frontiers in Genetics

ORIGINAL RESEARCH
published: 20 November 2019
doi: 10.3389/fgene.2019.01145

Deepprune: Learning Efficient and Interpretable Convolutional Networks Through Weight Pruning for Predicting DNA-Protein Binding

Xiao Luo[1†], *Weilai Chi*[2†] *and Minghua Deng*[1,2*]

[1] School of Mathematical Sciences, Peking University, Beijing, China, [2] Center for Quantitative Biology, Peking University, Beijing, China

OPEN ACCESS

Edited by:
Ping Ma,
University of Georgia,
United States

Reviewed by:
Lin Hou,
Tsinghua University,
China
Xiaoxiao Sun,
University of Arizona,
United States
Xin Xing,
Harvard University,
United States

***Correspondence:**
Minghua Deng
dengmh@math.pku.edu.cn

[†]These authors share first authorship

Specialty section:
This article was submitted to Bioinformatics and Computational Biology, a section of the journal Frontiers in Genetics

Received: 13 August 2019
Accepted: 21 October 2019
Published: 20 November 2019

Citation:
Luo X, Chi W and Deng M (2019) Deepprune: Learning Efficient and Interpretable Convolutional Networks Through Weight Pruning for Predicting DNA-Protein Binding. Front. Genet. 10:1145. doi: 10.3389/fgene.2019.01145

Convolutional neural network (CNN) based methods have outperformed conventional machine learning methods in predicting the binding preference of DNA-protein binding. Although studies in the past have shown that more convolutional kernels help to achieve better performance, visualization of the model can be obscured by the use of many kernels, resulting in overfitting and reduced interpretation because the number of motifs in true models is limited. Therefore, we aim to arrive at high performance, but with limited kernel numbers, in CNN-based models for motif inference. We herein present Deepprune, a novel deep learning framework, which prunes the weights in the dense layer and fine-tunes iteratively. These two steps enable the training of CNN-based models with limited kernel numbers, allowing easy interpretation of the learned model. We demonstrate that Deepprune significantly improves motif inference performance for the simulated datasets. Furthermore, we show that Deepprune outperforms the baseline with limited kernel numbers when inferring DNA-binding sites from ChIP-seq data.

Keywords: deep neural networks, motif inference, network pruning, convolutional neural networks, interpretation

BACKGROUND

Determining how proteins interact with DNA to regulate gene expression is essential for fully understanding many biological processes and disease states. Many DNA binding proteins have affinity for specific DNA binding sites. ChIP-seq combines chromatin immunoprecipitation (ChIP) with massively parallel DNA sequencing to identify DNA binding sites of DNA-associated proteins (Zhang et al., 2008). However, DNA sequences directly obtained by experiments typically contain noise and bias. Consequently, many computational methods have been developed to predict protein-DNA binding, including conventional statistical methods (Badis et al., 2009; Ghandi et al., 2016) and deep learning-based methods (Alipanahi et al., 2015; Zhou and Troyanskaya, 2015; Zeng et al., 2016). Convolutional neural networks (CNNs) have attracted attention for identifying protein-DNA binding motifs in many studies (Alipanahi et al., 2015; Zhou and Troyanskaya, 2015). Genomic sequences are first encoded in one-hot format; then, a 1-D convolution operation with four channels is performed on them. For conventional machine learning methods, the sequence specificities of a protein are often characterized by position

weight matrices (PWM) (Stormo, 2000). PWM has a direct connection to CNN-based model since the log-likelihood of the resulting PWM of each DNA sequence is exactly the sum of a constant and the convolution of the original kernel on the same sequence from the view of probability model (Ding et al., 2018). Zeng et al. (2016) experimented with different structures and hyperparameters and showed that the convolutional layers with more kernels could obtain better performance. They also showed that training models with gradient descent methods is sensitive to weight initialization, showing, in turn, that training could be obstructed at local optimum of loss function. However, the use of too many kernels could introduce too much noise and, thus, overfitting, leading to misinterpretation of the model. By visualizing the recovery of the underlying motifs in the models, we found that only the several best-recovered motifs, in the sense of information content, could be equated to the true motifs, demonstrating that most kernels only act during the process of training by increasing generalization ability in order to overcome the local optimum problem (Du et al., 2018). Such kernels can be termed auxiliary kernels, and these kernels produce noise and reduce performance at the end of training. Neural networks with circular filters (Blum and Kollmann, 2019) can address this problem, but performance was only found to significantly improve in the one-kernel CNN-based model. However, since some proteins likely bind multiple motifs in the DNA sequence in omics data, the one-kernel CNN-based model cannot meet the needs of motif finding. Moreover, its overall performance is lower than expected when kernel number is limited (e.g., 16). Luo et al, (2019) replaced global max pooling with expectation pooling, which is shown to increase the robustness for kernel numbers. However, expectation pooling only increases model robustness; it does not limit kernel numbers.

In contrast, neural network pruning can reduce kernel numbers and by doing so, improve inferential performance without harming accuracy in the field of computer vision (Han et al., 2015a; Abbasi-Asl and Yu, 2017; Frankle and Carbin, 2018). Pruning methods can be classified into structured and unstructured. The former refers to pruning at the level of channels, or even layers, for which the original network structure is still preserved (Hu et al., 2016; Li et al., 2016; Changpinyo et al., 2017; He et al., 2017). The latter includes individual weight pruning. Han et al. (Han et al., 2015b) developed a method whereby network weights of small magnitude were pruned, and it was very successful in highly compressed neural network models (Han et al., 2015a). Unstructured pruning can ensure that models will achieve sparse weight matrices which result in compression and acceleration with dedicated hardware (Han et al., 2016).

With evidence that models with only a few kernels can fit the PWM model very well, we propose a novel model, termed Deepprune, which utilizes pruning techniques in motif inference. Several assumptions underlie the design of Deepprune. First, by its stronger representation and optimization power, we believed that starting with training a large and over-parameterized network could provide a model with high performance. Second, for the PWM model, which often characterizes sequence specificities, several kernels which are viewed as motif detectors are enough for motif inference. Third, the inclusion of too many auxiliary kernels leads to misinterpretation of the model. Fourth, auxiliary kernels may produce noise and lower performance at the end of training. If the PWM model characterizes sequence specificities and if no interaction among different motifs is considered, then Deepprune achieves better performance with fewer kernels, markedly exceeding baseline in simulated datasets. In spite of the uncertainty of the true model, Deepprune still arrives at better performance with the same kernel numbers in real datasets, which shows the superiority of our model. Our model can also find more accurate motifs by model visualization and eliminate auxiliary kernels. All coding utilized to implement Deepprune and all the figure reproductions in the paper is available at https://github.com/klovbe/Deepprune.

