International Journal of Machine Learning and Cybernetics (2020) 11:841–851
https://doi.org/10.1007/s13042-019-00990-x

ORIGINAL ARTICLE

DeepSite: bidirectional LSTM and CNN models for predicting DNA–protein binding

Yongqing Zhang[1,2] · Shaojie Qiao[3] · Shengjie Ji[1] · Yizhou Li[4]

Received: 28 June 2018 / Accepted: 22 July 2019 / Published online: 29 July 2019

Abstract
Transcription factors are *cis*-regulatory molecules that bind to specific sub-regions of DNA promoters and initiate transcription, the process that regulates the conversion of genetic information from DNA to RNA. Several computational methods have been developed to predict DNA–protein binding sites in DNA sequence using convolutional neural network (CNN). However, these techniques could indicate the dependency information of DNA sequence information in the framework of CNN. In addition, these methods are not accurate enough in prediction of the DNA–protein binding sites from the DNA sequence. In this study, we employ the bidirectional long short-term memory (BLSTM) and CNN to capture long-term dependencies between the sequence motifs in DNA, which is called DeepSite. Apart from traditional CNN, which includes six layers: input layer, BLSTM layer, CNN layer, pooling layer, full connection layer and output layer, DeepSite approach can predict DNA–protein binding sites with 87.12% sensitivity, 91.06% specificity, 89.19% accuracy and 0.783 *MCC*, when tested on the 690 Chip-seq experiments from ENCODE. Lastly, we conclude that our proposed method can also be applied to find DNA–protein binding sites in different DNA sequences.

Keywords DNA–protein binding · Deep learning · Bidirectional long short-term memory · Convolutional neural networks

Abbreviations

Acc	Accuracy
AUC	The area under the ROC curve
BLSTM	Bidirectional long short-term memory
BP	Back-propagation algorithm
CNN	Convolutional neural network
ENCODE	The Encyclopedia of DNA elements
FN	The number of false negative
FP	The number of false positive
GPU	Graphical processing units
MCC	Mathews correlation coefficient
PFM	Positional frequency matrix
Pre	Precision
PSSM	Position specific scoring matrix
ROC	Receiver operating characteristic
Sen	Sensitivity
Spe	Specificity
TN	The number of true negatives
TP	The number of true positive
TFs	Transcription factors
TFBS	Transcription factor binding site

✉ Shaojie Qiao
sjqiao@cuit.edu.cn

[1] School of Computer Science, Chengdu University of Information Technology, Chengdu 610225, China

[2] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China

[3] School of Software Engineering, Chengdu University of Information Technology, Chengdu 610225, China

[4] College of Chemistry, Sichuan University, Chengdu 610064, China

1 Introduction

Accurately modeling the specificity of the transcription factors sequence is an essential problem in understanding the function and evolution of the genome [1–5]. TF is a protein that can bind to DNA sequence and regulate gene expression. The transcription factor binding sites are a subset of DNA binding sites. These sites can be defined as short segments of DNA that are specifically bound by one or more proteins with various functions. Particularly, the characterization of binding affinity of TFs to the DNA sequence determines the relative expression of genes downstream of the transcription factor binding sites (TFBS). The mechanism by which TFs select specific binding regions is complex and there are a large number of DNA–protein binding sites to be determined at different levels.

With the high-throughput technologies developing, such as ChIP-seq [6], ChIP-exo [7] and ChIP-nexus [8], a huge volume of experiments verified the TFs binding sites. However, they are time-consuming and expensive. Fortunately, these experimental data can serve as training data for machine learning models to learn the binding patterns of TF. Many computational approaches have been proposed to predict DNA–protein binding [9–12]. For example, Cirillo et al. [9] proposed PAnDA approach to predict DNA–protein binding with human transcription factors by using gene expression profiles, protein–protein interaction and recognition motifs. Zhang et al. [10] proposed an approach named DiseMLA to discover TFBS motifs on high-throughput dataset, which aims to optimize the phase of motif searching with a more comprehensive criterion. Zhu et al. [11] presented LSUE for inferring DNA–protein binding from new ChIP-seq datasets, which mainly utilize the local correlations between available datasets. Schmidt et al. [12] presented a framework, namely TEPIC2, allowing for a fast, accurate and versatile prediction and analyzing DNA–protein binding from epigenetic data.

