This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2825996, IEEE Access

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

DeepPolyA: a convolutional neural network approach for polyadenylation site prediction

XIN GAO[1,*], JIE ZHANG[2,*], ZHI WEI[1], (Senior Member, IEEE), and HAKON HAKONARSON[3,4]

[1]Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102, USA
[2]Adobe Systems, San Jose, CA 95110, USA
[3]The Center for Applied Genomics, Abramson Research Center, The Children's Hospital of Philadelphia, Philadelphia, PA 19104, USA
[4]Department of Pediatrics, University of Pennsylvania School of Medicine, Philadelphia, PA 19102, USA
[*]Both authors contributed equally to this work

Corresponding author: Zhi Wei (zhiwei@njit.edu)

ABSTRACT Polyadenylation (Poly(A)) plays crucial roles in gene regulation, especially in messenger RNA metabolism, protein diversification and protein localization. Accurate prediction of polyadenylation sites and identification of motifs that controlling polyadenylation are fundamental for interpreting the patterns of gene expression, improving the accuracy of genome annotation and comprehending the mechanisms that governing gene regulation. Despite considerable advances in using machine learning techniques for this problem, its efficiency is still limited by the lack of experiences and domain knowledge to carefully design and generate useful features, especially for plants. With the increasing availability of extensive genomic datasets and leading computational techniques, deep learning methods, especially convolutional neural networks, have been applied to automatically identify and understand gene regulation directly from gene sequences and predict unknown sequence profiles. Here, we present DeepPolyA, a new deep convolutional neural network-based approach, to predict polyadenylation sites from the plant Arabidopsis thaliana gene sequences. We investigate various deep neural network architectures and evaluate their performance against classical machine learning algorithms and several popular deep learning models. Experimental results demonstrate that DeepPolyA is substantially better than competing methods regarding various performance metrics. We further visualize the learned motifs of DeepPolyA to provide insights of our model and learned polyadenylation signals.

INDEX TERMS Polyadenylation prediction, deep learning, multi-layer neural network, motif discovery, genomics and machine learning algorithms.

I. INTRODUCTION

Polyadenylation is a vital process that occurs after gene transcription and produces mature messenger RNA (mRNA) for translation by synthesizing the polyadenylation tail at the RNA's 3'-end [1]. Recent discoveries have revealed that the 3'-end of most protein-coding and long-noncoding RNAs (lncRNAs; noncoding transcripts of 200 nucleotides or longer) is cleaved and polyadenylated [2]. In addition, alternative polyadenylation (APA) is prevalent in all eukaryotic species and plays critical roles in gene regulation, especially in the processes such as mRNA metabolism, protein diversification and protein localization [3]. Specifically, in addition to conducing to the intricacy of transcriptome by producing isoforms of distinct properties, it can regulate the translation efficiency, function, stability and localization of target RNAs [2], [4]. The polyadenylation site, also called the poly(A) site, is defined by surrounding RNA segments and conserved across metazoans with some minor variations in mammals [1]–[3]. Accurate prediction of poly(A) sites and identification of motifs that controlling them are fundamental for interpreting the patterns of gene expression, improving the accuracy of genome annotation and comprehending the mechanisms that governing gene regulation [5], [6].

However, this remains a challenging problem, especially for plants, to precisely identify the poly(A) signals and predict poly(A) sites. Unlike animals, plants possess much less conserved signal sequences in such regions [7]. For example, the upstream element signal "AAUAAA" (or "AATAAA" in DNA sequence), which has been identified as the best signal in plants, can only be found in approximately 10%

of Arabidopsis genes [8], [9]. In contrast, the same signal is utilized by 50% of human genes [10]. The variable structures composed of functional motifs [11], [12] also increase the difficulty in identifying poly(A) sites. In addition, because of the epidemic presence of alternative polyadenylation in intron and coding sequence (CDS), the poly(A) sites may locate in the genomic regions other than 3' untranslated region (3'-UTR). Thus, an ideal predictive model should be powerful and robust enough to overcome all barriers as mentioned above to achieve decent performance.

Quite a few methods have been proposed to predict poly(A) sites across diverse species. Among these studies, most of them focus on human sequences. Akhtar et al. proposed POLYAR, which applied the linear discriminant function (LDF) to classify poly(A) sites into three groups with distinct poly(A) signals [13]. A stand-alone program named polya_svm was developed for poly(A) sites prediction using the 15 cis-regulatory elements based on a Support Vector Machine (SVM) model [14]. Chang et al. proposed a predictive model of two SVMs for features extraction and poly(A) sites prediction [15]. All these three studies are conducted based on human genomic sequences dataset polya_DB[1] [16]. More recently, Xie et al. proposed a novel machine-learning method by marrying generative learning (hidden Markov models) and discriminative learning (support vector machines) [17]. Methods for analyzing other species, such as yeast and plants, were also proposed. Graber et al. proposed a contextual model to predict yeast poly(A) sites via a hidden Markov model (HMM) [18]. In view of the features of plant poly(A) signals, Ji et al. proposed a generalized hidden Markov model (GHMM) to effectively predict the poly(A) sites in Arabidopsis genes [7]. It yielded both high specificity and sensitivity in the testing datasets [8]. Later, the model was updated and re-trained for rice [19]. Recently, Ji et al. proposed a user-friendly framework called poly(A) site classifier (PAC) for predicting poly(A) sites in Arabidopsis genes [20]. PAC demonstrates the best performance with high specificity and sensitivity in the real data experiments. In addition, sub-models, like feature generation, feature selection and classification in PAC, could be replaced and updated, making it adaptable to different datasets [20].

Although methods mentioned above could achieve decent performance in solving the specific problem, researchers are required to carefully design and generate useful features based on their experiences and domain knowledge. Feature generation and extraction methods, including K-gram pattern, Z-curve, and position-specific scoring matrix, are critical components for previous SVM-based or HMM-based methods [7], [20], and the power of the method could be significantly reduced due to an inappropriate feature generation procedure. Thus, special efforts are needed to apply one method to another species.

With the increasing availability of extensive genomic datasets and leading computational techniques, deep-learning-based methods have been proposed to automatically identify and understand regulatory regions of the genome directly from DNA sequences, and predict the profile of unknown sequences based on learned knowledge [21]. Deep learning, in general, refers to methods that learn a hierarchical representation and detect complex patterns from feature-rich datasets through multiple layers of abstraction.