METHODS

Detecting Sequence Motifs With Convolutional Neural Network

We adopt the simplest model in DeepBind as our basic neural network architecture (Alipanahi et al., 2015). The sequences are represented as numerical vectors. Each of the four nucleotides is denoted as one of the four one-hot vectors [1,0,0,0], [0,1,0,0], [0,0,1,0], and [0,0,0,1]. Consequently, a sequence $X = X_1,\cdots, X_L$ is transformed into a $4 \times L$matrix S. We first add a 1-D convolutional layer with rectified linear units (ReLU) activation serving as a motif scanner layer (Radford et al., 2015), followed by a global max pooling layer. Then we add a mask layer to prune the weights according to some given criterion, which will be introduced in the next section. The last layer is a fully connected layer with sigmoid activation the output of which is the probability of a sample being positive (**Figure 1**).

Formally, if the convolutional kernels are denoted by $4 \times L_F$ matrices F^1, F^2, $\cdots$,F^d, in which L_F is the length of the kernel, we have

$$h(pk) = \sum_{i=1}^{L_F}\sum_{j=1}^{4} S_j,(i+p-)F_{ji}^k, k = 1.\ldots d, (Convolution)$$

$$a_{pk} = \max(0, h_{pk})(Activation)$$

$$z_k = \max\{a_{1,k},\ldots,a_{L-L_F+1,k}\}(Globalmax - pooling)$$

$$u_k = z_k \cdot m_k(MaskLayer)$$

$$p(C_{motif} \mid S) = \sigma\left(b + \sum_{k=1}^{d} w_k u_k\right)(DenseLayer),$$

where w_k and w are weights, b is bias and $\sigma(x)$ denotes the sigmoid function for classification. Compared to basic neural network architectures, note that a mask layer is added because we want to mask the kernels that have little impact on the performance at the end of training. As a result, m_k is set as 0 or 1, and $m_k = 0$ means that the information of the k-th kernel cannot pass through this

layer. Because the calculation of each kernel is independent in the convolutional layer, the pruned model can be viewed as a CNN-based model with fewer kernels. Accordingly, we can prune our network to get an efficient and interpretable architecture with limited kernels.

Deepprune

In this work, we take iterative pruning on the weights of the dense layer in the CNN-based model and drop the learning rate of each pruning step gradually for fine-tuning. First, we utilize $2^k \times d$ convolutional kernels in our model, i.e., the large, over-parameterized model. Half the number of kernels is pruned each time, according to a certain criterion. In other words, the number of values being 1 in the mask layer is halved each time. Since weight pruning may lead to decreased performance, we then fine-tune the pruned model to regain the lost performance. The above two steps are iterated for k times and then the final model is obtained. Deepprune first gives the weights in the architecture an appropriate area from the global view and adjusts the weights gradually by iterative pruning and fine-tuning. In this way, we can overcome the drawback of easily stopping at the local optima restricted by the local views in the original model with limited kernel numbers by the strong ability of representation in our model.

Three criteria are designed for Deepprune. For weight-based Deepprune, we consider the weight of scores (i.e., w_k) in the dense layer. The weights with small magnitude are pruned as

$$m_k = \begin{cases} 1 & |w_k| > median(|w_k|) \\ 0 & otherwise \end{cases}$$

in which the median operation takes the median of $|w_k|$ corresponding to unpruned weights. However, the scale problem below is not considered in the first criterion. We know that $b + \sum_{k=1}^{d} w_k u_k$ is the input for the sigmoid activation layer which predicts the label; that is to say $w_k u_k$ determines the importance of the k-th kernel. However, the score of the k-th kernel can be multiplied by m if weights in the convolutional layer are multiplied by m, and then the weight corresponding to this kernel in the dense layer will shrink by training. As a result, the score u_k obtained in the mask layer also counts, and the impact of the score over samples needs to be considered. For the score-based criterion, the scores with small difference between positive and negative samples are pruned.

$$m_k = \begin{cases} 1 & |AVG_P u_k - AVG_N u_k| > \\ & median(|AVG_P u_k - AVG_N u_k|) \\ 0 & otherwise \end{cases}$$

in which $AVG_P u_k$ means the average score over positive samples, and $AVG_N u_k$ means the average score over negative samples. For the score-and-weight-based criterion, we directly consider $AVG_P u_k$, which determines the input for the sigmoid activation layer as

$$m_k = \begin{cases} 1 & |AVG_P u_k * w_k - AVG_N u_k * w_k| > \\ & median(|AVG_P u_k * w_k - AVG_N u_k * w_k|) \\ 0 & otherwise \end{cases}$$

Implementation of the Models

The hyperparameters to train the simulated datasets contain the length of convolutional kernels, learning rate, times of pruning k, last pruned kernel number d, number of epochs, training batch size, learning rate decay schedule, and the optimizer. First, we train the basic model with $2^k \times d$ kernel numbers, and we get Deepprune models with $2^{k-1} \times d, \cdots, d$ kernel numbers. We also consider the strength of fine-tuning and denote the pruned model without fine-tuning from the last pruned model (twice the kernel numbers) as Deepprune-inter. To make a comparison, we match our model with baseline, which is the basic model utilizing identical kernel number trained directly without pruning.

For training, we used cross-entropy as a loss function without any weight decay (i.e., L_2 regularization over the weights), and trained the model utilizing the standard backpropagation algorithm and the Adam optimizer (Kingma and Ba, 2014). The area under the ROC (AUC) (Fawcett, 2004; Davis and Goadrich, 2006) is utilized to assess prediction performance. We took standard early stopping strategy to avoid overfitting (i.e., the training will be stopped as long as the loss over the validation set has stopped decreasing during continuous 15 epochs).

Our model is implemented with Keras for Python (Cholle, 2015).

Datasets

Simulated Datasets

For simulation, TRANSFAC database was utilized to evaluate the performance of Deepprune (Wingender et al., 1996). Each simulated data set includes both negative and positive samples, or sequences. Each negative sample consists of independent and identically distributed nucleotides obeying a multinomial distribution with the probability of 0.25 for each {A, C, T, G}. Each positive sample was built in the same manner as a negative sample except that sequences from certain motifs were inserted at some locations randomly. The sequences inserted in the positive samples for the five simulated data sets were listed below:

- **Simulated dataset 1**, **2**, **3:** Each sequence was generated from either the first or the second motif. We chose motif for each positive sample randomly with equal probability.
- **Simulated dataset 4:** Each sequence was generated from one of the four given motifs; other rule is the same.
- **Simulated dataset 5:** Each sequence was generated from one of the eight given motifs; other rule is the same.

