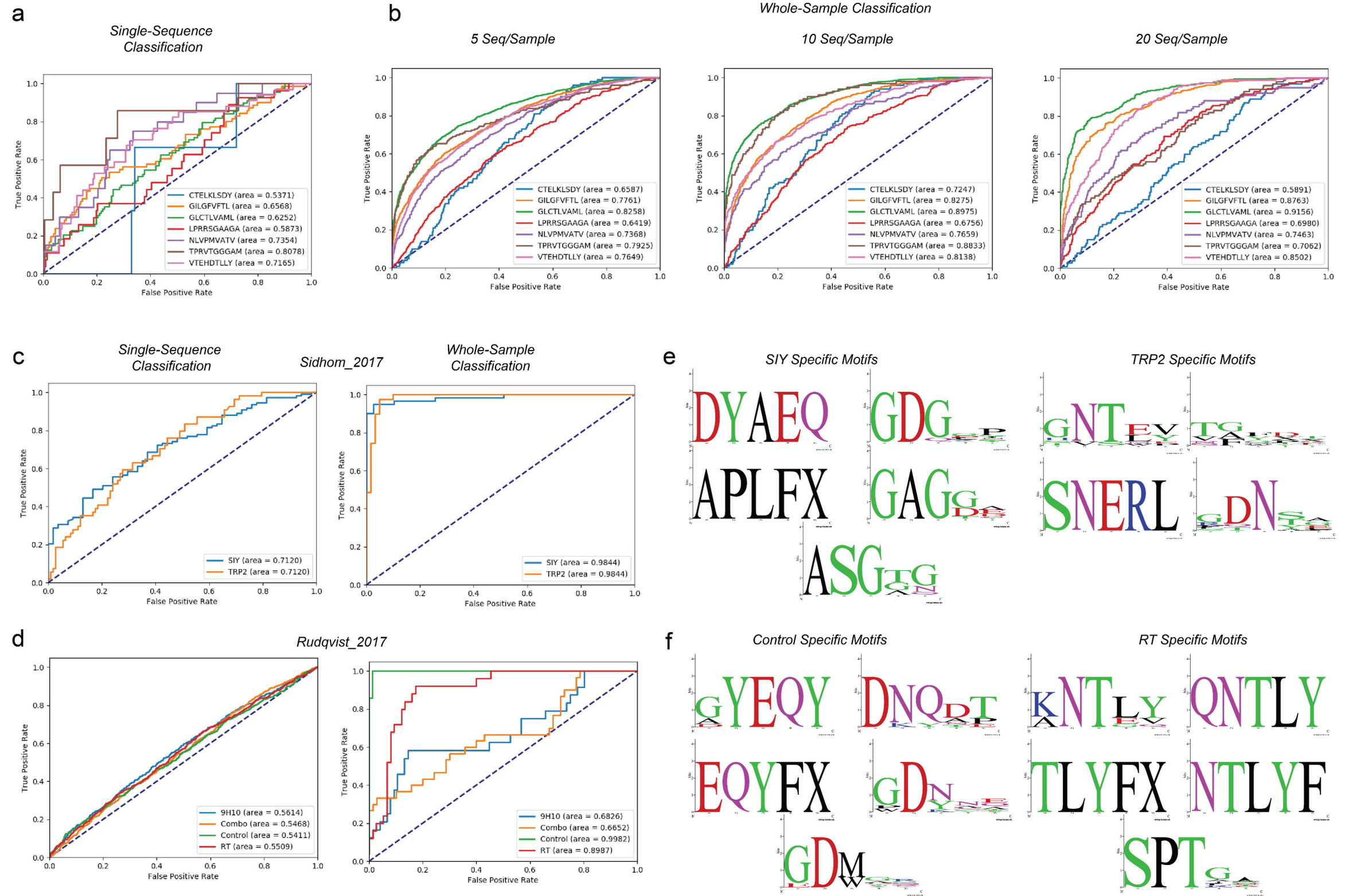


d

Whole Sample Classifier




VAE
GAN
a
Sequences
Latent Features
b
Samples
Latent Features
c
Percent Correctly Clustered
Percent Clustered
VAE
GAN
d
Percent Clustered
Number of Clusters
Length Variance of Clusters
Number of Clusters
VAE
GAN
e
Sidhom_2017
Samples
Latent Features
f
Latent Features
g
Rudqvist_2017
Latent Features


a
Single-Sequence Classification
True Positive Rate
False Positive Rate
CTELKLSDY (area = 0.5371)
GILGFVFTL (area = 0.6568)
GLCTLVAML (area = 0.6252)
LPRRSGAAGA (area = 0.5873)
NLVPMVATV (area = 0.7354)
TPRVTGGGAM (area = 0.8078)
VTEHDTLLY (area = 0.7165)
b
Whole-Sample Classification
5 Seq/Sample
CTELKLSDY (area = 0.6587)
GILGFVFTL (area = 0.7761)
GLCTLVAML (area = 0.8258)
LPRRSGAAGA (area = 0.6419)
NLVPMVATV (area = 0.7368)
TPRVTGGGAM (area = 0.7925)
VTEHDTLLY (area = 0.7649)
10 Seq/Sample
CTELKLSDY (area = 0.7247)
GILGFVFTL (area = 0.8275)
GLCTLVAML (area = 0.8975)
LPRRSGAAGA (area = 0.6756)
NLVPMVATV (area = 0.7659)
TPRVTGGGAM (area = 0.8833)
VTEHDTLLY (area = 0.8138)
20 Seq/Sample
CTELKLSDY (area = 0.5891)
GILGFVFTL (area = 0.8763)
GLCTLVAML (area = 0.9156)
LPRRSGAAGA (area = 0.6980)
NLVPMVATV (area = 0.7463)
TPRVTGGGAM (area = 0.7062)
VTEHDTLLY (area = 0.8502)
c
Sidhom_2017
Single-Sequence Classification
SIY (area = 0.7120)
TRP2 (area = 0.7120)
Whole-Sample Classification
SIY (area = 0.9844)
TRP2 (area = 0.9844)
d
Rudqvist_2017
9H10 (area = 0.5614)
Combo (area = 0.5468)
Control (area = 0.5411)
RT (area = 0.5509)
9H10 (area = 0.6826)
Combo (area = 0.6652)
Control (area = 0.9982)
RT (area = 0.8987)
e
SIY Specific Motifs
TRP2 Specific Motifs
f
Control Specific Motifs
RT Specific Motifs