Amongst a set of deep neural networks, convolutional neural networks (CNN) are extensively employed in both academia and industry. It can achieve superb results in computer vision, video analysis and speech recognition for its efficient feature extraction capability [22], [23]. CNN has also been applied as the premier model in piles of genomic problems, for example, motif discovery [24], HLA class I-peptide binding prediction [25], and identifying functional effects of noncoding variants [26], [27]. Zeng et al. proposed a series of CNN architectures to identify DNA sequence binding with a large compendium of transcription factor datasets [28]. Basset, a powerful computational tool, was proposed to apply CNN to discover the functional activity of genomic sequences [29]. Zhou et al. applied CNN model to capture the motif signals from the sequences around the target residues [30]. DeepSEA is a recently developed algorithm that utilizes CNN for predicting chromatin effects of sequence alterations with single-nucleotide sensitivity [26]. DanQ, a hybrid framework that combines convolutional and recurrent neural networks, further improves the performance of DeepSEA [27]. CNN models have demonstrated their advantages in automatically learning hierarchical feature representations of raw input data in previous studies. Their successes motivate us to develop novel CNN-based methods to automatically learn poly(A)-related features, signals, and patterns for predicting poly(A) sites.

In this paper, we propose a computational method, DeepPolyA, based on deep CNN for predicting poly(A) sites in Arabidopsis species. DeepPolyA automatically combines the feature extraction and model training stages, and learns predictive DNA patterns and motifs in a data-driven manner. In this work, we have made four contributions:

1) We propose a CNN-based model named DeepPolyA [2] to predict poly(A) sites in Arabidopsis. To the best of our knowledge, this is the first deep learning based approach to this research issue.
2) We show in this paper that DeepPolyA could automatically learn poly(A)-related motifs without involving any manual feature engineering. The model first learns low-layer features from DNA sequences via lower convolutional layers, and then forms high-level, sophisticated features through upper nonlinear transformation layers.
3) DeepPolyA outperforms not only the conventional machine learning methods including Support Vector Machine (SVM), Bayesian Networks and Random Forest,

[1]http://polya.umdnj.edu/PolyA_DB1/

[2]DeepPolyA is freely available on Github: https://github.com/stella-gao/DeepPolyA

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2825996, IEEE Access

but also the existing deep learning models including DanQ [27], DeepSEA [26], and VGG [31] models.

4) We also investigate the performance of alternative deep learning architectures in predicting poly(A) signals, including the recurrent neural network (RNN) model and the combination of CNN and RNN models (CNN-RNN).

II. METHODS

A. BACKGROUND

Recent studies have demonstrated that deep learning can solve genomic problems in a more accurate way than traditional machine learning approaches. Recent works based on CNN allowed training directly on genomic sequences rather than extracting features beforehand [24], [26], [29]. In CNN, the number of model parameters is significantly decreased compared to fully connected neural networks via convolution operation and parameter sharing. In addition, convolutional layers could extract high-level features from the raw sequences, which has the similar function to the traditional position-weight matrices (PWMs) [32]. Neurons in a fully connected layer have full connections to all activations in the previous layer, while neurons in a convolutional layer share parameters in a particular feature map. The most relevant motifs are automatically inferred by the deeper layers during model training, and can be visualized and analyzed through heatmaps and sequence logos [33].

B. CONSTRUCTION OF DEEPPOLYA MODEL

For the poly(A) site prediction problem, we illustrate the architecture of our DeepPolyA model in Figure 1. It consists of two convolutional layers and one max-pooling layer to identify predictive motifs from the context of DNA sequences and one fully connected hidden layers with a ReLU activation function to model motif interactions [21]. Techniques, like dropout [34], batch normalization [35] and early stopping, are employed to prevent overfitting. Table 1 shows the specification of each layer. The hyperparameters (e.g., convolution kernel size, number of layers, dropout rate, etc.) are selected based on the performance on a validation dataset. As shown in Figure 1 and Table 1, DeepPolyA takes a 162-nucleotide (nt) length DNA sequence as input with the position 131 as the target poly(A) site. The raw DNA sequence is encoded into a bit matrix using one-hot encoding method. We encode the entire DNA nucleotides sequentially, where each nucleotide is encoded to a four-element binary vector with only one element set to one and others set to zero: A=(1,0,0,0), C=(0,1,0,0), G=(0,0,1,0), and T=(0,0,0,1). A 162-nt DNA sequence can then be represented as an encoded 4×162 bit matrix, with columns corresponding to A, C, G and T. With one-hot encoding method, we can preserve the vital position information of each nucleotide in DNA sequences.

As shown in Table 1, there are 16 and 64 kernels in the first and second convolutional layers, respectively. Note that the architecture parameters (e.g., the kernel size, the number of layers and the dropout rate) are carefully selected

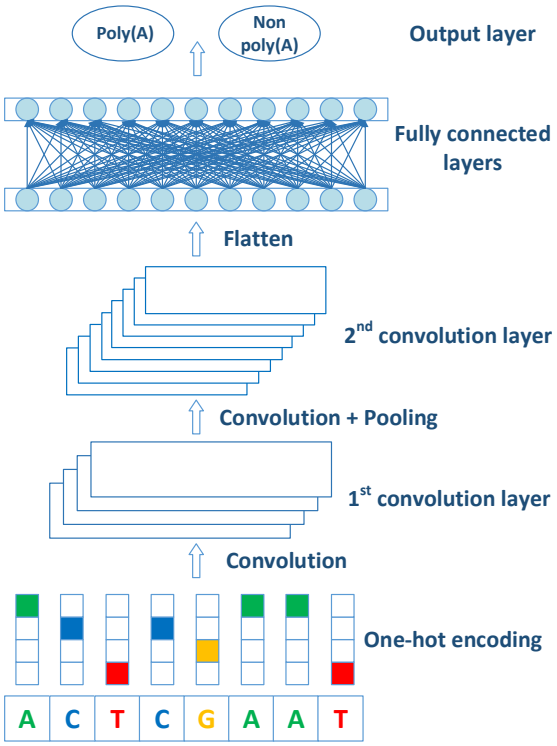


FIGURE 1: Overview of the neural network architecture of DeepPolyA.

TABLE 1: DeepPolyA architecture and hyperparameters. The size column describes the kernel size of the convolutional layer, the window size of the max-pooling layer and the size of the fully connected layer.

Layer No.	Layer Type	Size	Output
0	INPUT	-	4*162
1	CONV	16*4*8	16*157
2	RELU	-	16*157
3	DROPOUT	-	16*157
4	CONV	64*4*6	64*152
5	RELU	-	64*152
6	POOL	4*2	64*76
7	DROPOUT	-	64*76
8	FC	64	64
9	DROPOUT	-	64
10	FC	1	1
11	SIGMOID	1	1

based on the optimization performance on the validation set. By applying several convolutional and pooling operations, CNN could automatically extract high-level features from high-dimensional input data while making the number of model weights manageable. Model parameters are randomly initialized as suggested by Glorot et al. [36]. Model hyperparameters (e.g., learning rate and the number of epochs) are optimized based on the performance of the validation data. Note that validation loss is measured after each training epoch to monitor convergence. Dropout (dropout rate 40%;

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2825996, IEEE Access

i.e., randomly drop 40% neurons in each iteration) and batch normalization [35] are used for additional regularization. Note that dropout is not suitable to be put in the last fully-connected layer, because some significant features may be lost. Thus, we only apply dropout between the hidden layers.