The number of sequences in the training dataset and test dataset is equal. We emphasized because a given protein may

bind to multiple motifs in the DNA sequence, our simulation datasets were constructed reasonably.

Real Datasets

690 ChIP-seq ENCODE datasets utilized by DeepBind were chosen to be real datasets (Alipanahi et al., 2015). Each dataset corresponds to a specific DNA-binding protein. Its positive samples are 101 bp DNA sequences confirmed to bind to a given protein experimentally while its negative samples were constructed through shuffling dinucleotides in the positive sequences. All the datasets are available at http://cnn.csail.mit.edu/.

RESULTS

Deepprune Performs Better Than the Baseline on the Simulated Data

In this section, we use the simulated data to compare Deepprune with baseline. Baseline is the simplest CNN model with no hidden layers, in other words the architecture of Deepprune, but without the mask layer, with batch size = 256, $d = 4$, and $k = 6$. All the models in this paper are pruned from the basic model with kernels (**Table 1**). We chose $d = 4$ for 101 bp sequences, which can be divided into about 4 parts of 24 bp. If $d < 4$, the kernel number may be less than the number of the underlying motifs. Also, simulated dataset 5 can show how Deepprune performs when the kernel number is half the number of the underlying motifs. Several random seeds are set to evaluate the robustness of the models' performance for the simulated datasets. Our baseline is directly training the neural network without pruning, which is the simplest model raised in DeepBind. Only weight-based Deepprune is considered in this section.

TABLE 1 | Parameter settings for the simulated datasets.

Name	Values
Batch size	256
Kernel length	24
Optimizer	Adam with initial learning rate 0.01
Learning rate decay schedule	Drops the learning rate by 1.2 every pruning step
Random seed	0, 1, 2, 3, 4, 5, 6, 7, 8, 100, 123, 1,000, 1234, 10,000, 12,345, 100,000, 123,456, 1,000,000

Compared to the baseline model without pruning, we found that Deepprune improved motif inference performance on first three simulated datasets from **Figure 2**. Specifically, as kernel number increases, the performance of baseline has a tendency to improve, which is consistent with Zeng et al. (2016). However, as kernel number decreases, the performance of Deepprune shows a converse tendency such that the mean of AUC of Deepprune shows significant improvement as the iteration continues. What's more, variances of AUC of Deepprune are also more robust. When compared with models with the same kernel number, Deepprune shows its wonderful ability to limit kernels for accuracy and robustness, showing that Deepprune works effectively for motif inference.

Compared with the baseline, performance improvement was notably evident on the simulated dataset 4 and 5 with a hard true model, reflecting the excellence of Deepprune in cases with the complex motif settings (**Figure 3**). Distinctly, the performance of baseline with four kernels is close to that of random guess on the complex datasets. This result shows that the baseline

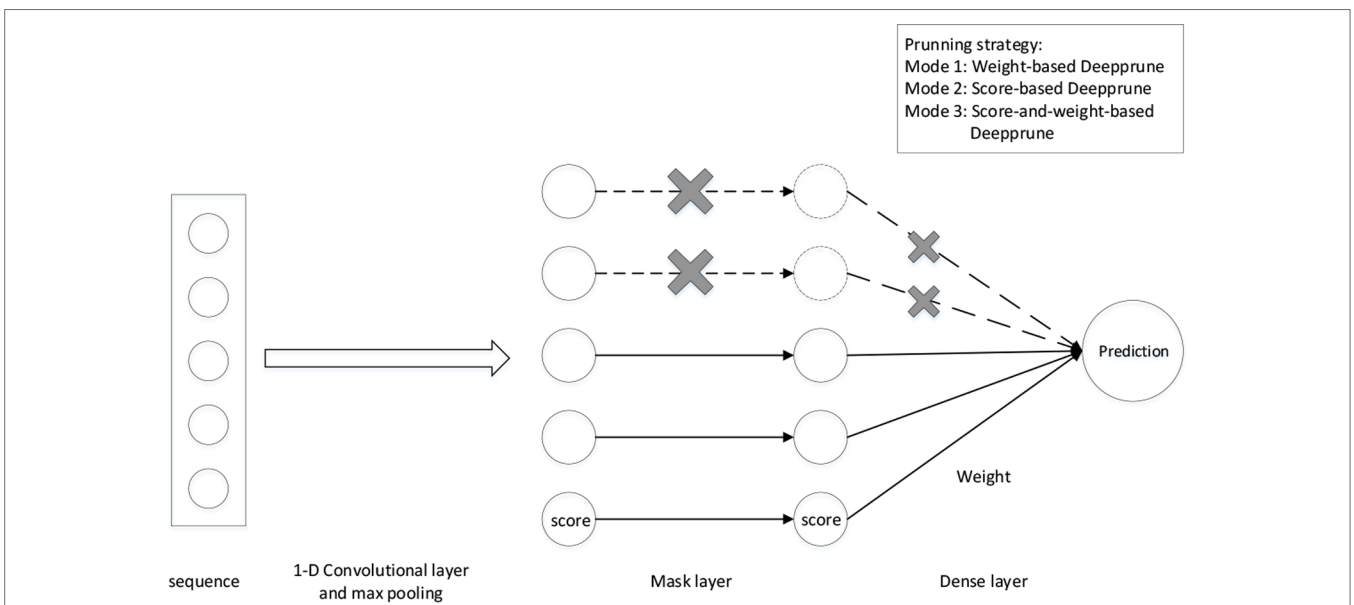


FIGURE 1 | The architecture of Deepprune. The first layer is a convolutional layer. The second layer is a rectified linear units activation function followed by global max pooling. A mask layer is added to prune the small-magnitude weights. The fourth layer is a dense layer which linearly combines the outputs of all the kernels. The last layer is a sigmoid activation function which converts the values obtained in the dense layer to a value between 0 and 1 which corresponds to a probability. Three pruning strategies are designed for Deepprune with different modes.

