Representation Learning of Genomic Sequence Motifs with Convolutional Neural Networks

Peter K. Koo[1], Sean R. Eddy[1,2]

April 11, 2019

1. Howard Hughes Medical Institute, Department of Molecular and Cellular Biology, Harvard University, Cambridge, MA
2. John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA

Abstract

Although convolutional neural networks (CNNs) have been applied to a variety of computational genomics problems, there remains a large gap in our understanding of how they build representations of regulatory genomic sequences. Here we perform systematic experiments on synthetic sequences to reveal how CNN architecture, specifically convolutional filter size and max-pooling, influences the extent that sequence motif representations are learned by first layer filters. We find that CNNs designed to foster hierarchical representation learning of sequence motifs – assembling partial features into whole features in deeper layers – tend to learn *distributed* representations, *i.e.* partial motifs. On the other hand, CNNs that are designed to limit the ability of hierarchically building sequence motif representations in deeper layers tend to learn more interpretable *localist* representations, *i.e.* whole motifs. We then validate that this representation learning principle established from synthetic sequences generalizes to *in vivo* sequences.

Introduction

Deep convolutional neural networks (CNNs) have recently been applied to predict transcription factor (TF) binding motifs from genomic sequences (Zhou and Troyanskaya, 2015; Quang and Xie, 2016; Kelley *et al*, 2016; Hiranuma *et al*, 2017). Despite the promise that CNNs bring in replacing methods that rely on k-mers and position weight matrices (PWMs) (Ghandi *et al*, 2016; Foat *et al*, 2006), there remains a large gap in our understanding of why CNNs perform well.

A convolutional layer of a CNN is comprised of a set of filters, each of which can be thought of as a PWM. Each filter scans across the inputs, and outputs a non-linear similarity score at each position, a so-called feature map. The filters are parameters of the CNN that are trained to detect relevant patterns in the data. Deep CNNs are constructed by feeding the feature maps of a convolutional layer as input to another convolutional layer. This can be repeated to create a network with any depth. CNNs typically employ *max-pooling* after each convolutional layer, which down-sample the feature maps by setting non-overlapping windows with a single maximum score, separately for each filter. Max-pooling enables deeper layers to detect features hierarchically across a larger spatial scale of the input. CNN predictions are then made by feeding the feature map of the final convolutional layer through a fully-connected hidden layer followed by a linear classifier.

In genomics, it is unclear how CNN architecture influences the representations of sequence motifs learned throughout the network. Previous studies have suggested that the first convolutional layer filters learn representations of sequence motifs, while deeper layers learn combinations of these motifs, so-called regulatory grammars (Alipanahi *et al*, 2015; Angermueller *et al*, 2016; Zeng *et al*, 2016; Quang and Xie, 2016; Kelley *et al*, 2016). Since first layer filters can be directly visualized, demonstrating that a CNN learns relevant sequence motifs is a common strategy for validation. The few studies that perform a quantitative motif comparison of the first layer filters against a motif database find that less than 50% have a statistically significant match (Kelley *et al*, 2016; Quang and Xie, 2016). Unmatched filters have been suggested to be either partial representations of known motifs or novel motifs, *i.e.* motifs not included in the database. Visualization of deeper layer filters is challenging, because they represent patterns of feature maps in previous layers, where the spatial positions of activations are obscured by pooling.

Learning whole-motif representations by first layer filters is not indicative of a deep CNN's classification performance. For instance, a deep CNN that employs a small first layer filter, *i.e.* 8 nts (Zhou and Troyanskaya, 2015), which is shorter than many common motifs found *in vivo*, has demonstrated comparable performance as CNNs that employ larger filters, *i.e.* $\geq$19 nts (Quang and Xie, 2016; Kelley *et al*, 2016). In principle, smaller filters that capture partial motif representations can be combined in deeper layers to assemble whole-motif representations, thereby allowing the CNN to make accurate predictions. It remains unclear to what extent we should expect first layer filters to learn whole-motif representations in the first convolutional layer and how a CNN's architecture influences this.

Here we demonstrate how architectural choice affects representation learning of genomic sequence motifs. We perform systematic experiments to demonstrate that a CNN's design, specifically max-pooling and filter size, is indicative of the extent that motif representations are learned in first layer filters. We then demonstrate that the same representation learning principles generalize to *in vivo* sequences.

Results

Internal representations of motifs depend on architecture

We conjecture that motif representations learned in first layer filters are largely influenced by a CNN's ability to assemble whole-motif representations in deeper layers, which is determined by architectural constraints set by: 1. the convolutional filter size, 2. the stride of the filter, which is the offset between successive applications of the filter (usually set to 1), 3. the max-pool size, and 4. the max-pool stride, which is the offset for each max-pool application. Although these are hyperparameters that may vary on a case-by-case basis, the max-pool stride is commonly set to the max-pool size in genomics, creating non-overlapping max-pool windows.

Despite the complexity of *in vivo* TF binding (Siggers and Gordan, 2013), for the purposes of this paper we make a simplifying assumption that TF binding sites can be represented by a single PWM-like motif pattern. Of course, a PWM-based method would perform well in this over-simplified scenario. However, the scope of this paper is to demonstrate how representations of sequence patterns are learned by a CNN and not a thorough demonstration of a CNN's ability to learn *in vivo* binding sites of TFs.

Assuming that accurate classification can only be made if the correct motifs are detected, a CNN that learns partial-motif representations in the first layer must assemble whole-motif representations at some point in deeper layers. To help explain how architecture can influence representation learning in a given layer, we use the concept of a receptive field, which is the sensory space of the data that affects a given neuron's activity. For the first convolutional layer, each neuron's receptive field has a size that is equal to the filter size at a particular region of the data. Since there are typically many filters in a convolutional layer, there are many neurons which have a receptive field that share the same spatial region. However, each neuron's activation is determined by a different filter. Max-pooling combines multiple neurons of a given filter within a specified window size to a single max-pooled neuron, thereby augmenting the size of its receptive field. In doing so, max-pooling obfuscates the exact positioning of the max-activation within each window. Thus the location of the max-activation has spatial invariance within its receptive field with an amount equal to the max-pool size.