The first convolutional layer operates directly on the encoded bit matrix, where the kernels scan for features across the matrix, compute the activation of multiple kernels at every position within the DNA sequence window and generate output matrices. To reduce dimensionality and accelerate convergence, max-pooling layer and batch normalization layer are applied subsequently, where the pooling size and stride size are both set to 2 to prevent any overlap. A second convolutional layer is added to model the interactions between motifs generated by previous layers and obtain high-level features and abstractions. The output of the convolutional layers is then flattened into vectors and fed to the fully connected layer. Finally, the outputs are converted into probabilities via "sigmoid" function.

The value of the proposed deep learning based method is two-fold. Firstly, classical machine learning methods require researchers' prior knowledge and experience to predefine and cultivate features by counting or summarizing known genomic patterns (e.g., regulatory variants, k-mer and structural elements). In contrast, our method can automatically learn the problem-related knowledge and extract high-level features from the raw sequence data. Therefore our approach could be readily applied to solve the same poly(A) prediction problem in different species. Secondly, the proposed method can capture nonlinear dependencies and interactions among the detected patterns and features at multiple genomic scales. It is challenging for even experienced researchers to design a schema of features including all potential interactions of different sub-signals and low-level features. Thus, it can yield better performance as demonstrated in the real data experiments.

III. EXPERIMENTS AND RESULTS

In this section, we describe the experimental environment, platform settings used for building models as well as the evaluation performance of the proposed model compared with the state-of-the-art approaches. The experiments show that DeepPolyA yields significantly more accurate predictions than existing baseline machine learning methods and other popular deep learning methods.

A. DATA SOURCE AND DATA PREPROCESSING

A large number of training samples are usually required to train deep neural networks in order to learn informative features and high-level representations from scratch. As a general guideline, the number of training samples should be at least as many as that of model parameters, although overfitting can be reduced through special architectures and model regularization techniques [37]. The same Arabidopsis datasets chosen from the baseline PAC's literature [8] were utilized for our experiments. Positive samples are randomly sampled from 16K dataset, which has over 16,000 Arabidopsis 3'-UTRs plus downstream sequences[3]. Following Ji et al [20], we selected the sequences, whose lengths are longer than 162 nt, for further analysis. As a result, 13427 positive sequences, which are equally distributed among five chromosomes (chr 1-5), are obtained. Negative samples are generated by randomly sampling sequences from the Arabidopsis Information Resources (TAIR) database[4], which consists of unequal-length introns, coding sequences and 5'-UTR sequences. The extracted negative samples contain 3222 introns, 9704 coding sequences and 501 5'-UTRs of Arabidopsis DNA sequences. Note that it maintains the same distribution of sequences as in the TAIR dataset and contributes 13427 negative sequences in total. As presented in [20], each sequence is then trimmed into a sequence of size 162 nt, containing 131 nt upstream and 31 nt downstream of a poly(A) site. After preprocessing, all experimental samples are 162nt-long genomic sequences.

B. MODEL TRAINING AND TESTING

We divide the whole procedure of model training and testing into three steps: Firstly, we configure and apply different hyperparameters to train the models. Secondly, during the training process, we aim to discover the parameters (also called weights) that can minimize the objective function, which is challenging due to the high dimensionality and non-convex. We tune the hyperparameters to find out the weights with the best performance on the validation set for the model. Finally, the model is evaluated against other machine learning models and popular deep learning models on the test set. The training performance significantly relies on parameter initialization, learning rate, and batch size of stochastic gradient descent. In our proposed model, model weights are initialized and sampled from a truncated normal distribution centered on zero with the square root of the average number of both the input units and the output units of the input layer [36], [38]. Since the model aims to classify the input into two categories: sequences with poly(A) sites and sequences without poly(A) sites, it is a two-class logistic regression problem. Thus, we prefer to adopt a sigmoid function rather than a softmax function that is used for a multiclass problem. Advanced adaptive learning rate methods, such as RMSprop, Adagrad [39], and Adam [40], are also applied to reduce the effect of initial and potentially sub-optimal learning rate for model training. Based on the preliminary experiments, the hyperparameters, such as a standard stochastic gradient descent optimizer, a learning rate of 0.001, a batch size of 128 and a momentum rate of 0.9, are appropriate for our models. Regularization, ensemble learning and cross-validated evaluation are always used for reducing overfitting. To reduce overfitting, we adopt the most common regularization technique – dropout with a dropout rate of 40%. Another popular method we employed is "early stopping", which means the training process will

[3]http://www.users.miamioh.edu/liq/links.html
[4]http://www.arabidopsis.org/

be stopped automatically at the best point. Thus, the latest parameters that perform best on the validation set are chosen as soon as the training process stopped.

We implement the proposed models with python, Keras (Version 2.0.8) [41], and Theano (Version 0.9.0) [42]. Keras [41] is a powerful python library which can run on top of Theano [42] or TensorFlow [43]. It provides highly modularized APIs for building and training deep learning models. We configure the experiment environment and run the python codes on a Linux server with 4 Intel E5-2650 CPUs, 256 GB memory and 4 Nvidia GeForce GTX 1080 GPUs (2560 NVIDIA CUDA cores, 8 GB GDDR5X memory and 10Gbps memory speed).