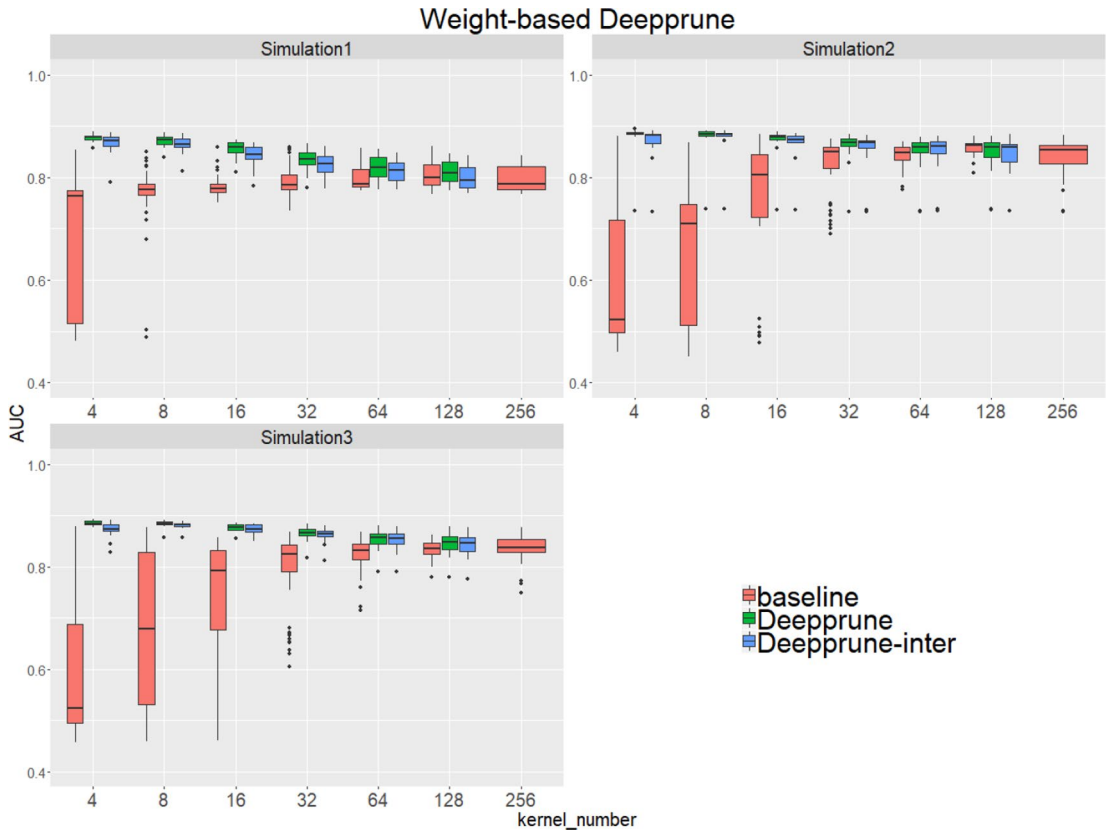


FIGURE 2 | Weight-based Deepprune performs much better and is much robuster to different random initialization than baseline when kernel number is limited in the first three simulated datasets. The x axis shows the kernel numbers utilized in the model, and the y axis shows the area under the receiver operating characteristic curves (AUCs) obtained in testing. As kernel number decreases, the performance of Deepprune shows a converse tendency compared to baseline; thus, as iteration continues in an upward gradient, the mean AUC of Deepprune significantly improves.

model with limited kernel numbers does not satisfy the need for overcoming the local optimum problem and that it lacks robustness to initialization. To our surprise, when the kernel number is half that of the motif number, the performance of Deepprune only drops a little, showing that the condition d = 4 is enough. What's more, fewer kernels helps to improve the interpretation of our model. We also find that Deepprune-inter always shows poorer results, no matter whether from the mean AUC or the variation of AUC, which demonstrates that fine-tuning is essential in Deepprune.

Comparison of Three Pruning Criteria

Next, we studied the effects of the three criteria on the performance of Deepprune, as noted previously. We selected three simulated datasets to determine the difference of three different rules. If the scores are considered when pruning, then all samples in the training set need to be calculated, which leads to substantial calculation.

From **Figure 4**, when kernel number is high (e.g., 8 and 16), the performance of the three methods is nearly identical. Thus, the choice of three pruning methods is not crucial because the restriction to the kernel number is loose. However, when the kernel number is extremely limited, weight-based Deepprune shows its superiority compared to the other two methods in simulated dataset 1, in which the samples are hard to classify because of the information entropy in the true model. It is likely that weight-based Deepprune does not depend on samples which may cause randomness. From the case study below, the weights in the dense layer have a close magnitude, indicating that the scaling problem of scoring is difficult to solve in the smooth training process. Based on this observation, we select weight-based Deepprune as default.

Performance on Real Datasets

We test the performance of DeepPrune on read data analysis in this section. CNN parameters are set the same as those for the simulated datasets, except the kernel length was changed to 15.

When the number of kernels is limited (i.e., four), Deepprune achieves a statistically significant improvement in AUC from one-sided Wilcoxon signed-rank test in **Figure 5**, $p=1.02\times10^{-58}$, with a better performance on 77.10% of the datasets [**Table 2**]. We also selected two representative real datasets to show the superiority of Deepprune (**Figure S5** and **S6**). Nevertheless, its accuracy is lower on 22.90% of the