Although max-pooling creates spatial uncertainty of the max-activation within a max-pooled neuron's receptive field, we surmise that neighboring max-pooled neurons of different filters, which share significantly overlapping receptive fields, can help to resolve spatial positioning of an activation. To illustrate, Figure 1A shows a toy example of two convolutional filters, each 7 nts long, which have learned partial-motifs: 'GTG' and 'CAC'. An example sequence contains three embedded patterns (highlighted in green): 'CACGTG', 'GTGCAC', and 'CACNNNGTG', where 'N' represents any nucleotide with equal probability. The resultant max-pooled, activated convolutional scans for each filter are shown above the sequence with a blue shaded region highlighting the receptive field of select max-pooled neurons. Even though the first convolutional layer filters have learned partial-motifs, the second convolutional layer filters can still resolve each of the three embedded patterns by employing filters of length 3. Of course situations may arise where the three second convolutional layer filters are unable to fully resolve the embedded patterns with fidelity. For instance, 'CACNGTG' could be activated by the same filter for 'CACGTG'. A CNN can circumvent these ambiguous situations by either learning more information about each pattern within each filter or by dedicating additional filters to help discriminate the ambiguous patterns.

It follows that by creating a situation where partial-motif representations cannot be assembled into whole-motifs in deeper layers, learning whole-motifs by first layer filters becomes obligatory for accurate classification. One method to limit the information flow through a CNN is by employing large max-pool sizes relative to the filter size. The max-pooled neurons then have large receptive fields with a large spatial uncertainty and only a small overlap in receptive fields with neighboring neurons of different filters. A deeper layer would be unable to resolve the spatial ordering of partial motifs to assemble whole-motifs with fidelity. To exemplify, figure 1B shows a toy example of a CNN that employs a larger pool size of 20. Importantly, there are large spatial regions within a receptive field for which a neighboring neuron cannot help to resolve

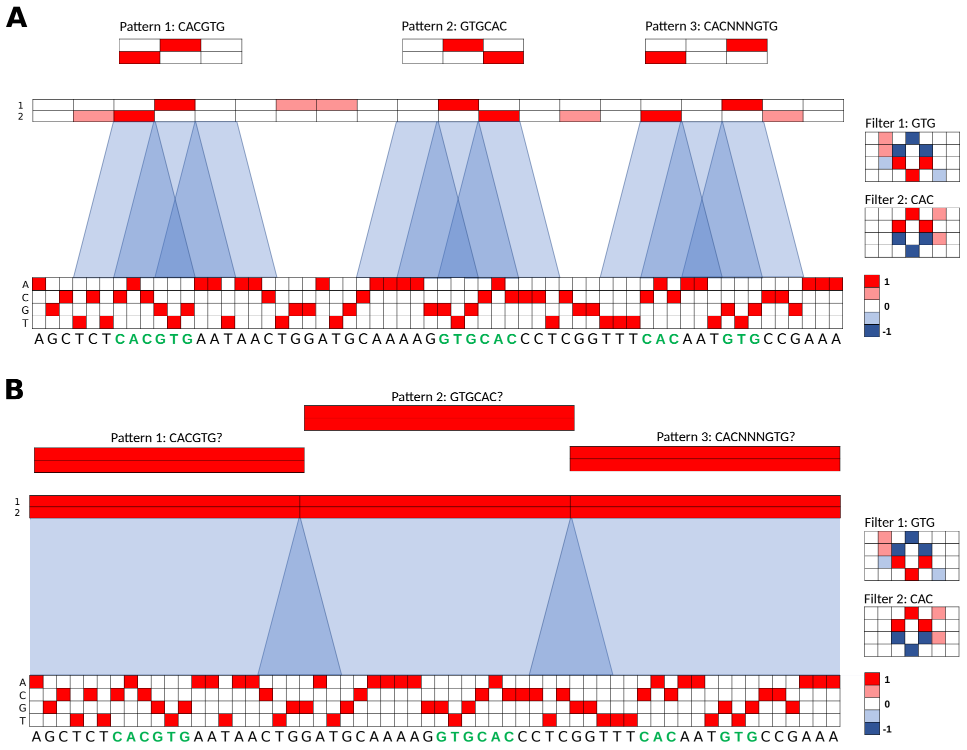

Figure 1: Toy model for representation learning of sequence motifs. (A,B) An example 60 nt one-hot encoded sequence contains 3 patterns (shown in green): CACGTG, GTGCAC, and CACNNNGTG. Two filters, each of length 7 (7 columns and 4 rows, one for each nucleotide), are shown to the right. A partial-motif representation has been captured by each filter: GTG for filter 1 (Top) and CAC for filter 2 (Bottom). The max-pooled feature maps are shown above the sequence. The feature maps have the same size as the sequence by adding 3 zero-padding units to each end of the sequence prior to convolution (not shown in diagram). (A) Shows the feature maps when employing a small max-pooling size of 3, which creates overlapping receptive fields, highlighted in blue. 3 second layer convolutional filters, shown above, demonstrate a feature map pattern that can resolve each embedded sequence pattern. (B) Shows the feature maps when employing a larger pooling size of 20 using the same filters as (A). The larger receptive fields have a large spatial uncertainty along with a small overlap in receptive fields from neighboring neurons. Each of the 3 second layer convolutional filters, shown above, is unable to find a unique feature map pattern that can resolve any embedded sequence pattern.

due to a lack of overlap in receptive fields. As a result, deeper convolutional layer filters which are dedicated to each pattern would yield the same signature, unable to resolve any of the three patterns.

More technically, the extent of motif information that each filter learns is guided by the gradients of the objective function, which serves as a measure of the classification error. Assuming accurate classification can only be achieved upon discriminating the underlying motifs of each class, once whole-motifs for each class are learned, then the objective function is minimized and the training gradients go to zero. If a CNN can build whole-motifs from partial-motifs in deeper layers, then there is no more incentive to learn additional information to build upon the partial-motif representations already learned. As a result, the first layer filters will maintain a *distributed* representations of motifs (Hinton *et al*, 1986). On the other hand, if architectural constraints limit the ability to build whole-motifs from partial-motifs in deeper layers, then accurate predictions cannot be made. Hence, gradients will persist because the objective function is not yet minimized, encouraging first layer filters to learn whole-motifs, also known as a *localist* representation of motifs (Hinton *et al*, 1986). Once the first layer filters have learned sufficient information of whole-motifs to discriminate each class, then the objective function can be minimized, signaling the end of training.