C. PREDICTION ASSESSMENT

To evaluate competing methods, we adopt several metrics including sensitivity (Sn), specificity (Sp), overall accuracy (ACC), Matthew's correlation coefficient (MCC), area under the receiver operating characteristic curve (AUC) and F-score. Let TP, TN, FP and FN represent true positive, true negative, false positive and false negative, respectively. As shown in Equation (1)–(6), we consider the sequences containing poly(A) sites as the positive samples and the sequences without poly(A) sites as the negative samples, and compute all the metrics accordingly. Sensitivity (Sn), also called recall, measures the proportion of the true positives that are correctly predicted. Specificity (Sp) measures the proportion of the true negatives that are correctly predicted. As a common metric in classification, the overall accuracy (ACC) is used to describe the closeness of a measurement to the true value and it fails when in the presence of highly imbalanced classes. MCC can describe the confusion matrix of TP, TN, FP and FN by a single number. MCC essentially is a correlation coefficient between the observed and predicted binary classifications, which could also be seen as a correlation coefficient between training and testing datasets. In addition, we adopt AUC and F-score to evaluate the performance and usability of all the models.

$$S_n = \frac{TP}{TP + FN} \tag{1}$$

$$S_p = \frac{TN}{TN + FP} \tag{2}$$

$$Accuracy = \frac{TP + TN}{TN + FP + TP + FN} \tag{3}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TN + FN)(TN + FP)(TP + FN)}} \tag{4}$$

$$AUC = \frac{1}{2}(S_n + S_p) \tag{5}$$

$$F_{score} = \frac{2TP}{2TP + FP + FN} \tag{6}$$

D. EXPERIMENTS ON MODEL COMPARISON

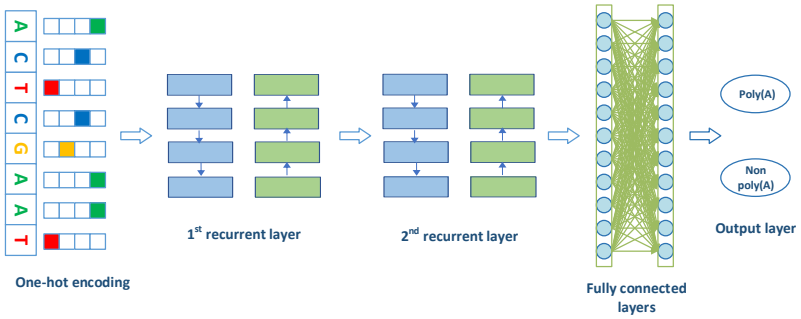
We compare the proposed method with state-of-the-art methods including Support Vector Machine (PAC.SVM), Bayesian Networks (PAC.BN), Random Forest (PAC.RF) [20] as well as other deep neural network models such as recurrent neural network (RNN), hybrid convolutional and recurrent neural network (CNN-RNN). Note that the architecture and parameters for RNN and CNN-RNN are tuned via validation set, which is exactly the same as for our proposed method. Figure 2(a) shows a graphical illustration of the RNN model, including two Long Short-Term Memory (LSTM) layers and one fully connected layer. Both LSTM layers contain 16 neurons and the fully connected layer contains 64 neurons. The dropout rate of each LSTM is set to 0.2. Figure 2(b) shows a graphical illustration of the CNN-RNN model, which is built with two convolutional layers, one max-pooling layer, one LSTM layer followed by one fully connected layer. To build a CNN-RNN model, we add one LSTM layer to DeepPolyA model between its second convolutional layer and the final fully connected layer. We also consider some other popular deep learning models (e.g., DeepSEA [26], DanQ [27], VGG [31]), fit them into our poly(A) problem, and compare them with our newly proposed CNN architecture (DeepPolyA).

For the traditional machine learning approaches (PAC.SVM, PSC.NB and PAC.RF), we follow Ji et al. and extract features from DNA sequences, including 1) the frequency of some nearby nucleotides; 2) Hexamer weight of NUE region; 3) PSSM-based CIS score of NUE region or CS region; 4) components of Z-curve [20]. After feature extraction, we train the model based on different algorithms with the default parameter settings in Weka [44]. It should be noted that, as shown by the preliminary experiments, reasonable choices of the parameters (e.g. different type of kernels for Support Vector Machine, different number of decision trees for Random Forest) yield similar prediction results.

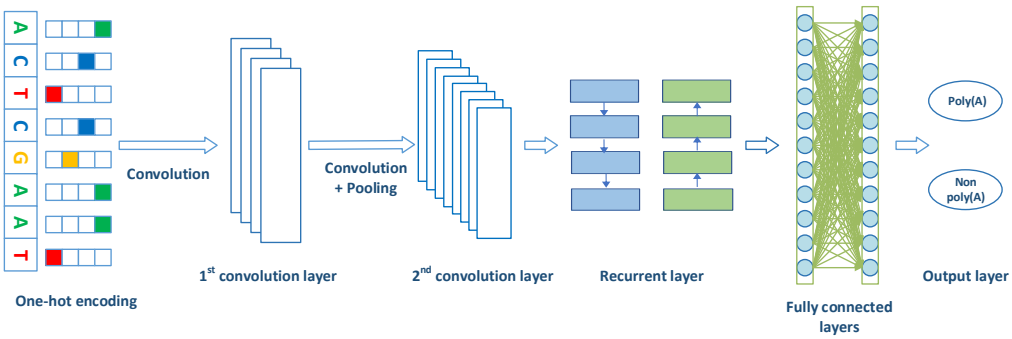
We divide the entire dataset into training, validation and testing sets. The training set is utilized to learn weights and parameters of the models, which are then evaluated on the validation set. The best model is selected and tested on the testing set to evaluate the model performance. Note that the data are partitioned on chromosome level and there is no overlap between training and testing data (i.e., three chromosomes for training, one for validation and one for testing). In each run, we randomly pick one chromosome as validation set and one chromosome as testing set, and leave the rest three chromosomes as the training set. We repeat the experiments several times to demonstrate that our model could achieve better performance, and it is not obtained by chance. All the hyperparameters of above-mentioned competing methods were selected based on the performance on the validation data. The model performance is then evaluated on the testing set.

As shown in Figure 3, we repeat the experiment 5 times and show the prediction performance based on testing data. DeepPolyA achieves remarkable and stable performance in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2825996, IEEE Access



(a) RNN model



(b) CNN-RNN model

FIGURE 2: RNN and CNN-RNN model architectures.

classification on the testing set with an accuracy of 91.28%, a recall of 93.01%, a specificity of 89.51%, a Matthews correlation coefficient of 82.60%, an area under a receiver operating characteristic (ROC) curve of 97.06% and an F-measure of 91.51%, which outperforms all other methods among all metrics. The two methods next to DeepPolyA are DanQ and DeepSEA.

The RNN method can get a high performance sometimes, but the result fluctuates according to different training and testing set. We can also discover that the performance of three baseline machine learning methods is always worse than deep learning methods. Overall, the results indicate that DeepPolyA can automatically learn high-level features from the DNA sequences and yields more accurate predictions than classical approaches and other popular deep learning models. A major advantage of our model compared to previous methods is its convolutional architecture, which allows for discovering predictive motifs in larger DNA sequence contexts, as well as for capturing complex patterns around poly(A) sites.

Figure 4 shows the area under receiver operating characteristics (ROC curve) and the area under precision-recall curve (PR curve) for deep neural network based methods. We can find from the curves that DeepPolyA demonstrates the best performance. The following methods are DanQ, DeepSEA, CNN-RNN, VGG and RNN.