Recently, deep learning technology has shown the capability of improving discriminating ability compared with other machine learning methods [13, 14], and has been widely applied in bioinformatics [15, 16], i.e., protein structure prediction [17], gene expression regulation [18, 19] and protein classification [20]. The convolutional neural network (CNN) has successfully predicted the DNA–protein binding [21–24]. These methods not only outperform other existing methods in terms of prediction accuracy, but also can easily extract binding motifs directly from the learned parameters of CNN. For example, DeepBind [21] is known to outperform the state-of-the-art experimental and computational methods to identify the binding preference of DNA-binding and RNA-binding proteins, which is a convolutional neural network trained on a large amount of data from high-throughput experiments. DeepSEA [22] also trains a CNN framework to predict the noncoding-variant effects from DNA sequences. Zeng et al. [23] proposed a systematic exploration of CNN architectures to predict DNA sequence binding in 690 transcription factor ChIP-seq experiments from the Encyclopedia of DNA Elements (ENCODE) project [25]. Cao et al. [24] introduced some tricks of CNN to improve the performance of DNA sequence related prediction tasks and took the DNA–protein binding as an illustrative task for demonstration. Fast convolution on the graphic processing unit (GPU) allows CNN to be trained on large-scale datasets. Wang et al. [26] proposed a specific study on the relationship between generalization and uncertainty by incorporating complexity of classification, which concludes that the generalization ability of a classifier is statistically becoming better with the increase of uncertainty when the complexity of the classification problem is relatively high. Wang et al. [27] investigated the multiple-instance active learning (MIAL) by incorporating diversity and informativeness. Two diversity criteria are proposed for MIAL by utilizing a support vector machine based MIL classifier. However, these techniques cannot indicate the dependency information of DNA sequences in the framework of CNN. In addition, these methods are not accurate enough in predicting DNA–protein binding from DNA sequences.

In this study, we focus on exploring the method of classifying whether a DNA segment binds to any TF. Therefore, we propose a computational prediction approach for DNA–protein binding based on BLSTM [28] and CNN, we call it DeepSite, to solve the aforementioned disadvantages of the existing methods. Based on DeepSite model, both the long as well as short dependency information of DNA sequences can be captured by mining the information from every mediate hidden value of BLSTM and CNN. The experimental results on the benchmark datasets show that DeepSite outperforms other existing deep learning methods. DeepSite approach can predict DNA-binding sites with 87.12% sensitivity, 91.06% specificity, 89.19% accuracy and 0.783 *MCC* when tested on the dataset used in 690 Chip-seq experiments. When compared with the CNN model, our method predicts DNA-binding sites with a 5.28%, 8.35%, 6.89% and 0.138 improvement in sensitively, specificity, accuracy and *MCC* value, respectively.

The original contributions of the proposed model are threefold: (1) we introduce BLSTM layer in the DeepSite algorithm to capture the long and short dependency information of DNA sequence, which improves its predictive performance; (2) a novel hybrid BLSTM and CNN framework for predicting DNA–protein binding from DNA sequences; (3) the experimental results demonstrate that the proposed approach performs better in identification of DNA–protein binding in DNA sequence.

2 Materials and methods

In this study, we present a deep learning-based approach, DeepSite (Fig. 1), to predict TBFS in DNA sequence by integrating a BLSTM and a CNN. We first describe the problem of transcription factor binding site by deep learning method. Then, we introduce the ChIP-seq experiments dataset from ENCODE, which is used to train and evaluate DeepSite. Next, we give the technical details about two different deep neural networks, BLSTM and CNN. Finally, we describe the proposed method DeepSite and how to implement in detail.

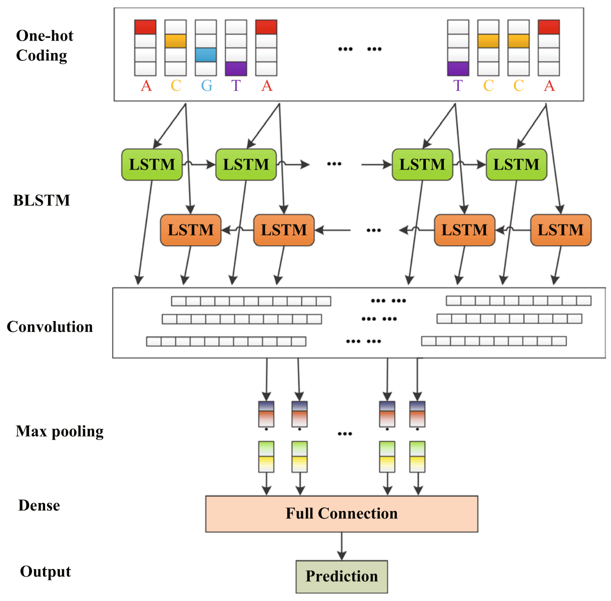

Fig. 1 The working mechanism of DeepSite model

2.1 Problem statement

This study focus on discovering the DNA–protein binding in DNA sequence, and the task of DNA–protein binding can be viewed as a binary sequence classification problem. The problem can be formulized as: as input, the training set is represented by $\{X^{(i)}, y^{(i)}\}_{i=1}^{n}$, where $X^{(i)}$ is a matrix, of dimension $4 \times N$, and N is the length of a DNA sequence (101 base pairs in our experiments). Each base pair in the sequence is represented 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]. This matrix is called Positional frequency matrix (PFM), which has four rows corresponding to each channel of genetic alphabet, namely $\{A, T, C, G\}$. Our labels, $y^{(i)}$ can be a scalar or vector, depending on the number of transcription factor binding sites being studied. Nonetheless, the number of dimension is equal to the classification tasks, and each element of $y^{(i)}$ is a binary label in the standard space $\{0, 1\}$. The goal is to accurately predict the label in the testing data, that is, to accurately predict whether a transcription factor combined with a given DNA sequence.