Max-pooling influences ability to build hierarchical motif representations

To test this idea, we created a synthetic dataset that enforces our simplifying assumption that the only important pattern for a given TF binding event is the presence of a PWM-like motif in a sequence. Briefly, synthetic sequences, each 200 nts long, were implanted with 1 to 5 known TF motifs, randomly selected with replacement from a pool of 12 transcription factor motifs embedded in random DNA (see Methods for details). The motifs were manually selected from the JASPAR database to represent a diverse, non-redundant set. The goal of this computational task is to simultaneously make 12 binary predictions for the presence or absence of each transcription factor motif in the sequence. Since we have ground truth for all of the relevant TF motifs and where they are embedded in each sequence, we can test the efficacy of the representations learned by a trained CNN. We note that the ground truth is only from embedded motifs and not from motifs that occasionally arise by chance; the latter effectively creates false negative labels in this dataset.

A CNN that employs at least two convolutional layers is required to test our hypotheses of representation learning. We constructed a CNN with 3 hidden layers: two convolutional layers, each followed by max-pooling, and a fully-connected hidden layer. Specifically, our CNN takes as input one-hot encoded sequences, processes them with the hidden layers, and outputs a prediction for the binding probability for each of the 12 classes. The number of filters in each convolutional layer, the number of units in the fully-connected hidden layer, and the dropout probabilities are fixed (see Methods). The filter sizes, the max-pool window sizes, and the max-pool strides are the hyperparameters that can be varied. For a given hyperparameter setting, we trained the CNN as a multi-class logistic regression (see Methods for training details). All reported metrics are strictly drawn from the held-out test set using model parameters that yielded the best performance on the validation set.

To explore how spatial uncertainty within receptive fields set by max-pooling influences the representations learned by first layer filters, we systematically altered the max-pool sizes while keeping all other hyperparameters fixed, including a first and second layer filter size of 19 and 5, respectively. To minimize the influence of architecture on classification performance, we coupled the max-pool sizes between the first and second layer, such that their products are equal, which makes the inputs into the fully-connected hidden layer the same size across all CNNs. The max-pool sizes we employed are (first layer, second layer): (2, 50), (4, 25), (10, 10), (25, 4), (50, 2), and (100,1). For brevity, we denote each CNN with only the first max-pool window size, *e.g.* CNN-2 for (2, 50).

We first verified that the performance of each model is similar as measured by the average area-under the receiver-operator-characteristic (AU-ROC) curve across the 12 classes (Table 1), which is in the range of previously reported values for a similar task using experimental ChIP-seq data (Zhou and Troyanskaya, 2015; Quang and Xie, 2016). Next, we converted each filter to a sequence logo to visually compare the motif represenations learned by the first layer filters across the different architectures (see Methods). As expected, we found CNNs that employ large max-pool sizes ($\geq$10) learn representations that qualitatively resemble the ground truth motifs (Fig. 2). On the other hand, CNNs that employ a small max-pool size ($\leq$4) do not seem to qualitatively capture any ground truth motif in its entirety, perhaps learning, at best, parts of a

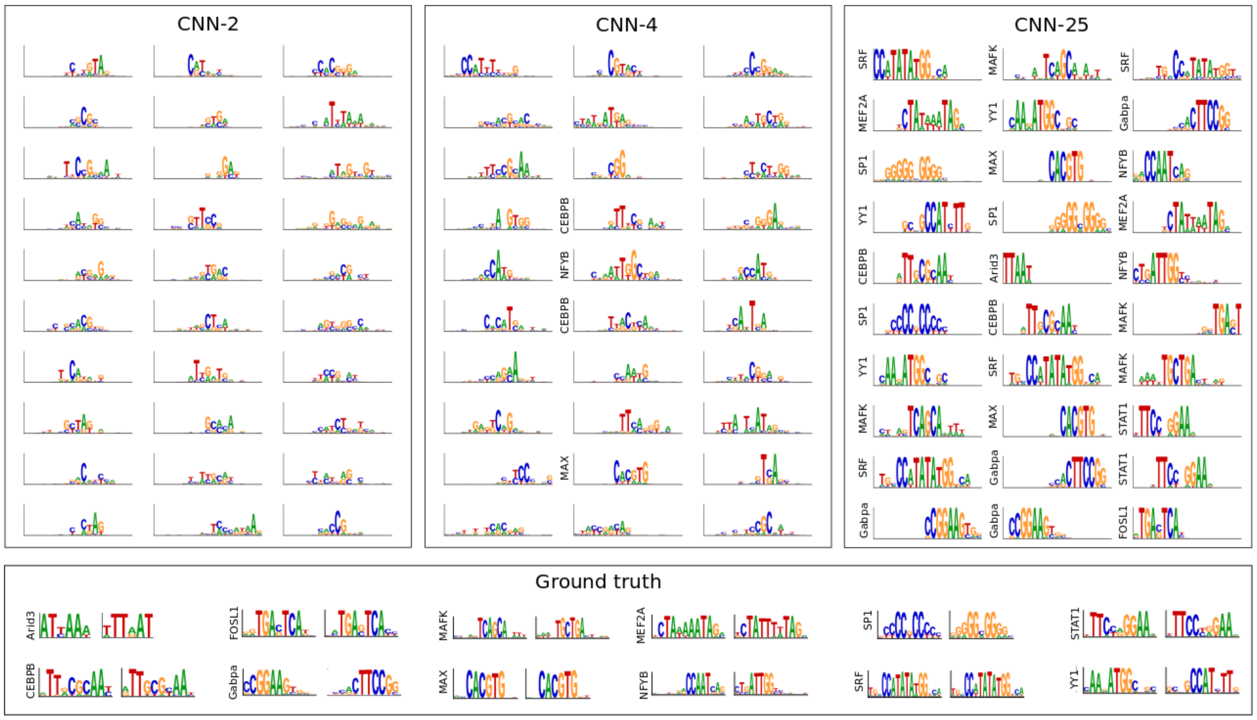

Figure 2: Comparison of first layer filters for CNNs with a different max-pool size. Sequence logos for normalized first convolutional layer filters are shown for CNN-2 (Left), CNN-4 (Middle), and CNN-25 (Right). The sequence logo of the ground truth motifs and its reverse complement for each transcription factor is shown at the bottom. The y-axis label on select filters represents a statistically significant match to a ground truth motif.

motif.