We further investigate the impacts of the length of input sequences by varying the input DNA sequence length from 54 nt to 216 nt. As shown in Figure 5, the performance of the proposed model increases when sequence length increases from 54 nt to 162 nt, and it keeps the same from 162 nt to 216 nt. Note that the performance will not increase if we further improve the input sequence length. These observations suggest that the sequences around poly(A) sites do contain the poly(A) signals and thus an appropriate input sequence length is critical for predicting the poly(A) sites (i.e., shorter sequence will miss some signals and longer sequences will include lots of noise).

We also evaluate the performance of our model in different genomic contexts. As shown in Figure 6, the model yields the best performance in CDS, and the worst performance in intron, suggesting that poly(A) sites located in introns may have less conserved signals and therefore are harder to detect.

E. VISUALIZATION OF LEARNED MOTIFS

Model visualization is critical in computational biology. Several approaches have been proposed to interpret the parameters of neural networks and to obtain insights into learned features. Similar to conventional position weight matrices (PWMs), gene sequence motifs can be recognized by the filters of the first convolutional layer in DeepPolyA and visualized as sequence logos. Traditional approaches for visualizing convolutional filters could be generally classified into alignment-based and optimization-based methods.

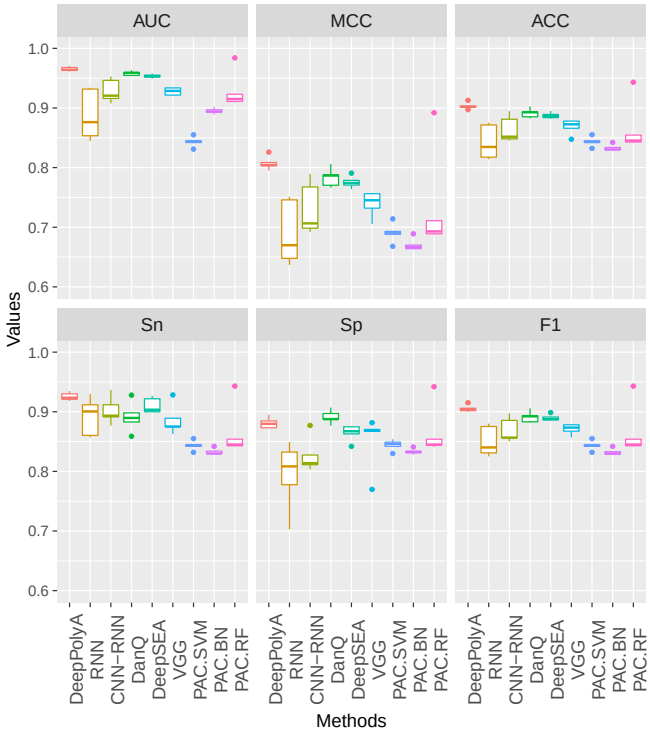


FIGURE 3: Prediction performance of DeepPolyA comparing with other methods. Performance is measured using boxplot by six evaluation metrics, including the area under the receiver-operating characteristic curve (AUC), accuracy (ACC), recall (Sn), specificity (Sp), Matthew's correlation coefficient (MCC), and F-score (F1). Three constructed deep neural network methods (including DeepPolyA), three referred popular deep learning methods and three baseline methods are considered. The lower and upper hinges are the 25 and 75 quartiles, respectively.

Alignment-based approaches [24], [27], [29] align DNA sequence fragments that maximize the activation of a certain convolutional filter and visualize the outcome alignments as sequence logos using WebLogo [45]. Optimization-based approaches [46] optimize the input gene sequence to maximize the activation of a certain convolutional filter by gradient descent. Following the alignment-based method, we accumulate the motifs for the total 16 filters in DeepPolyA and assess them through comparing against JASPAR database [47], which has abundant known motifs and is widely applied as standard representation of transcription factor DNA-binding preferences. We utilize a tool named TOMTOM with predefined statistical measure of motif-motif similarity [48], [49] to compare the motifs detected by DeepPolyA with those in JASPAR database and present an alignment for each pair of considerable matches. Our visualization results of extracted motifs and comparison with JASPAR are shown in Figure 7. The second motif among them has the similar format with "AATAAA", which is a significant upstream element signal for poly(A) sites. Moreover, Figure 8 illustrates the saliency map [50] visualization of the entire testing DNA sequence. We can discover that the signal is strong around the poly(A) site. As shown in Figure 8, the strongest signal is at position 131 nt, which is exactly the position of the target poly(A) sites in positive samples.

IV. CONCLUSION AND FUTURE WORKS

In this paper, we proposed DeepPolyA, a deep convolutional neural network approach, to automatically and accurately modeling the poly(A) signals. Using the plant Arabidopsis thaliana gene sequences datasets with one-hot encoding method, we trained several competing deep learning models with various architectures and compared the classification performance with baseline machine learning methods through several significant metrics. The evaluation results show that DeepPolyA outperforms all the competing methods without involving extensive manual feature engineering. We visualized the learned motifs of the first convolutional layer using TOMTOM against the JASPAR motif datasets to demonstrate that DeepPolyA can automatically extract poly(A) signals and features from the raw sequence data.

REFERENCES

[1] D. Xing and Q. Q. Li, "Alternative polyadenylation and gene expression regulation in plants," Wiley Interdisciplinary Reviews: RNA, vol. 2, no. 3, pp. 445–458, 2011.

[2] R. Elkon, A. P. Ugalde, and R. Agami, "Alternative cleavage and polyadenylation: extent, regulation and function," Nature Reviews Genetics, vol. 14, no. 7, pp. 496–506, 2013.

[3] B. Tian and J. L. Manley, "Alternative polyadenylation of mrna precursors," Nature Reviews Molecular Cell Biology, 2016.

[4] X. Liu, M. Hoque, M. Larochelle, J.-F. Lemay, N. Yurko, J. L. Manley, F. Bachand, and B. Tian, "Comparative analysis of alternative polyadenylation in s. cerevisiae and s. pombe," Genome research, vol. 27, no. 10, pp. 1685–1695, 2017.

[5] S. Zhang, J. Han, J. Liu, J. Zheng, and R. Liu, "An improved poly(a) motifs recognition method based on decision level fusion," Computational biology and chemistry, vol. 54, pp. 49–56, 2015.

[6] Y. Liu, P. Wu, J. Zhou, T. L. Johnson-Pais, Z. Lai, W. H. Chowdhury, R. Rodriguez, and Y. Chen, "Xbseq2: a fast and accurate quantification of differential expression and differential polyadenylation," BMC bioinformatics, vol. 18, no. 11, p. 384, 2017.