2.2 Dataset

As was performed in Alipanahi [21], Zhou [22] and Zeng [23], we obtain 690 ChIP-seq experiments from ENCODE[1]. We use the similar DNA sequence data by Zeng [23], the positive dataset consists of the centering 101 base pair region of each ChIP-seq peak, and the negative dataset consists of shuffled positive sequences with matching dinucleotide composition.

We generate the dataset based on the 690 ChIP-seq experimental data. In this study, we focus on the task of classifying whether a DNA segment binds to any TF. All the training data are combined into a whole dataset, the number of DNA sequences in the training set is 2,725,808, and the number of DNA sequences in the testing set is 255,700. In order to reduce the runtime of DeepSite, we firstly use 10% of training set and testing set to evaluate the performance. Finally, all the datatsets are used to assess the performance of DeepSite.

2.3 Bidirectional LSTM networks

Compared with traditional RNN, LSTM shows the ability to increase the dependence on long-distance evolution. Zhu et al. [29] used the traditional RNN to solve protein–protein network problems. One explanation may be their different processing of protein sequence data. Given a sequence, the tradition RNN, from $t = 1$ to n, works iteratively by Eqs. (1) and (2) to calculate the hidden vector sequence $h = (h_1, h_2, \dots, h_n)$ and outputs a vector sequence $y = (y_1, y_2, \dots, y_n)$.

$$h_t = f(W_{xh} * x_t + W_{hh} * h_{t-1} + b_h) \tag{1}$$

$$y_t = g(W_{hy} * h_t + b_y) \tag{2}$$

where $x = (x_1, x_2, \dots, x_n)$ is the input vector, t represents the index of input, output and hidden vectors, W is a weight matrix that is computed in the phase of training, b_* is the offset vector, and $f()$ and $g()$ denote the activation function.

LSTM is a special type of RNN and is well suitable for capturing the long and short dependency information in sequence [30]. A memory mechanism is applied in LSTM to replace the hidden function in the traditional RNN. The commonly-used LSTM unit consists of a memory cell, a forget gate, an input gate and an output gate, which is designed to enhance the ability of LSTM to model long-range dependence. LSTM memory cell is given in the following equations:

$$f_t = \sigma(W_{xf} * x_t + W_{hf} * h_{t-1} + b_f) \tag{3}$$

$$i_t = \sigma(W_{xi} * x_t + W_{hi} * h_{t-1} + b_i) \tag{4}$$

$$c_t = f_t \bigotimes c_{t-1} + i_t \bigotimes tanh(W_{xc} * x_t + W_{hc} * h_{t-1} + b_c) \tag{5}$$

$$o_t = \sigma(W_{xo} * x_t + W_{ho} * h_{t-1} + b_o) \tag{6}$$

$$h_t = o_t \bigotimes tanh(c_t) \tag{7}$$

[1] http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeAwgTfbsUniform/

where σ is the logistic Sigmoid function, *tanh* is a function to push the values to be between -1 and 1, f, i, c, o represent the forget gate, input gate, cell vectors and output gate, respectively, which are specified to be the same value as given in the hidden vector h, W_{xf} is the input-forget gate matrix, and W_{hf} is the hidden-forget gate matrix. The index t refers to the time step. $\otimes$ represents the vector product. It is worthwhile to note that the initial values of $c_0 = 0$ and $h_0 = 0$.

In the phase of sequence tagging, we have access to both past and future input features for a given time, so we can use a BLSTM as proposed in [28]. By doing so, we can efficiently make use of past features and future features within a specific time interval. The back-propagation is used to train BLSTM. In this study, we apply the forward and backward LSTM in the entire DNA sequence in order to capture long-term dependent relationship of DNA sequence. The hidden states only need to be set to 0 at the beginning of each sequence. In particular, we make a batch implementation that can handle multiple sentences at the same time.

2.4 Convolutional neural networks

CNN is a well-known deep learning framework, which has been widely applied in image recognition [31], speech recognition [32], computer vision [33], natural language processing [34], bioinformatics [21, 22] and other artificial intelligence research fields [35, 36]. Wang et al. [37] investigated essential relationships between generalization capabilities and fuzziness of fuzzy classifiers. The study makes a claim and offers sound evidence behind the observation that higher fuzziness of a fuzzy classifier may imply better generalization aspects of the classifier. The components of CNN include convolutional, pooling and fully connected layers. The convolutional layer is proposed to extract and represent the local information of original features through several feature maps and kernels. The pooling layer is employed to compress the resolution of the feature maps to achieve spatial invariance. After several convolution and pooling operations, there may be one or more fully connected layers to perform advanced reasoning. The output of the last fully connected layer transfer to an output layer. For a classifier or regression task, softmax regression is commonly-used because it can produce a well-formed probability distribution corresponding to the outputs.

2.5 The proposed model

The proposed model is introduced in this section, including the structure of the networks and its learning algorithm. Adam algorithm [38] is used to update the parameters. We used a bidirectional LSTM structure to deal with the order and reverse order dependency information in the DNA sequence. The network structure and the proposed algorithm are implemented based on Keras library. All of them are conducted on graphical processing units (GPU) to accelerate the training time.