To quantify the number of filters that have learned motifs, we employed the Tomtom motif comparison search tool (Gupta *et al*, 2007) to compare the similarity of each filter against all motifs in the JASPAR 2016 vertebrate database (Mathelier *et al*, 2016) using an E-value cutoff of 0.1. In agreement with our qualitative observation, we found CNNs that employ a small max-pool size (≤ 4) have, at best, 57% of their filters match any known motifs. Of these, only 3 filters in CNN-4 matches a ground truth motif. In contrast, CNNs that employ a large max-pool size yield, at worst, a 97% match to ground truth motifs (Table 1).

Motif representations are not very sensitive to 1st layer filter size

Because it has been widely thought that first convolutional layer filters learn motifs, deep learning practitioners have traditionally employed CNN architectures with large first layer filters to capture motif patterns in their entirety. However, we have shown that employing a large filter does not necessarily lead to whole-motif representations. To test the sensitivity of filter size to representation learning, we created two new CNNs that employ a first layer filter size of 9 (CNN$_9$), in contrast to a filter size of 19 which was previously used, with max-pool combinations of 4 and 25, *i.e.* CNN$_9$-4 and CNN$_9$-25. Since the combination of a filter size of 9 with a max-pool size of 4 creates overlapping receptive fields with a small spatial uncertainty, we expect that this architecture setting will lead to partial-motif representations. On the other hand, the filter size of 9 is insufficient to resolve spatial positions when employing a max-pool size of 25. Hence, we predict that this architecture setting will yield whole-motif representations. As expected, CNN$_9$-25 learns representations that qualitatively better reflect the ground truth motifs compared to CNN$_9$-4 (Fig. 3, A-B). Interestingly, CNN$_9$-25 also learns partial motif representations of larger motifs, *i.e.* MEF2A, SRF, STAT1, CEBPB, but in a more visually identifiable way compared to CNN$_9$-4. By quantifying the percentage of filters that statistically match ground truth motifs, CNN$_9$-25 yields an 93% match compared to CNN$_9$-4 which only

Table 1: Performance on the synthetic dataset. The table shows the average area under the receiver-operator-characteristic curve (AU-ROC) across the 12 TF classes, percentage of matches between the 30 first convolutional layer filters and the entire JASPAR vertebrates database (JASPAR), and the percentage of filters that match to any ground truth TF motif (Relevant) for different CNNs. Motif matches were determined by the Tomtom motif comparison search tool using an E-value cutoff of 0.1. The Average AU-ROC error represents the standard deviation of the AU-ROC across the 12 classes.

Model	Average AU-ROC	% Motif match (JASPAR)	% Motif match (Relevant)
CNN-2	0.968±0.028	30	0
CNN-4	0.952±0.056	57	13
CNN-10	0.970±0.038	100	97
CNN-25	0.954±0.094	100	100
CNN-50	0.965±0.040	100	97
CNN-100	0.961±0.042	97	97
CNN_9-4	0.958±0.039	13	0
CNN_9-25	0.961±0.038	93	93
CNN_3-2	0.968±0.027	33	0
CNN_3-50	0.652±0.060	17	0
CNN-50-2	0.917±0.136	97	90
$\text{CNN}_{19\text{-}1}$-2	0.969 ±0.033	90	87

yield no matches (Table 1).

As a control, we created a new CNN with a filter size of 3 with max-pool size combinations of 2 and 50, *i.e.* CNN_3-2 and CNN_3-50. Even though the filter size is smaller than many of the embedded motifs, we expect that CNN_3-2 will still be able to assemble whole motifs to some extent in deeper layers, because it employs small max-pooling. On the other hand, since CNN_3-50 has only one chance to learn whole motifs, we expect that the small filter size will be unable to discriminate the embedded motifs for each class, leading to a poor classification performance. Indeed, CNN_3-50 yields a mean AU-ROC of 0.652±0.060 across the 12 classes, compared to CNN_3-2 which yields 0.968±0.039 (error is the standard deviation across the 12 classes).

Representations of motifs are affected by the ability to assemble whole-motifs in deeper layers

One aspect of max-pooling that we did not consider in our toy model is the max-pool stride, which is typically set to the max-pool size. Employing a large max-pool size with a small max-pool stride can create a situation where the receptive field of max-pooled neurons overlap significantly, which should improve the spatial resolution of partial-motifs. However, a deeper convolutional filter would still be unable to assemble whole motifs, because each receptive field has a large spatial uncertainty, which provides the same activation pattern irrespective of whether partial motifs are close together or very distant. Hence, we expect this CNN to learn whole-motif representations.

To test this, we created a new CNN which employs a large max-pool size of 50 with a max-pool stride of 2 (CNN-50-2). Consequently, the length of the feature maps after the first convolutional layer are half of the input sequence, which is the same shape as the feature maps of CNN-2, which employs a max-pool size of 2 with a stride of 2. Similar to CNN-2, CNN-50-2 employs a max-pool size and stride of 50 after the second convolutional layer. As expected, CNN-50-2 learns whole-motif representations with 90% of its filters matching ground truth motifs in the synthetic dataset (Table 1). Moreover, the motifs that are learned by CNN-50-2 qualitatively better resemble whole-motif representations (Fig. 3) compared to CNN-2 (Fig. 2). Together, this result further supports that architecture, specifically the ability to assemble whole-motifs in deeper layers, plays a major role in how CNNs learn genomic representations in a given layer.