[7] G. Ji, J. Zheng, Y. Shen, X. Wu, R. Jiang, Y. Lin, J. C. Loke, K. M. Davis, G. J. Reese, and Q. Q. Li, "Predictive modeling of plant messenger rna polyadenylation sites," BMC bioinformatics, vol. 8, no. 1, p. 43, 2007.

[8] J. C. Loke, E. A. Stahlberg, D. G. Strenski, B. J. Haas, P. C. Wood, and Q. Q. Li, "Compilation of mrna polyadenylation signals in arabidopsis revealed a new signal element and potential secondary structures," Plant physiology, vol. 138, no. 3, pp. 1457–1468, 2005.

[9] Q. Li and A. G. Hunt, "A near-upstream element in a plant polyadenylation signal consists of more than six nucleotides," Plant molecular biology, vol. 28, no. 5, pp. 927–934, 1995.

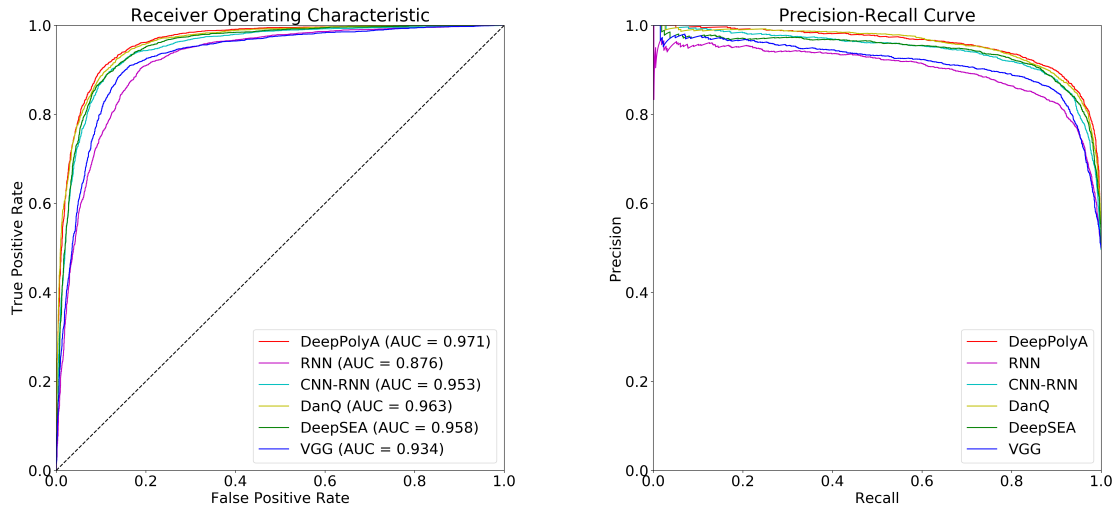
[10] J. Hu, C. S. Lutz, J. Wilusz, and B. Tian, "Bioinformatic identification of candidate cis-regulatory elements involved in human mrna polyadenylation," Rna, vol. 11, no. 10, pp. 1485–1493, 2005.

[11] X. Wu, M. Liu, B. Downie, C. Liang, G. Ji, Q. Q. Li, and A. G. Hunt, "Genome-wide landscape of polyadenylation in arabidopsis provides evidence for extensive alternative polyadenylation," Proceedings of the National Academy of Sciences, vol. 108, no. 30, pp. 12 533–12 538, 2011.

[12] V. Solovyev and R. Umarov, "Prediction of prokaryotic and eukaryotic promoters using convolutional deep learning neural networks," arXiv preprint arXiv:1610.00121, 2016.

[13] M. N. Akhtar, S. A. Bukhari, Z. Fazal, R. Qamar, and I. A. Shahmuradov, "Polyar, a new computer program for prediction of poly(a) sites in human sequences," Bmc Genomics, vol. 11, no. 1, p. 646, 2010.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2825996, IEEE Access



(a) ROC curves

(b) Precision-Recall curves

FIGURE 4: ROC curves and Precision-Recall curves of the prediction performance among all the deep learning methods.

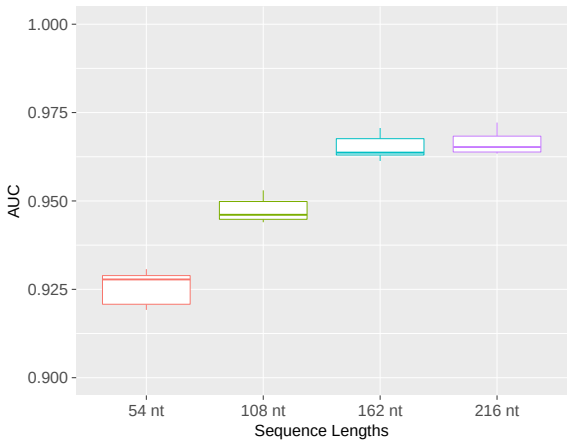


FIGURE 5: Prediction performance of DeepPolyA for DNA sequence windows of length 54 nt to 216 nt.

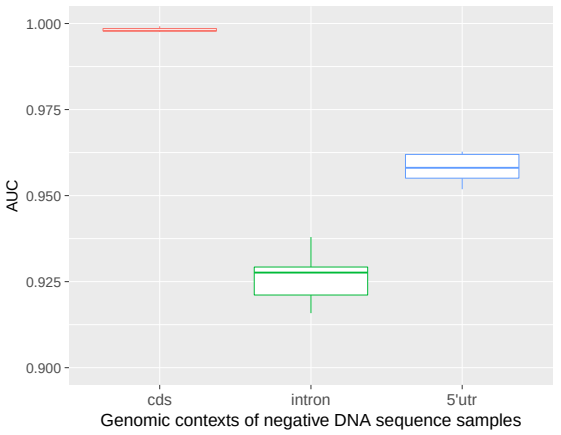


FIGURE 6: Prediction performance of DeepPolyA for alternative genomic contexts of negative DNA sequence samples.