We combine a BLSTM network and a CNN network to build a BLSTM-CNN model, which is shown in Fig. 1. This framework can efficiently characterize a possibly highly-complex order in DNA sequence via BLSTM layer and to generate filters that generalize sequence patterns via CNN and max pooling layers. With this neural network, both the long and short dependency information of DNA sequence can be captured by tapping the information from every mediate hidden value of BLSTM and CNN.

As shown in Fig. 1, the first input layer uses one-hot coding to represent each input sequence as a 4-row binary matrix, and the length of each sequence is 101 base pair.

The second layer is a BLSTM layer where each LSTM block in the first layer will receive the input sequence extracted from the trace of interest on the DNA and encodes its own interpretation regarding the overall contributions of the past history into its hidden state. Then, this interpretation is propagated to the next LSTM blocks located above and to the right of itself. Once the last nucletide is observed, the last unrolled LSTM block makes the final decision on the goodness of the probe.

The third layer is a convolutional layer composed of different convolutional kernels with rectified linear units as the activation function. Each convolutional kernel works as a motif detector that scans the input matrices and produces different strengths of signals that are correlated to underlying sequence patterns. The vertical and horizontal dimensions in the convolution box are 1 and 24, respectively.

The fourth layer is a max pooling layer that maximize the output signals of each convolutional kernel along the whole sequence.

The fifth layer is a fully connected layer with rectified linear units as activation unit. The size of fully connected layer is 32, the same as Zeng [23].

The last layer performs a non-linear transformation with sigmoid activation and produces a value between 0 and 1 that represents the probability of a binding preference of each probe.

2.6 Model parameters and training procedure

DeepSite is trained by using the standard back-propagation algorithm [39] and mini-batch gradient descent with the Adagrad [40] variation. Wang et al. [41] proposed a new deep learning approach to train multilayer feed-forward neural networks, which dose not need to iteratively tune the weights. It uses restricted Boltzman machine as the layer-wise training and use the generalized inverse of a matrix as the supervised fine-tuning. Dropout [42] and the phase of early stopping are used for regularization and model selection. Detailed parameter configurations are given in the next section.

All models use a genetic SGD forward and backward training method in this study. We choose the most complicated and best model BLSTM-CNN to display the performance of training. In experiments, the training dataset is divided into batches and one batch is processed at a time. Each batch contains a series of sentences which is determined by the parameter of batch size. As recommended by Alipanahi [21], the batch size is specified to 64. The weights and bias are set to the default values in Keras. Each model is optimized by training for 100 epochs. The learning rate changes from 0.001 to 0.008. The dropout ratio is specified to 0.1, 0.3, and 0.5, respectively. The number of cells w.r.t. BLSTM changes from 32 to 400 and the default value is 32. The filter number of CNN changes from 32 to 400 and the default value is 32.

All experiments are conducted by the Python library Keras, running on a machine with 24 Xeon processor and 256GB of memory and 1 Nvidia Tesla K40C GPU.

3 Results and discussions

In order to examine the performance of the proposed DeepSite, experiments based on ChIP-seq from ENCODE benchmarks against three selected state-of-the-art algorithms are performed. In the following, the evaluation matric are outlined first. Then the parameter tuning was discussed, including learning rate, dropout ratio, number of cells in LSTM and number of convolution kernels in CNN. Finally, the performance comparison was employed other deep learning method, three existing predictors and other different datasets.

3.1 Evaluation metrics

In this study, five evaluation measurements are used in this study, that is, sensitivity (*Sen*), specificity (*Spe*), accuracy (*Acc*), precision (*Pre*) and the Mathew's correlation coefficient (*MCC*) are employed to evaluate predictive capability. They are calculated by the following equations:

$$Sen = \frac{TP}{TP + FN} \tag{8}$$

$$Spe = \frac{TN}{TN + FP} \tag{9}$$

$$Acc = \frac{TP + TN}{TP + FN + TN + FP} \tag{10}$$

$$Pre = \frac{TP}{TP + FP} \tag{11}$$

$$MCC = \frac{TP \cdot TN - FN \cdot FP}{\sqrt{(TP + FN)(TP + FP)(TN + FN)(TN + FP)}} \tag{12}$$

Table 1 Performance of DeepSite model with different learning rates

Learning rate	*Sen* (%)	*Acc* (%)	*Pre* (%)	*MCC*
0.001	72.23	79.85	83.05	0.598
0.002	71.41	79.31	82.72	0.582
0.003	70.28	79.18	83.19	0.586
0.004	69.06	78.26	81.08	0.575
0.005	67.30	78.20	83.54	0.569
0.006	67.01	77.50	85.47	0.560
0.007	66.87	77.36	82.05	0.551
0.008	64.49	76.43	82.04	0.534

where *TP* is the number of true positives, *TN* is the number of true negatives, *FP* is the number of false positives, *FN* is the number of false negatives, *P* is the number of positives, and *N* is the number of negatives.