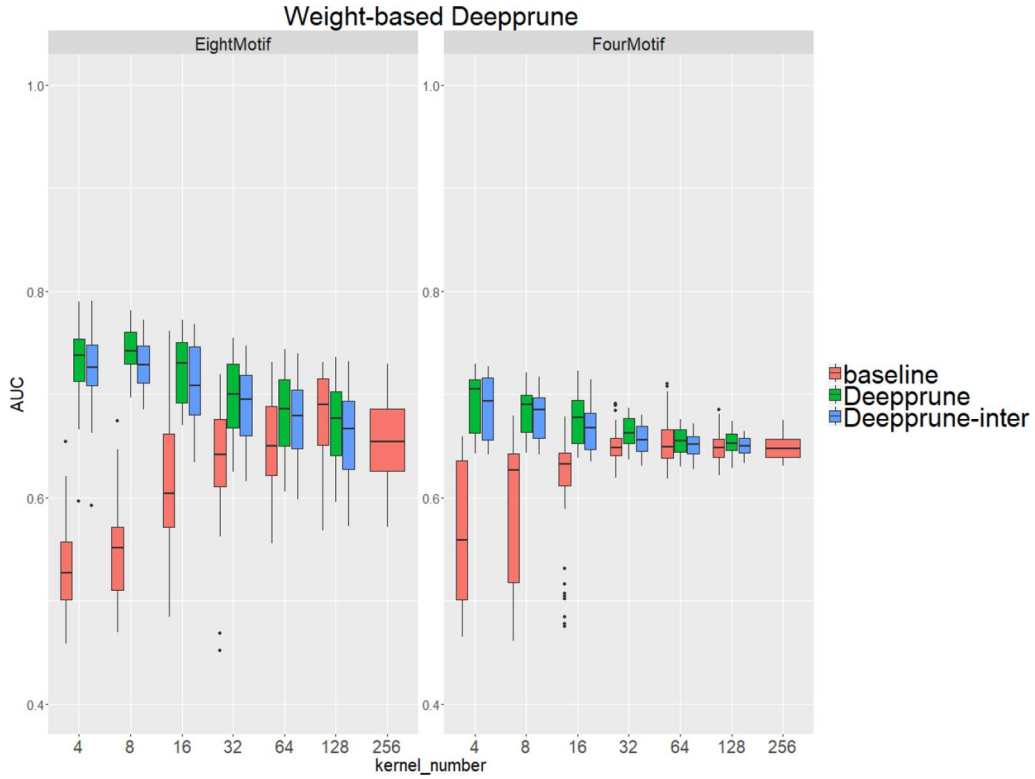


FIGURE 3 | Weight-based Deepprune performs much better and is much robuster to different random initialization than baseline when kernel number is limited in the last two complex simulated datasets, even when kernel number is half the motif number at which time the performance of Deepprune only drops slightly.

datasets, which does not match our expectation. This may be due to the non-convexity of the neural network model where local optimum is obtained. So we select the datasets for which our model's performance is lower, and we initialize the training with several different random seeds. In some of the selected datasets, the mean performance of DeepPrune is almost as good as the baseline (**Figure S2**). However, a consistent gap still appears in a small number of datasets in which the baseline shows better performance than our method, suggesting that the interaction of motifs is not considered in our architecture. It follows that the proposed architecture cannot represent the true model for some proteins in motif inference, which, therefore, creates bias for Deepprune. The fact that the performance of Deepprune gets worse when the number of kernels increases in simulations while the trend is opposite in real data also shows the above point. What's more, Deepprune outperforms the network with circular filters (Blum and Kollmann, 2019) significantly (**Figure S4**).

Case Study

We selected several kernels to track the change of their corresponding weights at different pruning stages in the dense layer. In this section, we utilized simulated dataset 3 for we only knew the true models in simulated datasets. We chose the weights of four unpruned kernels and two pruned kernels at the end of each fine-tuning step. All the weights were collected after fine-tuning. It should be noted that the weights of the kernels in the convolutional layer changed during fine-tuning.

From **Table 3**, we can see that the magnitude of weights is gained step-by-step for four unpruned kernels, indicating that kernels show their importance over a gradual upward gradient. Before pruning, the weights of unpruned kernels are scrapped by auxiliary kernels. After pruning auxiliary kernels, the weights of unpruned kernels aren't affected any more, which shows the superiority of Deepprune.

Model Visualization

Now we study the ability of Deepprune to recover the underlying motifs more accurately. As in the last section, we utilized simulated dataset 3 because we only knew the true motifs in simulated datasets. Two models are both trained with the same parameters in **Table 1** and the kernel number is set as 4. The sequence logos are generated from kernels the way introduced in Section Sequence logos of the DeepBind (Alipanahi et al., 2015) **Supplementary Materials**. The two best-recovered motifs, from the perspective of information content, were compared to the true motifs utilized on the simulated data. Their similarity (E-value) were also calculated utilizing the Tomtom algorithm (Gupta et al., 2007).

In **Figure 5** the motifs recovered by Deepprune and the baseline were both aligned to the true motifs. We clearly found the sequence logos generated by Deepprune were

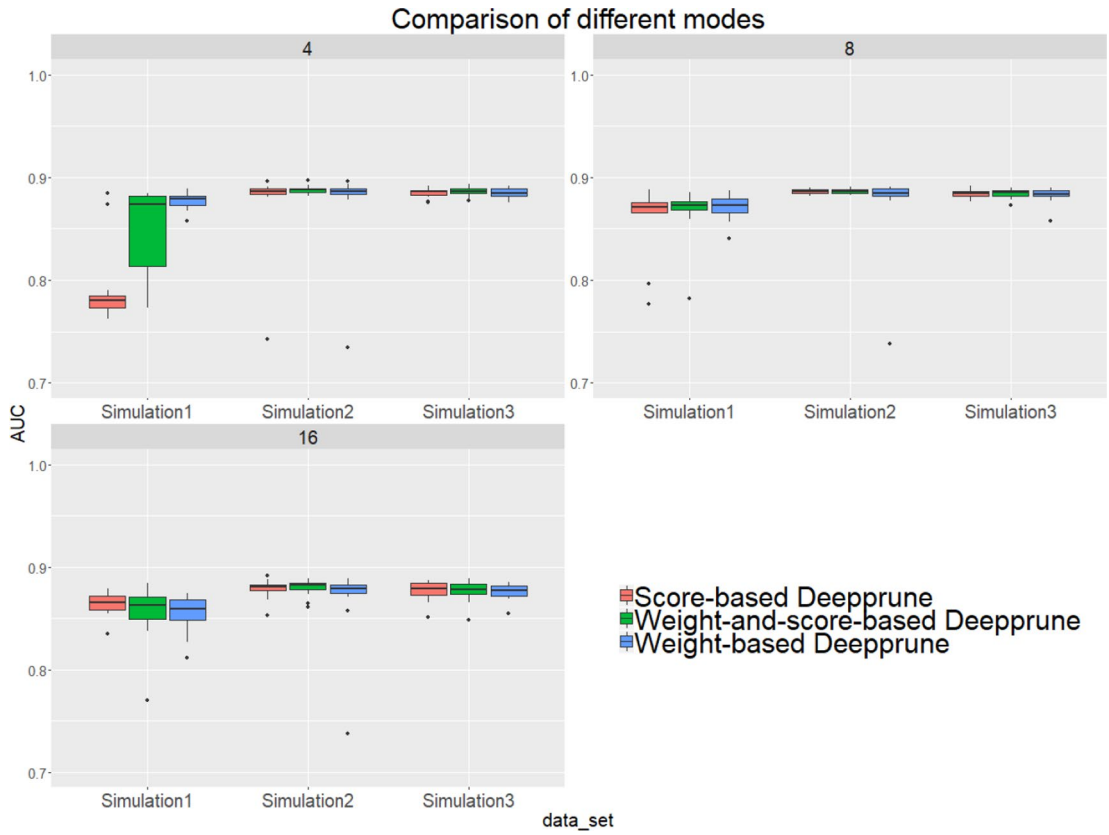


FIGURE 4 | Three models of Deepprune are compared on the simulated datasets. We show the performance of Deepprune based on different criteria with 4, 8, and 16 kernels. The performance of three different criteria is almost identical to that with 8 and 16 kernels during iterative pruning. However, the final model with four kernels shows that weighted-based Deepprune is superior to the other two methods in simulated dataset 1, but hard to classify owing to high entropy.

informative and accurate from the E-value. The base-recovered motif by the baseline with four kernels exhibited very bad performance and the short motif in simulated dataset 3 could not be matched by four filters. In addition, we found that the motif regions could be distinguished from other regions which clearly obey background distribution. Although the length of kernels is far beyond that of the true motifs, the extra positions, which are not aligned to the true motifs, do not contain any noise, owing to the ability of Deepprune to lessen the impact of auxiliary kernels at the end of training. We further explored the case of eight kernels and found a consist pattern (**Figure S3**).