[14] Y. Cheng, R. M. Miura, and B. Tian, "Prediction of mrna polyadenylation sites by support vector machine," Bioinformatics, vol. 22, no. 19, pp. 2320–2325, 2006.
[15] T.-H. Chang, L.-C. Wu, Y.-T. Chen, H.-D. Huang, B.-J. Liu, K.-F. Cheng, and J.-T. Horng, "Characterization and prediction of mrna polyadenylation sites in human genes," Medical & biological engineering & computing, vol. 49, no. 4, pp. 463–472, 2011.
[16] J. Y. Lee, I. Yeh, J. Y. Park, and B. Tian, "Polya_db 2: mrna polyadenylation sites in vertebrate genes," Nucleic acids research, vol. 35, no. suppl 1, pp. D165–D168, 2007.
[17] B. Xie, B. R. Jankovic, V. B. Bajic, L. Song, and X. Gao, "poly(a) motif prediction using spectral latent features from human dna sequences," Bioinformatics, vol. 29, no. 13, pp. i316–i325, 2013.
[18] J. H. Graber, G. D. McAllister, and T. F. Smith, "Probabilistic prediction of saccharomyces cerevisiae mrna 3'-processing sites," Nucleic acids research, vol. 30, no. 8, pp. 1851–1858, 2002.
[19] Y. Shen, G. Ji, B. J. Haas, X. Wu, J. Zheng, G. J. Reese, and Q. Q. Li, "Genome level analysis of rice mrna 3'-end processing signals and alternative polyadenylation," Nucleic acids research, vol. 36, no. 9, pp. 3150–3161, 2008.
[20] G. Ji, X. Wu, Y. Shen, J. Huang, and Q. Q. Li, "A classification-based prediction model of messenger rna polyadenylation sites," Journal of theoretical biology, vol. 265, no. 3, pp. 287–296, 2010.
[21] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.
[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, 2012, pp. 1097–1105.
[23] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural networks, vol. 61, pp. 85–117, 2015.
[24] B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey, "Predicting the sequence specificities of dna-and rna-binding proteins by deep learning," Nature biotechnology, vol. 33, no. 8, pp. 831–838, 2015.
[25] Y. S. Vang and X. Xie, "Hla class i binding prediction via convolutional neural networks," Bioinformatics, p. btx264, 2017.
[26] J. Zhou and O. G. Troyanskaya, "Predicting effects of noncoding variants with deep learning-based sequence model," Nature methods, vol. 12, no. 10, pp. 931–934, 2015.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2825996, IEEE Access

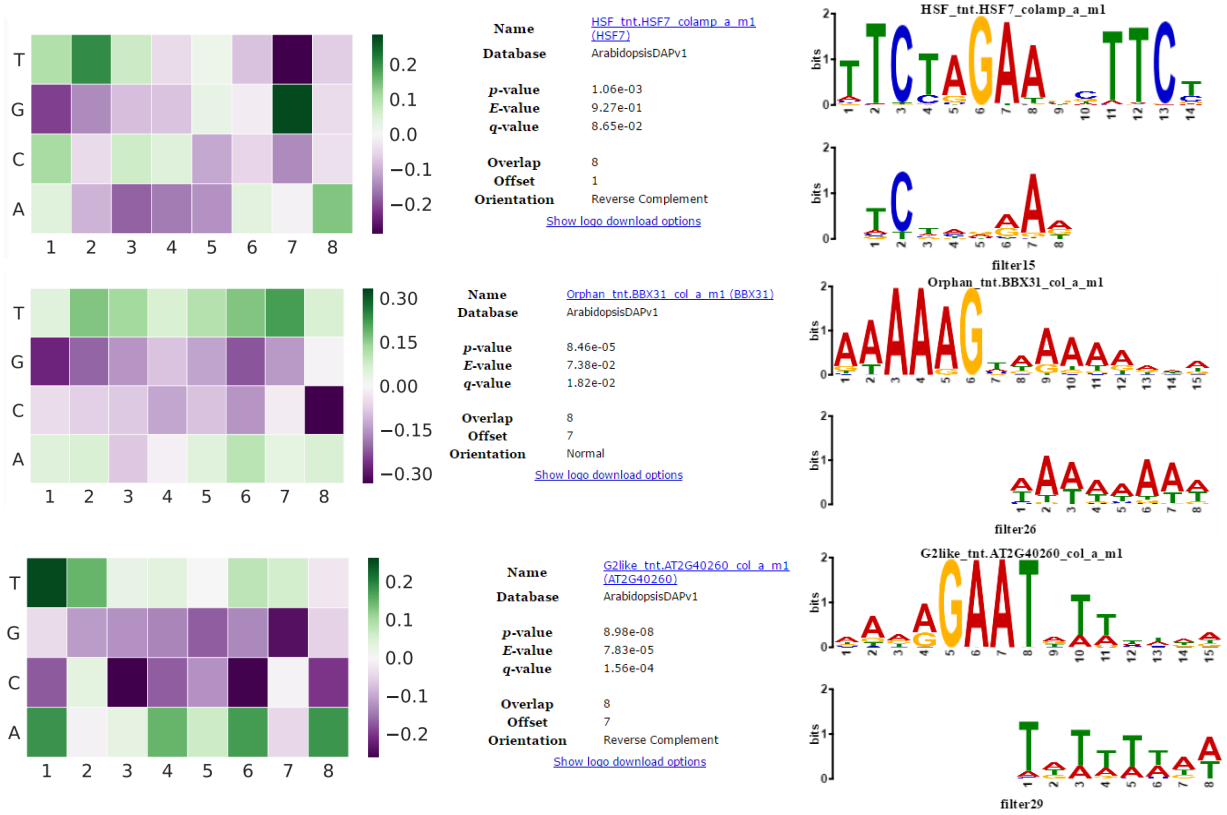


FIGURE 7: Three convolution kernels visualized from JASPAR using TOMTOM.

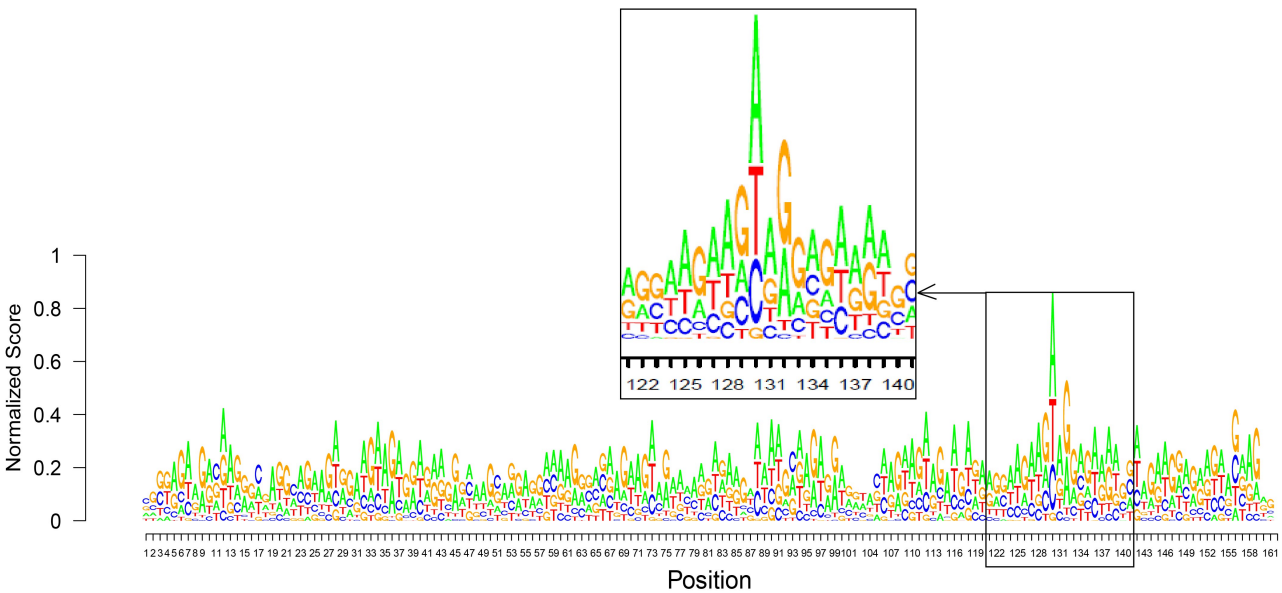
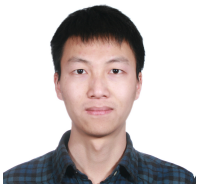