However, these five measurements are threshold dependent. Hence, the method chosen for reporting these evaluation measurements is critical for making a fair comparison between different predictors. In this study, the area under the receiver operating characteristic (ROC) curve (*AUC*), which is threshold-independent and increases in direct proportion to the overall prediction performance, is used to evaluate the prediction performance.

3.2 Parameter tuning

3.2.1 Selecting the learning rate

The hyper-parameters for TFBS task needed to be tuned in order to obtain optimal results. The learning rate is one of the most important hyper-parameters to be tuned for training deep neural networks. If the learning rate is a little bit lower, the phase of training is more reliable, but the phase of optimization will cost much time because the update value of the loss function is small for each time optimization. If the learning rate is high, the phase of training may not converge or even diverge. A higher interval of learning rate may cause the optimizer skips the optimal value, which makes the optimization of loss function become worse. The range of learning rate is different for different datasets as well as parameter configuration. In this study, we observe different metrics when the learning rate changes from 0.001 to 0.008. The experimental results are given in Table 1 and Fig. 2.

From Table 1, we observe that when learning rate is set to 0.001, the proposed algorithm obtains the best values of all evaluation metrics. The values of *Sen*, *Acc*, *Pre* and *MCC* of are 72.23%, 79.85%, 83.05% and 0.598, respectively, when the learning rate is set to 0.001 which improves approximately 7.74%, 3.42%, 1.01% and 0.064 when the learning rate is set to 0.008. Figure 2 shows the performance of *AUC* when the learning rate changes from 0.001 to 0.008. We can see that: as

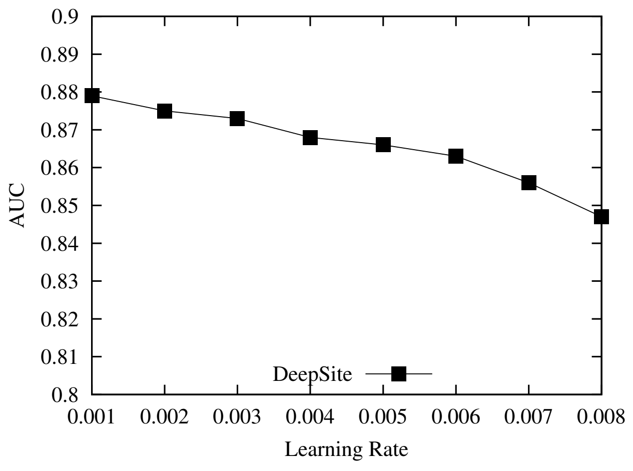

Fig. 2 *AUC* of DeepSite model with different learning rates

the learning rate increases gradually, the *AUC* of the predictor decreases drastically. By empirical studies, the learning rate is specified to 0.001 in the following experiments.

3.2.2 Selecting the dropout ratio

Overfitting is a common problem in deep neural network. Dropout is a technique for addressing this problem, which randomly set some intermediate values to zero in training the neural network [42]. To prevent the phenomenon of overfitting, we investigate whether the dropout method was a feasible strategy to improve training accuracy. Based on Fig. 3, as the dropout ratio increases, the *AUC* substantially grows, suggesting that adding dropout to the model may improve the robustness. A similar trend is also observed from the results in Table 2. The *MCC* is 0.706, 0.704 and 0.70 on 0.1, 0.3 and 0.5 dropout ratio, respectively. Therefore, the dropout ratio is chosen as 0.1 in the next model.

3.2.3 Selecting the number of cells in LSTM

In this section, we attempt to empirically demonstrate how to choose the number of cells in LSTM. We evaluate the *Sen*, *Spe*, *Acc*, *Pre*, *MCC* and *AUC* values on the training dataset by gradually varying the value from 32, 64, 128 to 400.

Figure 4 shows the performance of the metric of *AUC* w.r.t these four algorithms including LSTM, BLSTM, LSTM-CNN and DeepSite with different number of cells from 32 to 400. As shown in Fig. 4, the *AUC* of BLSTM improves significantly with the number of cells from 32 to 300. After that, the value of the *AUC* remain unchanged. For the LSTM, when the number of cells varies from 32 to 300, the *AUC* increase drastically. After that, the values of *AUC* keep stable. LSTM-CNN and DeepSite algorithm have almost the same trend in terms of *AUC* with different number of cells from 32 to 400, the *AUC* increase gradually. According to Fig. 4, we find that the value of *AUC* increases with the number of cells from 32 to 256. After that, the improvement of four methods is not obvious, even more cells have been used. This can be explained by the reason that these four methods have reached to the peak value of *AUC*. We can conclude that the best number of cells is 256 in this set of experiments.

Table 3 shows the values of *Sen*, *Spe*, *Acc*, *Pre* and *MCC* by specifying different values of the cell numbers. Experimental results show that our algorithm achieve 0.686, 0.691, 0.706, 0.713, 0.716, 0.724 and 0.721 for *MCC* on 32, 64, 128, 256, 300, 350 and 400 cells, respectively, which outperforms BLSTM with the gap of 0.089, 0.044, 0.039, 0.015, 0.017, 0.021 and 0.017 for *MCC* on 32, 64, 128, 256, 300, 350 and 400 cells, respectively. In order to facilitate comparison, we lastly specify the number of cells to 256 in four methods.