Another factor that can affect the ability to assemble whole-motifs in deeper layers is the size of second

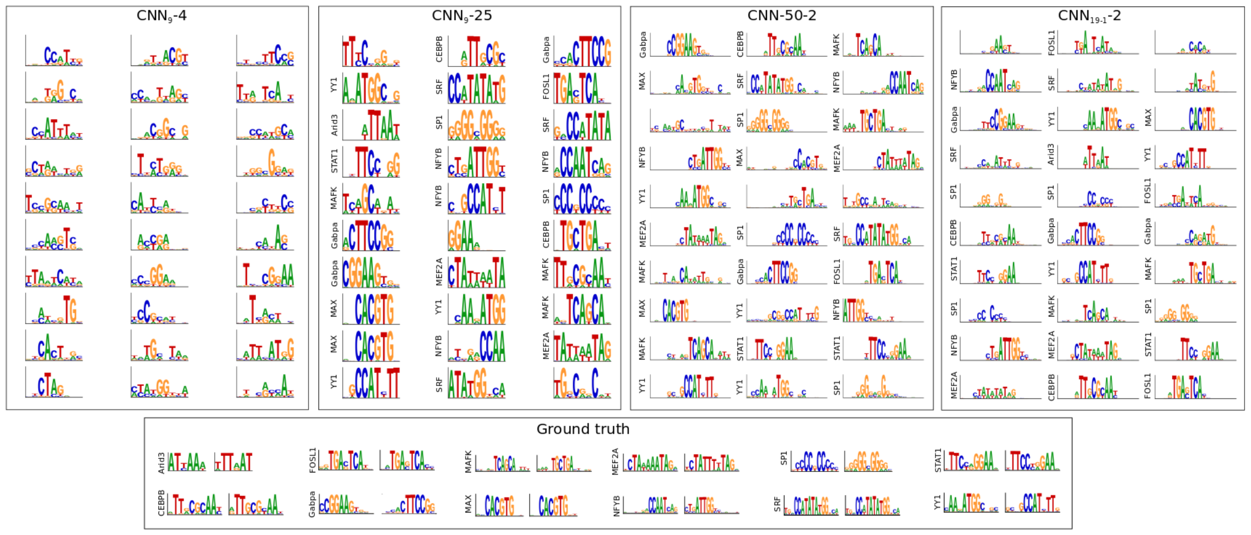

Figure 3: Representations learned by first layer filters for alternative CNN architectures. Sequence logos for normalized first convolutional layer filters are shown for (from left to right): CNN$_9$-4, CNN$_9$-25, CNN-50-2, and CNN$_{19\text{-}1}$-2. The sequence logo of the ground truth motifs and its reverse complement for each transcription factor is shown at the bottom. The y-axis label on select filters represent a statistically significant match to a ground truth motif.

convolutional layer filters. A small filter size can make it challenging to assemble whole motifs, even if the max-pool size and stride are small. To test this, we modified the second convolutional filter size of CNN-2 from a size of 5 to 1 (CNN$_{19\text{-}1}$-2). As expected, the first layer filters learn representations of whole-motifs (87% match to ground truth motifs) even though CNN$_{19\text{-}1}$-2 employs a small max-pool size of 2 (Fig. 3 and Table 1).

Distributed representations build whole-motif representations in deeper layers

The high overall classification performance of each CNN suggests that they must have learned whole-motif representations at some point. Thus, CNN-2, whose first layer filters do not match any relevant motifs, must be assembling whole-motif representations in deeper layers. To verify that CNN-2 eventually learns whole-motif representations, we visualize the learned representation with saliency analysis, specifically guided-backpropagation (Springenberg *et al*, 2014) (see Methods). Saliency analysis provides the sensitivity that each nucleotide variant in a given sequence towards the network's predictions. Importantly, saliency analysis considers the entire network, integrating across distributed representations learned throughout the network. Unlike standard backpropagation (Simonyan *et al*, 2013), guided-backpropagation does not provide class-specific attribution scores (Kindermans *et al*, 2017; Shrikumar *et al*, 2017); instead, it has been shown to provide edge detection when applied to images (Nie *et al*, 2018). Nevertheless, we have found that it provides more *human-interpretable* representations of motifs. To limit false-positive representations of motifs from different classes, we only visualize guided-backprop saliency maps from test sequences that have a single class label.

The representative sequence logo of a saliency map generated by guided-backpropagation applied to CNN-2 and CNN-25 for sequences associated with different TF classes confirms that the underlying motif representations are indeed learned irrespective of whether the first layer learns whole-motifs or partial-motifs (Fig. 4). Interestingly, CNN$_3$-2 is also able to largely learn representations of whole-motifs, despite employing a very small first layer filter size of 3 (Fig. 4).

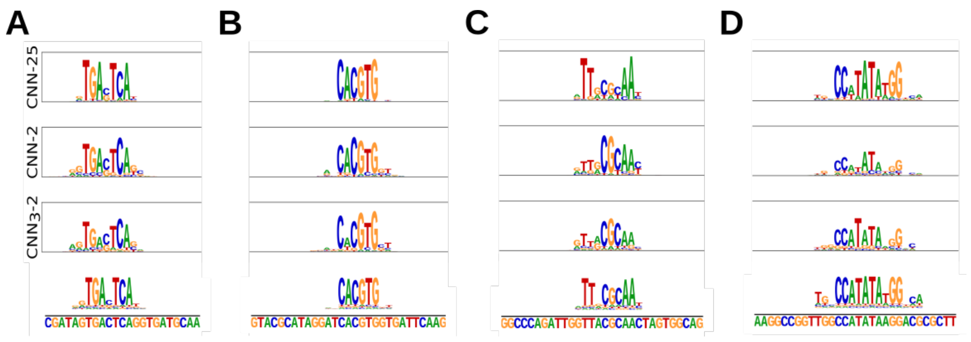

Figure 4: Representative sequence logos of the saliency maps generated by CNNs. Sequence logos of the saliency maps generated by CNN-25, CNN-2, and CNN$_3$-2 for a sequence that contains a label for (A) FOSL1, (B) MAX, (C)CEBPB, and (D) SRF. The underlying sequence and the sequence logo of the sequence model (ground truth) is shown below. A saliency map was generated by employing guided-backprop from the pre-activated output neuron with respect to the input layer. The saliency map was then normalized to a PWM and converted to a sequence logo. Each saliency map was further clipped about the embedded motif for brevity. The unclipped saliency plots are shown in Supplemental Fig. S1.

Generalization to *in vivo* sequences

To test whether the same representation learning principles generalize to *in vivo* sequences, we modified the DeepSea dataset (Zhou and Troyanskaya, 2015) to include only *in vivo* sequences that have a peak called for at least one of 12 ChIP-seq experiments, each of which correspond to a TF in the synthetic dataset (see Supplemental Table S1). The truncated-DeepSea dataset is similar to the synthetic dataset, except that the input sequences now have a size of 1,000 nt in contrast to the 200 nt sequences in the synthetic dataset.