TABLE 2 | Performance of Deepprune on real data.

Kernel number	Method	AUC	Percentage improved	P-value
	Circular filters	0.6923		
	Deepprune	0.8016	0.9826	$7.22 \times 10^{(-112)}$
	Baseline	0.7785		
4	Deepprune	0.8016	0.7710	$1.02 \times 10^{(-58)}$
	Baseline	0.8169		
8	Deepprune	0.8288	0.7174	$6.44 \times 10^{(-38)}$
	Baseline	0.8432		
16	Deepprune	0.8476	0.6826	$2.63 \times 10^{(-21)}$
	Baseline	0.8602		
32	Deepprune	0.8625	0.6681	$3.25 \times 10^{(-15)}$
	Baseline	0.8728		
64	Deepprune	0.8743	0.6507	$1.20 \times 10^{(-15)}$
	Baseline	0.8809		
128	Deepprune	0.8820	0.6986	$4.41 \times 10^{(-26)}$
256		0.8849		

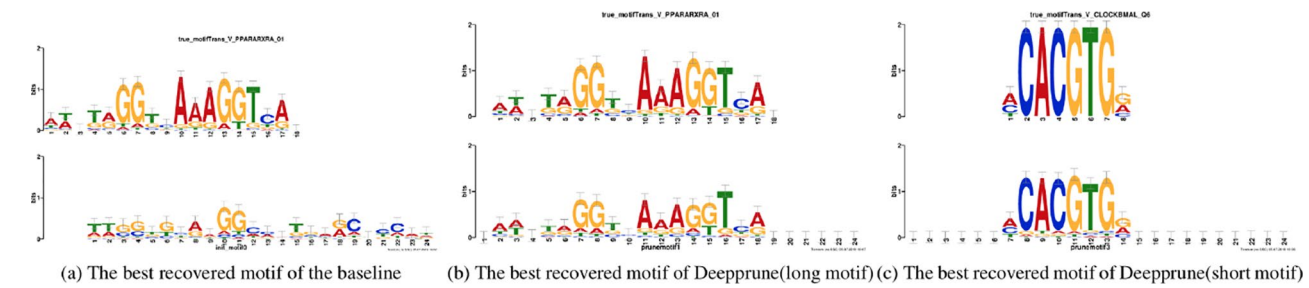
DISCUSSION

Regularization Behind Deepprune

L_0, L_1, and L_2 regularizations are three significant shrinkage methods for variable selection, and they are widely utilized in deep learning (He et al., 2016; Liu et al., 2017; Luo et al., 2019). However, the architecture of deep learning is multilayered and complex. Thus, for the same result, all weights in the architecture have the same infinite solution, e.g., the scaling problem noted before. L_1 and L_2 regularization update the original loss function by adding differentiable regularization terms, while L_0 regularization needs to be realized by pruning. Actually, Deepprune adds L_0 regularization to the weight in the

TABLE 3 | Absolute value of weights of several kernels during different pruning stages in the dense layer.

Kernel number	Kernel 1	Kernel 2	Kernel 3	Kernel 4	Kernel 5	Kernel 6
256	0.9150	0.9019	0.7043	0.8112	0.2192	0.3582
128	0.9462	0.9294	0.7322	0.8344	0.2625	0.4015
64	0.9548	0.9364	0.7305	0.8252	0.2100	0.3750
32	0.9616	0.9403	0.7540	0.8387	0.0000	0.4433
16	0.9836	0.9504	0.8153	0.8589	0.0000	0.4584
8	1.0919	1.0501	0.9620	1.0249	0.0000	0.0000
4	1.2832	1.1979	1.2065	1.2542	0.0000	0.0000



(a) The best recovered motif of the baseline (b) The best recovered motif of Deepprune(long motif) (c) The best recovered motif of Deepprune(short motif)

FIGURE 5 | Motifs recovered by our model (the last two) and by baseline (the first) aligned to the true given motifs (top row). Utilizing Tomtom algorithm, the E-values of the motifs recovered by Deepprune are 1.35×10^{-23}, 2.27×10^{-7}, respectively. At the same time, the ones recovered by baseline are 1.64×10^{-2}.

dense layer instead of the entire architecture. Iterative pruning can help avoid wrong pruning for the greedy algorithm compared with one-shot pruning, thus showing its superiority in many tasks (Frankle and Carbin, 2018). Although L_1 and L_2 penalties have been added to our model, the result shows little difference.

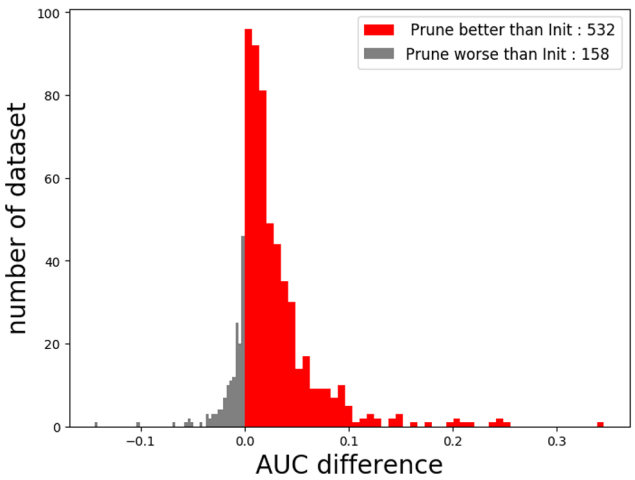


FIGURE 6 | The performance of Deepprune with four kernels on real datasets where kernel length = 15. Deepprune greatly increases the AUC for real datasets. The AUC difference under the baseline (Init) and Deepprune (Prune) is shown from the x axis. Deepprune is better than baseline on 532 datasets, but worse than baseline with 158 datasets. This figure clearly shows that Deepprune achieves better performance with limited kernel number.