As we can see from Table 3, our proposed, DeepSite algorithm, achieve the best results in all metrics when the number of cells are specified to different values, e.g., for DeepSite, when the cell number equals 350, it obtains the best value of the *Sen* metric.

3.2.4 Selecting the number of convolution kernels in CNN

In this section, we discuss how to choose the number of convolutional kernels in CNN. We evaluate the values of *Sen*, *Spe*, *Acc*, *Pre*, *MCC* and *AUC* on the training dataset by gradually varying the value of the convolution kernels from 32, 64 to 400.

Figure 5 shows the variation curves of *AUC* under different number of convolution kernels. We can observe that the value of *AUC* increases with the number of convolution kernels and DeepSite model outperforms CNN. Specifically, the *AUC* of CNN significantly improves with the number of convolution kernels from 32 to 300 and keeps stable with the number of convolution kernels from 300 to 400. In terms of DeepSite and LSTM-CNN models, they have the same trend

Table 2 Performance of the DeepSite model with different dropout ratios

	Dropout ratio	*Sen* (%)	*Spe* (%)	*Acc* (%)	*Pre* (%)	*MCC*
DeepSite	0.1	80.24	89.82	85.28	87.64	0.706
	0.3	81.90	88.26	85.25	86.26	0.704
	0.5	78.76	90.49	84.93	88.18	0.700

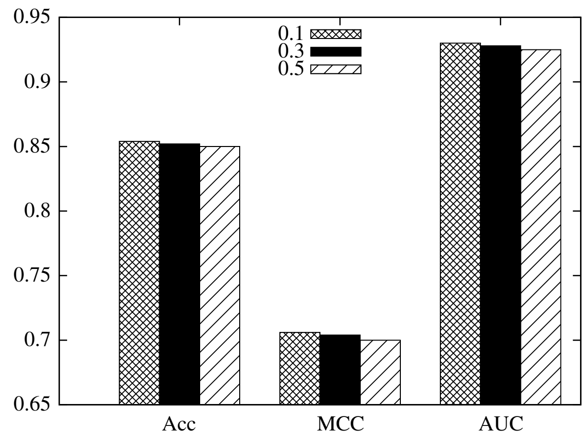

Fig. 3 The performance variation curves of *Acc*, *MCC* and *AUC* under different dropout ratio by DeepSite

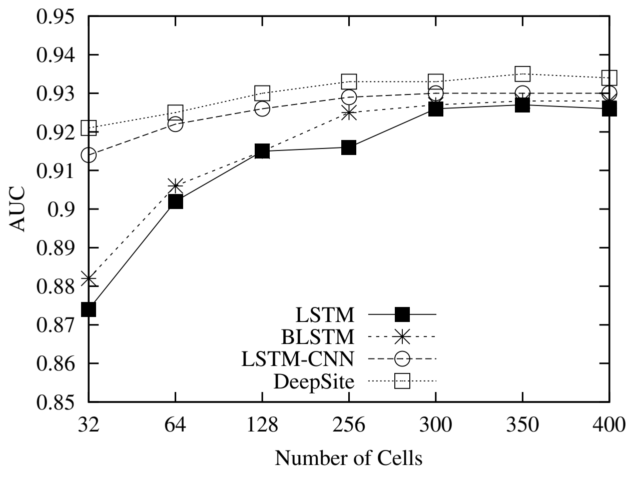

Fig. 4 Performance of variation curves of *AUC* of LSTM, BLSTM, LSTM-CNN and DeepSite models with different number of cells

of *AUC* with the number of convolution kernels changing. These two methods are increased slowly with the number of convolution kernels from 32 to 128. After that, the value of *AUC* is stable. The peak performance of DeepSite is better than LSTM-CNN and CNN models.

Table 4 demonstrates the mean value and standard deviation of *AUC* between DeepSite and other deep learning predictors with different number of convolution kernels. According to Table 4, we can see that the best value of the average *AUC* w.r.t DeepSite is higher than that of the CNN and LSTM-CNN algorithms. In addition, the standard deviation of DeepSite is lower than that of CNN and LSTM-CNN algorithms. The results demonstrate that DeepSite model is more accurate and stable at predicting the DNA–protein bindings.

Table 5 shows the measurements of *Sen*, *Spe*, *Acc*, *Pre* and *MCC* under different number of convolution kernels.