We trained each CNN on the *in vivo* dataset following the same protocol as the synthetic dataset. Similar to CNNs trained on the synthetic dataset, a qualitative comparison of the first layer filters of different CNNs show that employing a larger pool size yield representations that better reflect whole-motifs (Fig. 5). By employing the Tomtom motif comparison search tool, we quantified the percentage of statistically significant hits between the first layer filters against the JASPAR database (see Table 2). Similar to the synthetic dataset, CNNs that employ a smaller max-pool size (≤ 4) yield a percent match that is, at best, 57% (Table 2). In contrast, CNNs that employ a larger max-pool size (≥ 10) yield a percent match that is, at worst, 83% (Table 2). Since *in vivo* sequences contain many additional signals compared to the synthetic sequences, we were unable to reliably quantify the percentage of filters that learn *relevant* motifs. Interestingly, we found that each CNN consistently struggled to identify *known* motifs for ARID3A, SP1, and STAT1. However, it is unclear whether this arises because of: experimental or post-processing errors which create label noise in the sequences we assign as having a ChIP-seq peak, the large variance in numbers of sequences for different classes (class imbalance), and/or an inability of the CNNs to learn the correct motif, among the many other possible explanations. Notwithstanding, the same trends in the amount of motif information learned by first layer filters *in vivo* suggests that we have identified a general principle for representation learning by CNNs.

Discussion

Here, we reveal principles of how architecture design influences representation learning of sequence motifs by exploring different CNN architectures on a synthetic dataset with a known ground truth. Typical deep CNN architectures in genomics employ large filters and small max-pool sizes, which we would expect to learn *distributed* representations of sequence motifs. Hence, visualization of first convolutional layer filters is not as meaningful for these CNNs. However, we showed that *localist* representations, *i.e.* whole motifs, can be learned by constraining the architecture such that the ability of deeper layers cannot reliably assemble hierarchical representations of motifs. Although this study focuses on first layer representations, we believe that the same principles hold for deeper layers in the network. While we only explored the role of architecture in this study, we note that there may be other factors that contribute to the quality of the learned

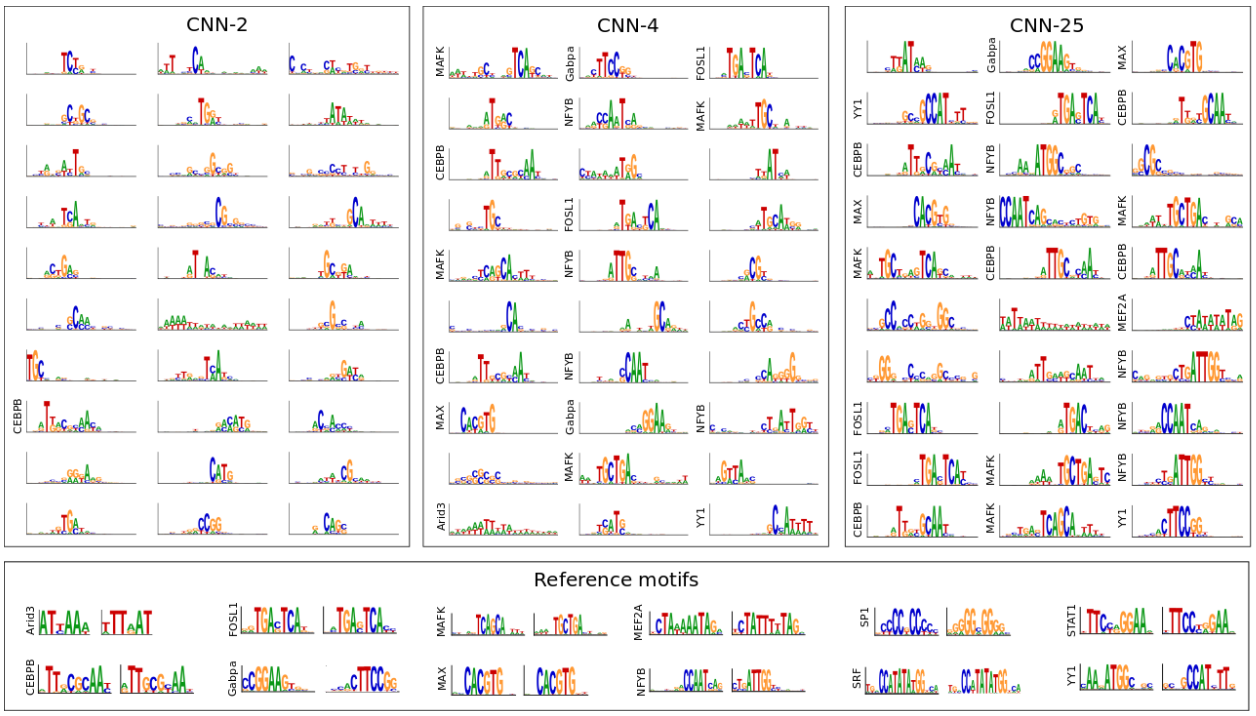

Figure 5: Comparison of the first layer filters for CNNs trained on *in vivo* sequences. Sequence logos for normalized first convolutional layer filters are shown for CNN-2 (Left), CNN-4 (Middle), and CNN-25 (Right). The sequence logos of reference motifs and their reverse complements for each transcription factor from the JASPAR database is shown at the bottom. The y-axis label on select filters represents a Tomtom match to a reference motif.

representations, including regularization and optimization algorithms.

There are many caveats to interpreting a CNN by just visualizing first layer filters, even if it learns whole-

Table 2: Performance of deep learning models on the *in vivo* dataset. The table shows the average area under the receiver-operator-characteristic curve (AU-ROC) and the average area under the precision recall curve (AU-PR) across the 12 TF classes, percentage of matches between the 30 first convolutional layer filters and the entire JASPAR vertebrates database (JASPAR), and the percentage of filters that match to any ground truth TF motif (Relevant) for different CNNs. Motif matches were determined by the Tomtom motif comparison search tool using an E-value cutoff of 0.1. The average AU-ROC error and the average AU-PR error represents the standard deviation across the 12 classes.