Deep Models Are Necessary for Modeling Transcription Factor -DNA Specificities

Blum and Kollmann (Blum and Kollman, 2019) supposed that deep models may be unnecessary for modeling transcription factor-DNA specificities because they think that biological sequences are not composed of complex hierarchies of patterns as those in images. Deepprune can improve the performance of motif inference on real-world data compared with baseline, even with the same kernel number. However, since the weights are pruned iteratively, the performance of Deepprune does not change as what we saw in the simulated datasets. If PWM characterizes the specificities of motif inference and motif relationships are the same with those in simulated datasets, we will most likely see consistent performance in real-world and simulated datasets. In actuality, however, about 23% of datasets have a decrease compared to baseline with four kernels. As a result, we suspect that the interaction of different motifs and other complex relationships corresponding to motif inference need to be considered. Actually we suggest using different architectures to model different protein-binding problems. It is clear that adding the hidden layer gives deep learning architectures the ability to represent the interaction of different motifs and sequences of recurrent neural network models from the viewpoint of natural language processes, allowing various representations with different parameters. However, based on the results of our experiment, many biological sequences cannot be modeled very well by the simple DeepBind model, making it necessary to create deeper architectures to identify the underlying model for some proteins.

Lottery Ticket Hypothesis

Recently, the lottery ticket hypothesis has attracted attention in the field of deep learning. This hypothesis holds that dense, randomly initialized networks contain subnetworks that, when trained in isolation (i.e., utilizing the same initialization), reach test accuracy comparable to the original network in a similar number of iterations (Frankle and Carbin, 2018). The subnetworks are called winning tickets. Liu et al. (2018) even suggested that the value of automatically structured pruning algorithms sometimes lies in identifying efficient structures and performing implicit architecture search, rather than selecting "important" weights, irrespective of the initiation. First, if the architecture of our pruned model is equal to that of baseline, our result in the simulated dataset shows that either weight or initiation also counts for the performance of unstructured pruning algorithms. Second, we tried to find our winning tickets by following the methods in the original paper. We substituted the fine-tuning step in Deepprune for retraining, which resets the remaining parameters to their values before initial training. Experiments on real data with winning tickets realize slightly better performance (mean AUC is 0.8035 with four kernels), which shows that this hypothesis may be true for Deepprune (**Figure S1**).

CONCLUSION

In this study, we proposed a novel deep-learning framework called Deepprune, to improve the performance of predicting the binding preference of DNA-protein binding. Deepprune prunes weights of kernels in the dense layer and fine-tunes iteratively by adding a mask layer in the architecture of motif inference. Deepprune utilizes limited kernel number in the convolutional layer, which shows the efficiency and interpretability of our model. In this study, Deepprune is shown to improve model performance compared with baseline with the same limited kernel number, both in simulated and real-world datasets. Our method improves performance without changing the basic architectures or adding extra parameters at the end of training (**Figure 6**).

To the best of our knowledge, we are the first to introduce a pruning framework in the field of motif inference. Network pruning has been widely applied in the framework of deep learning for its ability to reduce storage and computation without affecting accuracy. Although the architecture of motif inference is very simple, network pruning is meaningful for the model since the use of fewer kernels can still achieve better interpretation as can be seen from model visualization.

The motif-finding problem remains unsolved. Deep learning is very useful for complex structures and large datasets. What's more, it has greatly improved the state-of-the-art in many areas. Neural networks have achieved a lot of success, such as DeepBind and DeepSEA for motif-finding. However, in spite of the great achievements, deep learning is blamed due to the lack of interpretability as well (Castelvecchi, 2016; Zou et al., 2019). DeepBind shows its superiority compared with conventional machine learning methods, which proves that both deep and complex representation of the sequences is essential for motif inference. Because the gap between the performance on simulated and real-world datasets, we wonder if this is due to the underlying model behind some of the real-world datasets is complex.

Recently, many studies have investigated the interpretation of neural networks and the underlying model behind real-world datasets. They utilize complex models, such as recurrent neural network (RNN) and the model with attention mechanism, which comes from the field of natural language processing, to represent the complex information of biological sequences (Pan and Yan, 2017; Pan et al., 2018; Pan and Shen, 2018; Shen et al., 2018; Zuallaert et al., 2018; Li et al., 2019; Luo et al., 2019). Actually, from the diversity of DNA-protein binding, we suggest using different architectures to model motif inference for specific proteins. Complex network architectures combined with pruning technology can result in approximating the true model of motif inference. Since our basic architecture is simple, adding a hidden layer before the dense layer and then adding an RNN layer after the convolutional layer, as well as replacing global max pooling with expectation pooling, can also be considered, but these topics are outside the scope of the present paper.

DATA AVAILABILITY STATEMENT

All code is public and can be found at https://github.com/klovbe/Deepprune.

AUTHOR CONTRIBUTIONS

XL, WC, and MD designed the experiments. XL drafted the manuscript. WC carried out the experiments. XL and WC analyzed the results. All authors read and approved the final manuscript.

FUNDING

This study was supported by the National Key Basic Research Project of China (No. 2015CB910303), The National Key Research and Development Program of China (No.2016YFA0502303), and the National Natural Science Foundation of China (No.31871342).

ACKNOWLEDGMENTS

This manuscript has been released as a Pre-Print at https://www.biorxiv.org/content/10.1101/729566v1

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fgene.2019.01145/full#supplementary-material

REFERENCES

Abbasi-Asl, R., and Yu, B. Structural compression of convolutional neural networks based on greedy filter pruning. [Preprint]. Available at: https://arxiv.org/pdf/1705.07356.pdf

Alipanahi, B., Delong, A., Weirauch, M. T., and Frey, B. J. (2015). Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nat. Biotechnol.* 33, 831. doi: 10.1038/nbt.3300

Badis, G., Berger, M. F., Philippakis, A. A., Talukder, S., Gehrke, A. R., and Jaeger, S. A. (2009). Diversity and complexity in dna recognition by transcription factors. *Sci.* 324, 1720–1723. doi: 10.1126/science.1162327

Blum, C. F., and Kollmann, M. (2019). Neural networks with circular filters enable data efficient inference of sequence motifs. *Bioinf.* 35, 3937–3943. doi: 10.1093/bioinformatics/btz194

Castelvecchi, D. (2016). Can we open the black box of ai? *Nat. News* 538, 20. doi: 10.1038/538020a

Changpinyo, S., Sandler, M., and Zhmoginov, A. (2017). The power of sparsity in convolutional neural networks. [Preprint]. Available at: https://arxiv.org/pdf/1702.06257.pdf

Chollet, F. (2015). Keras.