Table 3 Performance comparison of DeepSite and other deep learning predictors with different number of cells

	Cell numbers	LSTM	BLSTM	LSTM-CNN	DeepSite
Sen (%)	32	71.20	78.24	77.32	81.94
	64	78.95	77.69	78.98	75.43
	128	75.55	79.50	81.56	81.24
	256	78.73	81.50	81.22	82.09
	300	77.30	78.45	82.61	83.46
	350	82.96	82.98	83.57	84.94
	400	80.28	81.43	81.05	81.58
Spe (%)	32	87.26	81.42	89.06	86.55
	64	85.17	86.59	89.28	92.29
	128	90.65	86.86	87.98	89.82
	256	89.67	87.99	89.32	90.60
	300	87.40	90.71	89.21	89.57
	350	89.72	90.90	90.22	90.95
	400	89.71	89.38	89.90	89.14
Acc (%)	32	79.18	79.91	83.50	84.37
	64	82.22	82.37	84.40	84.30
	128	83.50	83.37	84.94	85.28
	256	83.96	84.92	85.48	85.62
	300	84.84	84.89	85.08	85.88
	350	85.29	85.12	85.67	86.20
	400	85.24	85.02	85.48	86.10
Pre (%)	32	84.52	79.13	86.42	84.58
	64	82.74	83.91	86.90	89.80
	128	87.92	84.49	85.93	87.64
	256	87.42	85.94	87.26	88.47
	300	87.28	88.37	88.07.	88.28
	350	87.54	88.62	88.12	88.79
	400	87.05	88.55	87.78	87.56
MCC	32	0.591	0.597	0.671	0.686
	64	0.643	0.647	0.688	0.691
	128	0.673	0.667	0.698	0.706
	256	0.692	0.698	0.709	0.713
	300	0.698	0.699	0.702	0.716
	350	0.704	0.703	0.713	0.724
	400	0.705	0.704	0.710	0.721

According to results of the model with the best-performing number of kernels, we can see that the proposed method achieves better performance than other classical models.

3.2.5 Peak performance of LSTM, BLSTM, LSTM-CNN and DeepSite models

Different methods have very different architectures, and we compare the peak performance of LSTM, BLSTM, LSTM-CNN and DeepSite models based on the results from Table 3. The peak performance results are shown in Table 6.

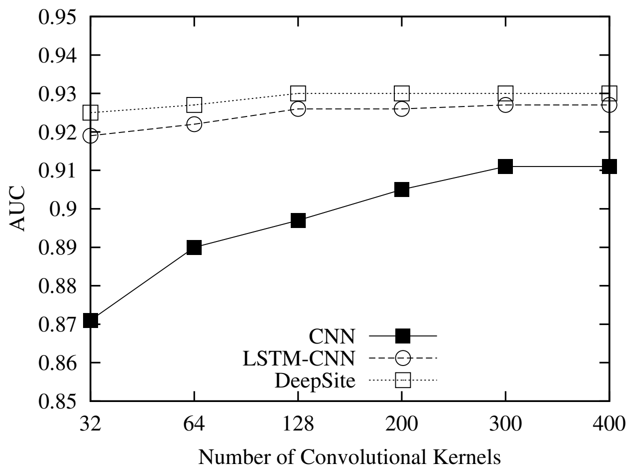

Fig. 5 Performance of variation curves of *AUC* under different number of convolution kernels

Table 4 *AUC* of DeepSite and other deep learning predictors with different convolution kernels

Convolution kernel numbers	CNN	LSTM-CNN	DeepSite
32	0.871 ± 0.0026	0.920 ± 0.0012	0.925 ± 0.0004
64	0.890 ± 0.0023	0.922 ± 0.0012	0.927 ± 0.0002
128	0.897 ± 0.0019	0.925 ± 0.0006	0.930 ± 0.0005
200	0.905 ± 0.0009	0.927 ± 0.0006	0.930 ± 0.0007
300	0.911 ± 0.0008	0.927 ± 0.0008	0.930 ± 0.0006
400	0.910 ± 0.0009	0.927 ± 0.0008	0.929 ± 0.0006

According to Table 6, the peak values of different experiments show that our method achieves 84.94% for *Sen* and 0.724 for *MCC* , respectively, works better than LSTM, BLSTM and BLSTM-CNN models in all cases. The results demonstrates that the combination of BLSTM and CNN obtains much better performance than other deep learning models. Furthermore, the results show the advantage of BLSTM which captures the long and short dependency information of DNA sequences.

3.3 Performance comparison

3.3.1 Performance comparison with different methods

In this section, the discriminative performances of these three deep learning methods, including CNN, BLSTM and BLSTM-CNN, will be investigated. Each method was evaluated on the same training dataset. The details of the parameters for different methods are shown in Table 7. These parameters are optimized by above analysis and then choose the best parameters for these methods.

Table 5 Performance comparison of DeepSite and other deep learning predictors with different number of convolution kernels

	Convolution kernel numbers	CNN	LSTM-CNN	DeepSite
Sen (%)	32	74.29	79.18	80.09
	64	73.56	79.67	80.55
	128	76.20	80.01	80.24
	200	78.88	79.13	80.88
	300	80.07	80.33	82.08
	400	77.18	79.12	81.19
Spe (%)	32	83.38	88.62	89.36
	64	87.51	86.96	89.04
	128	86.63	87.98	89.82
	200	85.89	88.70	89.96
	300	88.54	90.24	90.98
	400	85.90	89.18	88.98
Acc (%)	32	79.07	84.15	84.97
	64	80.90	84.45	85.02
	128	81.69	84.94	85.28
	200	82.57	84.97	85.30
	300	83.16	84.98	85.36
	400	83.14	84.97	85.28
Pre (%)	32	80.10	86.23	87.14
	64	84.13	84.93	86.87
	128	83.69	85.93	87.64
	200	83.45	86.57	88.76
	300	86.85	87.96	88.90
	400	83.66	86.98	87.32
MCC	32	0.580	0.683	0.699
	64	0.619	0.688	0.700
	128	0.633	0.698	0.706
	200	0.650	0.699	0.708
	300	0.663	0.704	0.706
	400	0.661	0.699	0.705