FIGURE 8: Saliency map visualization of an entire sequence with a zoom-in view of the sites around poly(A) sites.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2825996, IEEE Access

[27] D. Quang and X. Xie, "Danq: a hybrid convolutional and recurrent deep neural network for quantifying the function of dna sequences," Nucleic acids research, vol. 44, no. 11, pp. e107–e107, 2016.
[28] H. Zeng, M. D. Edwards, G. Liu, and D. K. Gifford, "Convolutional neural network architectures for predicting dna–protein binding," Bioinformatics, vol. 32, no. 12, pp. i121–i127, 2016.
[29] D. R. Kelley, J. Snoek, and J. L. Rinn, "Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks," Genome research, vol. 26, no. 7, pp. 990–999, 2016.
[30] J. Zhou, Q. Lu, R. Xu, L. Gui, and H. Wang, "Cnnsite: Prediction of dna-binding residues in proteins using convolutional neural network with sequence features," in Bioinformatics and Biomedicine (BIBM), 2016 IEEE International Conference on. IEEE, 2016, pp. 78–85.
[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.
[32] G. D. Stormo, T. D. Schneider, L. Gold, and A. Ehrenfeucht, "Use of the 'perceptron' algorithm to distinguish translational initiation sites in e. coli," Nucleic acids research, vol. 10, no. 9, pp. 2997–3011, 1982.
[33] T. D. Schneider and R. M. Stephens, "Sequence logos: a new way to display consensus sequences," Nucleic acids research, vol. 18, no. 20, pp. 6097–6100, 1990.
[34] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," arXiv preprint arXiv:1207.0580, 2012.
[35] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in International Conference on Machine Learning, 2015, pp. 448–456.
[36] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.
[37] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in Neural networks: Tricks of the trade. Springer, 2012, pp. 437–478.
[38] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.
[39] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," Journal of Machine Learning Research, vol. 12, no. Jul, pp. 2121–2159, 2011.
[40] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.
[41] F. Chollet et al., "Keras," 2015.
[42] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," arXiv e-prints, vol. abs/1605.02688, May 2016. [Online]. Available: http://arxiv.org/abs/1605.02688
[43] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/
[44] G. Holmes, A. Donkin, and I. H. Witten, "Weka: A machine learning workbench," in Intelligent Information Systems, 1994. Proceedings of the 1994 Second Australian and New Zealand Conference on. IEEE, 1994, pp. 357–361.
[45] G. E. Crooks, G. Hon, J.-M. Chandonia, and S. E. Brenner, "Weblogo: a sequence logo generator," Genome research, vol. 14, no. 6, pp. 1188–1190, 2004.
[46] J. Lanchantin, R. Singh, B. Wang, and Y. Qi, "Deep motif dashboard: Visualizing and understanding genomic sequences using deep neural networks," arXiv preprint arXiv:1608.03644, 2016.
[47] A. Mathelier, O. Fornes, D. J. Arenillas, C.-y. Chen, G. Denay, J. Lee, W. Shi, C. Shyr, G. Tan, R. Worsley-Hunt et al., "Jaspar 2016: a major expansion and update of the open-access database of transcription factor binding profiles," Nucleic acids research, vol. 44, no. D1, pp. D110–D115, 2016.
[48] T. L. Bailey, M. Boden, F. A. Buske, M. Frith, C. E. Grant, L. Clementi, J. Ren, W. W. Li, and W. S. Noble, "Meme suite: tools for motif discovery and searching," Nucleic acids research, vol. 37, no. suppl_2, pp. W202–W208, 2009.
[49] S. Gupta, J. A. Stamatoyannopoulos, T. L. Bailey, and W. S. Noble, "Quantifying similarity between motifs," Genome biology, vol. 8, no. 2, p. R24, 2007.
[50] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," arXiv preprint arXiv:1312.6034, 2013.

XIN GAO (S'14) received her B.E. degree in Information Security and LL.B. degree in Law from Nankai University, Tianjin, China in 2012. Currently she is a Ph.D. candidate at the Department of Computer Science, New Jersey Institute of Technology, Newark, NJ in 2017. Her research interests include deep learning, machine learning, bioinformatics and data mining.



JIE ZHANG (S'15) received his B.E. degree in Software Engineering from Nanjing University, Nanjing, China in 2012, and Ph.D. degree in Computer Science from New Jersey Institute of Technology, Newark, NJ in 2017. Currently, he is a data scientist at Adobe. His research interests include data mining, bioinformatics, statistical modeling and machine learning.

ZHI WEI (SM'17) received B.S. degree in Computer Science from Wuhan University, China, and Ph.D. degree in Bioinformatics from the University of Pennsylvania, Philadelphia, PA in 2008. Currently he is an associate professor at the Department of Computer Science, New Jersey Institute of Technology. His research interests include statistical modelling, machine learning and big data analytics. His works have been published in prestigious journals including Nature, Nature Medicine, JASA, Biometrika, AOAS, AJHG, PLoS Genetics, Bioinformatics, and Biostatistics. He is an associate editor of BMC Bioinformatics, BMC Genomics, IEEE Transactions on Computational Social Systems, and PLoS ONE.

HAKON HAKONARSON received his M.D. and Ph.D. from University of Iceland School of Medicine 1986 and 2002, respectively. Currently he is a professor of Pediatrics at University of Pennsylvania School of Medicine, with research focused on human genetics. He has published numerous high-impact papers on genomic discoveries and their translations in some of the most prestigious scientific medical journals, including Nature, Nature Genetics and The New England Journal of Medicine.