Model	Average AU-ROC	Average AU-PR	Motif match (JASPAR)	Motif match (Relevant)
CNN-2	0.906±0.050	0.598±0.259	23	3
CNN-4	0.907±0.048	0.601±0.255	57	30
CNN-10	0.901±0.053	0.584±0.267	93	77
CNN-25	0.900±0.060	0.580±0.291	90	77
CNN-50	0.901±0.054	0.583±0.277	83	70
CNN-100	0.889±0.061	0.556±0.284	93	73
CNN^9-4	0.900±0.052	0.565±0.281	56	30
CNN^9-25	0.882±0.066	0.539±0.296	83	57
CNN-50-2	0.890±0.070	0.553±0.316	87	60

motif representations. This approach only represents the sequence features that are learned across the entire dataset. It does not inform how the CNN combines these features to make predictions. If the computational task is multi-task classification, there is no correspondence between features and their respective class. In addition, not all filters will learn motifs, and a motif comparison search does not necessarily identify whether a CNN learns *relevant* motifs. Moreover, there are redundancies in the motifs by different filters. The number of filters dedicated to a motif may not be a reliable measure of the importance of the motif. The variation in the number of filters dedicated to a motif on the synthetic sequences, which do not have any class imbalance, suggests that the observed difference is more likely due to the difficulty of finding that motif and random initialization, not the importance of the motif.

The similar performance across CNNs explored here suggests that motif discovery does not require complicated architectures that learn distributed representations. Nevertheless, we posit that building *distributed* representations may be more beneficial in more complicated tasks, such as *in vivo* TF binding, because a wider array of representations can be combinatorically constructed from partial representations. Moreover, there becomes less dependence on convolutional filter lengths and numbers of filters as long as there exist deeper layers that can build representations hierarchically. In contrast, building *localist* representations enforces harder constraints set by the numbers of filters and the filter lengths, limiting the amount of representations and their sizes that can be learned. However, when the main features in the dataset is simple, such as whether or not a motif is present, then CNN architectures that learn localist representations achieve an easier to interpret model that still performs competitively.

Alternative approaches to visualize internal representations of a CNN can be achieved with attribution methods (Simonyan *et al*, 2013; Shrikumar *et al*, 2017; Springenberg *et al*, 2014; Smilkov *et al*, 2017; Lundberg and Lee, 2017). Attribution methods take into account distributed representations across the entire network, but they only consider one sequence at a time. Their scores can be noisy, leading to significant importance of nucleotide variants that are not necessarily biologically relevant. While attribution scores can be clustered to average down noise (Shrikumar *et al*, 2018), it is still unclear to what extent the attribution scores faithfully recapitulate features learned by the network (Adebayo *et al*, 2018; Kindermans *et al*, 2017; Nie *et al*, 2018), especially for genomic sequences. Further research is required to understand the factors that affect the fidelity of recovering biological signals from attribution scores.

Methods

Synthetic dataset

The synthetic dataset consists of sequences with known motifs embedded in random DNA sequences to mimic a typical multi-class binary classification task for ChIP-seq datasets. We acquired a pool of 24 PWMs from 12 unique transcription factors (forward and reverse complements) from the JASPAR database (Mathelier *et al*, 2016): Arid a, CEBPB, FOSL1, Gabpa, MEF2A, MAFK, MAX, MEF2A, NFYB, SP1, SRF, STAT1, and YY1. For each sequence, we generated a 200 nt random DNA sequence model with equal probability for each nucleotide. 1 to 5 TF PWMs were randomly chosen with replacement and randomly embedded along the sequence model such that each motif has a buffer of at least 1 nucleotide from other motifs and the ends of the sequence. We generated 25,000 sequence models and simulated a single synthetic sequence from each model. A corresponding label vector of length 12, one for each unique transcription factor, was generated for each sequence with a one representing the presence of a TF's motif or its reverse complement along the sequence model and zero otherwise. The 25,000 synthetic sequences and their associated labels were then randomly split into a training, validation, and test set according to the fractions 0.7, 0.1, and 0.2, respectively.

In vivo dataset

Sequences which contain ENCODE ChIP-seq peaks were downloaded from the DeepSEA dataset via (Zhou and Troyanskaya, 2015). The human reference genome (GRCh37/hg19) was segmented into non-overlapping 200 nt bins. A vector of binary labels for ChIP-seq peaks and DNase-seq peaks was created for each bin, with a 1 if more than half of the 200 nt bin overlaps with a peak region, and 0 otherwise. Adjacent 200 nt bins were then merged to 1,000 nt lengths and their corresponding labels were also merged. Chromosomes

8 and 9 were excluded from training to test chromatin feature prediction performances, and the rest of the autosomes were used for training and validation. We truncated the DeepSea dataset to include only the sequences which contain at least one of 12 transcription factor labels: Arid3a, CEBPB, FOSL1, Gabpa, MEF2A, MAFK, MAX, MEF2A, NFYB, SP1, SRF, STAT1, and YY1 (See Supplementary Table S1 for ENCODE filenames and class indices from the original DeepSea dataset). 270,382 (92%) sequences comprise the training set and 23,768 (8%) sequences comprise the test set. Each 1000 nt DNA sequence is one-hot encoded into a 4x1000 binary matrix, where rows correspond to A, C, G and T.

CNN Models

All CNNs take as input a 1-dimensional one-hot-encoded sequence with 4 channels (one for each nucleotide: A, C, G, T), then processes the sequence with two convolutional layers, a fully-connected hidden layer, and a fully-connected output layer with 12 output neurons that have sigmoid activations for binary predictions. Each convolutional layer consists of a 1D cross-correlation operation, which calculates a running sum between convolution filters and the inputs to the layer, followed by batch normalization (Ioffe and Szegedy, 2015), which independently scales the features learned by each convolution filter, and a non-linear activation with a rectified linear unit (ReLU), which replaces negative values with zero.

The first convolutional layer employs 30 filters each with a size of 19 and a stride of 1. The second convolutional layer employs 128 filters each with a size of 5 and a stride of 1. All convolutional layers incorporate zero-padding to achieve the same output length as the inputs. Each convolutional layer is followed by max-pooling with a window size and stride that are equal, unless otherwise stated. The product of the two max-pooling window sizes is equal to 100. Thus, if the first max-pooling layer has a window size of 2, then the second max-pooling window size is 50. This constraint ensures that the number of inputs to the fully-connected hidden layer is the same across all models. The fully-connected hidden layer employs 512 units with ReLU activations.

Dropout (Srivastava *et al*, 2014), a common regularization technique for neural networks, is applied during training after each convolutional layer, with a dropout probability set to 0.1 for convolutional layers and 0.5 for fully-connected hidden layers. During training, we also employed L_2-regularization with a strength equal to 1e-6. The parameters of each model were initialized according to (He *et al*, 2015), commonly known as He initialization.