Davis, J., and Goadrich, M. (2006). The relationship between precision-recall and roc curves in *Proceedings of the 23 international conference on Machine learning (ACM,* 2016*)*, 233–240. doi: 10.1145/1143844.1143874

Ding, Y., Li, J.-Y., Wang, M., Tu, X., and Gao, G. (2019). An exact transformation of convolutional kernels enables accurate identification of sequence motifs. An exact transformation for CNN kernel enables accurate sequence motif identification and leads to a potentially full probabilistic interpretation of CNN. *bioRxiv* 163220. doi: 10.1101/163220

Du, S. S., Zhai, X., Poczos, B., and Singh, A. (2018). Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:*1810.02054.

Fawcett, T. (2004). Roc graphs: Notes and practical considerations for researchers. *Mach. Learn.* 31, 1–38.

Frankle, J., and Carbin, M. (2018). The lottery ticket hypothesis: finding sparse, trainable neural networks. [Preprint]. Available at: https://arxiv.org/pdf/1803.03635.pdf

Ghandi, M., Mohammad-Noori, M., Ghareghani, N., Lee, D., Garraway, L., and Beer, M. A. (2016). gkmsvm: an r package for gapped-kmer svm. *Bioinf.* 32, 2205–2207. doi: 10.1093/bioinformatics/btw203

Gupta, S., Stamatoyannopoulos, J. A., Bailey, T. L., and Noble, W. S. (2007). Quantifying similarity between motifs. *Genome Biol.* 8, R24. doi: 10.1186/gb-2007-8-2-r24

Han, S., Mao, H., and Dally, W. J. (2015a). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. [Preprint]. Available at: https://arxiv.org/pdf/1510.00149.pdf

Han, S., Pool, J., Tran, J., and Dally, W. (2015b). Learning both weights and connections for efficient neural network, in *Advances in neural information processing systems*, 1135–1143.

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., and Horowitz, M. A. (2016). Eie: efficient inference engine on compressed deep neural network, in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA) (IEEE)*, 243–254. doi: 10.1109/ISCA.2016.30

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778. doi: 10.1109/CVPR.2016.90

He, Y., Zhang, X., and Sun, J. (2017). Channel pruning for accelerating very deep neural networks In *Proceedings of the IEEE International Conference on Computer Vision*, 1389–1397. doi: 10.1109/ICCV.2017.155

Hu, H., Peng, R., Tai, Y.-W., and Tang, C.-K. (2016). Network trimming: a data-driven neuron pruning approach towards efficient deep architectures. [Preprint]. Available at: https://arxiv.org/pdf/1607.03250.pdf

Kingma, D. P., and Ba, J.Kingma, D. P., and Ba, J. (2014). Adam: a method for stochastic optimization. [Preprint]. Available at: https://arxiv.org/pdf/1412.6980.pdf

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. (2016). Pruning filters for efficient convnets. [Preprint]. Available at: https://arxiv.org/pdf/1608.08710.pdf

Li, W., Wong, W. H., and Jiang, R. (2019). Deeptact: predicting 3d chromatin contacts *via* bootstrapping deep learning. *Nucleic Acids Res.* 47, e60. doi: 10.1101/353284

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. (2017). Learning efficient convolutional networks through network slimming in *Proceedings of the IEEE International Conference on Computer Vision.* 2736–2744. doi: 10.1109/ICCV.2017.298

Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. (2018). Rethinking the value of network pruning. [Preprint]. Available at: https://arxiv.org/pdf/1810.05270.pdf

Luo, X., Tu, X., Ding, Y., Gao, G., and Deng, M. (2019). Expectation pooling: An effective and interpretable pooling method for predicting dna-protein binding. *Bioinf.* 658427. doi: 10.1093/bioinformatics/btz768

Pan, X., and Shen, H.-B. (2018). Predicting rna-protein binding sites and motifs through combining local and global deep convolutional neural networks. *Bioinf.* 34, 3427–3436. doi: 10.1093/bioinformatics/bty364

Pan, X., and Yan, J. (2017). Attention based convolutional neural network for predicting rna-protein binding sites. [Preprint]. Available at: https://arxiv.org/pdf/1712.02270.pdf

Pan, X., Rijnbeek, P., Yan, J., and Shen, H.-B. (2018). Prediction of rna-protein sequence and structure binding preferences using deep convolutional and recurrent neural networks. *BMC Genomics* 19, 511. doi: 10.1186/s12864-018-4889-1

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. [Preprint]. Available at: https://arxiv.org/pdf/1511.06434.pdf

Shen, Z., Bao, W., and Huang, D.-S. (2018). Recurrent neural network for predicting transcription factor binding sites. *Sci. Rep.* 8, 15270. doi: 10.1038/s41598-018-33321-1

Stormo, G. D. (2000). Dna binding sites: representation and discovery. *Bioinf.* 16, 16–23. doi: 10.1093/bioinformatics/16.1.16

Wingender, E., Dietze, P., Karas, H., and Knu¨ppel, R. (1996). Transfac: a database on transcription factors and their dna binding sites. *Nucleic Acids Res.* 24, 238–241. doi: 10.1093/nar/24.1.238

Zeng, H., Edwards, M. D., Liu, G., and Gifford, D. K. (2016). Convolutional neural network architectures for predicting dna–protein binding. *Bioinf.* 32, i121–i127. doi: 10.1093/bioinformatics/btw255

Zhang, Y., Liu, T., Meyer, C. A., Eeckhoute, J., Johnson, D. S., and Bernstein, B. E. (2008). Model-based analysis of chip-seq (macs). *Genome Biol.* 9, R137. doi: 10.1186/gb-2008-9-9-r137

Zhou, J., and Troyanskaya, O. G. (2015). Predicting effects of noncoding variants with deep learning–based sequence model. *Nat. Methods* 12, 931. doi: 10.1038/nmeth.3547

Zou, J., Huss, M., Abid, A., Mohammadi, P., Torkamani, A., and Telenti, A. (2019). A primer on deep learning in genomics. *Nat. Genet.* 51, 12–18. doi: 10.1038/s41588-018-0295-5

Zuallaert, J., Godin, F., Kim, M., Soete, A., Saeys, Y., and De Neve, W. (2018). Splicerover: Interpretable convolutional neuralnetworks for improved splice site prediction. *Bioinf.* 34, 4180–4188. doi: 10.1093/bioinformatics/bty497

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.