All models were trained with mini-batch stochastic gradient descent (mini-batch size of 100 sequences) for 100 epochs, updating the parameters after each mini-batch with Adam updates (Kingma and Ba, 2014), using recommended default parameters with a constant learning rate of 0.0003. Training was performed on a NVIDIA GTX Titan X Pascal graphical processing unit with acceleration provided by cuDNN libraries (Chetlur *et al*, 2014). All reported performance metrics and saliency logos are drawn strictly from the test set using the model parameters which yielded the lowest binary cross-entropy loss on the validation set, a technique known as early stopping.

Visualizing saliency analysis

Saliency analysis is performed by calculating the gradients of a neuron-of-interest with respect to the input one-hot representation. We use a variant of saliency analysis, called guided-backpropagation (Springenberg *et al*, 2014), which rectifies negative gradients through each ReLU activation. To generate a saliency logo, we calculated the saliency map using guided-backpropagation from the logits of a given class to the inputs. We then normalized the saliency map by dividing the maximum absolute value across the saliency map. Next, we applied an exponential filter according to: $\hat{S} = \exp\left[\lambda \frac{S}{\max|S|}\right]$, where $\hat{S}$ is the normalized saliency map, S is the saliency map generated by guided-backprop, λ is a scaling factor that we set to 3 for all of saliency logos in this paper. We then separately normalized each position across by dividing the sum of the filtered saliency map across nucleotides, thereby providing a probability for each nucleotide at each position. To generate a sequence logo,the normalized saliency values of each nucleotide a at each position i is scaled according to: $\hat{S}_{a,i} \times I_i$, where $I_i = 2 + \sum_a \hat{S}_{a,i} \log_2 \hat{S}_{a,i}$.

Visualizing saliency analysis and 1st layer filters

A first layer filter was visualized by scanning it across every sequence in the test set. Sequences whose max activations was less than a cutoff of 70% of the maximum possible activation achievable for that filter were removed. A subsequence the size of the filter is taken about the max activation for each remaining sequence and assembled into an alignment. Subsequences that are shorter than the filter size, because their max activation is too close to the ends of the sequence, are also disregarded. A position probability matrix is created from the alignment and converted to a sequence logo.

Availability

Python scripts to download and process the datasets and TensorFlow code to build, train, and evaluate the CNNs can be found via `https://github.com/p-koo/learning_sequence_motifs`.

Acknowledgements

The authors thank Tim Dunn, Soohyun Cho, and Steffan Paul for helpful feedback on the manuscript.

References

Adebayo J, Gilmer J, Muelly M, Goodfellow I, Hardt M, Kim B (2018) Sanity checks for saliency maps. *Advances in Neural Information Processing Systems* : 9525–9536

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

Angermueller C, Parnamaa T, Parts L, Stegle O (2016) Deep learning for computational biology. *Molecular Systems Biology* **12**: 878

Chetlur S, Woolley C, Vandermersch P, Cohen J, Tran J, Catanzaro B, Shelhamer E (2014) cudnn: Efficient primitives for deep learning. *arXiv* **1410.0759**

Foat BC, Morozov AV, Bussemaker HJ (2006) Statistical mechanical modeling of genome-wide transcription factor occupancy data by MatrixREDUCE. *Bioinformatics* **22**: e141–e149

Ghandi M, Mohammad-Noori M, Ghareghani N, Lee D, Garraway L, Beer MA (2016) gkmSVM: an R package for gapped-kmer SVM. *Bioinformatics* **32**: 2205–2207

Gupta S, Stamatoyannopoulos JA, Bailey TL, Noble WS (2007) Quantifying similarity between motifs. *Genome Biology* **8**

He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*

Hinton G, McClelland J, Rumelhart D (1986) Distributed representations. *Parallel distributed processing Explorations in the microstructure of cognition* **1**: 77–109

Hiranuma N, Lundberg S, Lee S (2017) DeepATAC: A deep-learning method to predict regulatory factor binding activity from ATAC-seq signals. *bioRxiv* **172767**

Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv* **1502.03167**

Kelley DR, Snoek J, Rinn JL (2016) Basset: Learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Research* **26**: 990–999

Kindermans P, Hooker S, Adebayo J, Alber M, SchÃijtt K, DÃďhne S, Erhan D, Kim B (2017) The (un) reliability of saliency methods. *arXiv* **1711.00867**

Kingma D, Ba J (2014) Adam: A method for stochastic optimization. *arXiv* **1412.6980**

Lundberg S, Lee S (2017) A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems* **4765-4774**

Mathelier A, Fornes O, Arenillas DJ, Chen Cy, Denay G, Lee J, Shi W, Shyr C, Tan G, Worsley-Hunt R, *et al* (2016) JASPAR 2016: a major expansion and update of the open-access database of transcription factor binding profiles. *Nucleic Acids Research* **44**: D110–D115

Nie W, Zhang Y, Patel A (2018) A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. *arXiv* **1805.07039**

Quang D, Xie X (2016) DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences. *Nucleic Acids Research* **44**: 107

Shrikumar A, Greenside P, Kundaje A (2017) Learning important features through propagating activation differences. *In Proceedings of the 34th International Conference on Machine Learning* **70**: 3145–3153

Shrikumar A, Tian K, Shcherbina A, Avsec Z, Banerjee A, Sharmin M, Nair S, Kundaje A (2018) TF-MoDISco v0. 4.4. 2-alpha. *arXiv* **1811.00416**

Siggers T, Gordan R (2013) ProteinâĂŞDNA binding: complexities and multi-protein codes. *Nucleic Acids Research* **42**: 2099–2111

Simonyan K, Vedaldi A, Zisserman A (2013) Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv* **1312.6034**

Smilkov D, Thorat N, Kim B, Viegas F, Wattenberg M (2017) Smoothgrad: removing noise by adding noise. *arXiv* **1706.03825.**

Springenberg JT, Dosovitskiy A, Brox T, Riedmiller M (2014) Striving for simplicity: The all convolutional net. *arXiv* **1412.6806**

Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* **15**: 1929–1958

Zeng H, Edwards MD, Liu G, Gifford DK (2016) Convolutional neural network architectures for predicting DNA–protein binding. *Bioinformatics* **32**: i121–i127

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