Table 6 Peak Performance of LSTM, BLSTM, LSTM-CNN and DeepSite models

	LSTM	BLSTM	BLSTM-CNN	DeepSite
Sen (%)	82.96	82.98	83.57	84.94
Spe (%)	90.65	90.71	90.22	92.29
Acc (%)	85.29	85.12	85.67	86.20
Pre (%)	87.54	88.62	88.12	89.80
MCC	0.705	0.704	0.713	0.724

Figure 6 illustrates the ROC curves of three deep learning methods on the same dataset.

As shown in Fig. 6, we find that the *AUC* of BLSTM-CNN is 0.932, which demonstrates improvement of

Table 7 Parameter setting of CNN, BLSTM, BLSTM-CNN models

Parameter	CNN	BLSTM	BLSTM-CNN
Learning rate	0.001	0.001	0.001
Dropout ratio	0.1	0.1	0.1
Kernel numbers	128	–	128
Cell numbers	–	256	256
Epochs	100	100	100
Batch size	64	64	64

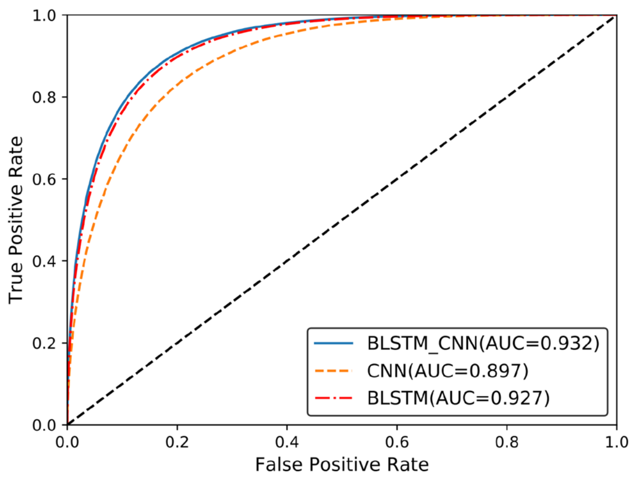

Fig. 6 Performance comparison of ROC curves for CNN, BLSTM and BLSTM-CNN on the same dataset

approximately 0.005 and 0.035, when compared with the BLSTM and CNN, respectively. From the comparison results between these three methods given in Fig. 6, we empirically demonstrate that these three deep learning methods are highly useful, and the combination of BLSTM and CNN, DeepSite, obtains the best ROC curve for effectively predicting DNA–protein binding. It indicated the advantage of BLSTM which captured the long and short dependency information of DNA sequence.

3.3.2 Performance comparison with existing predictors

In this section, we demonstrate the efficacy of the proposed DeepSite algorithm, by comparing it with the state-of-the-art method, including DeepBind [21], DeepSEA [22] and Zeng [23], on the same training and testing datasets, and the results are shown in Table 8.

We obtained the source code of DeepBind from the url: http://tools.genes.toronto.edu/deepbind/nbtcode/. We run DeepBind with the Docker Enterprise container platform so it can be run on different systems without the environment dependency problems.

Table 8 Performance comparison of DeepSite and classical predictors

Predictor	*Sen* (%)	*Acc* (%)	*Pre* (%)	*MCC*
DeepBind	69.26	78.21	81.96	0.566
DeepSEA	64.66	81.58	94.81	0.656
Zeng	76.20	81.69	83.69	0.633
DeepSite	80.09	85.62	88.47	0.713

DeepSEA model contains three convolution layers with 320, 480 and 960 kernels and two max pooling layers in alternating order to learn the motifs. On the third convolution layer, it has a fully connected layer and the last layer is the Sigmoid output layer.

We implemented the best model in Zeng using the training pipeline. The best model in Zeng has 128 convolution filters and the window size is 24, a global max pooling layer. In addition, their model always has a fully connected layer with 32 neurons after the global max pooling layer.

According to Table 8, we can see that DeepSite outperforms other classifiers in all metrics including the *MCC* which is an overall index for evaluating the quality of binary prediction. The *Sen*, *Acc*, *Pre* and *MCC* of DeepSite predictor are 80.09%, 85.62%, 88.47% and 0.713, respectively, which improves approximately 3.89%, 3.93%, 4.78% and 0.08 when compared with the Zeng predictor, respectively. As for DeepBind and DeepSEA models, tDeepSite also have an improvement of 0.147 and 0.057 in *MCC*, respectively. These results demonstrate that: by adding the recurrent connections, the performance of DeepSite algorithm can be significantly improved.

3.3.3 Performance comparison with different datasets

To further assess the performance of DeepSite, we conduct experiments on four different datasets with 10%, 30%, 50% and 100% of the size of data by using DeepSite and CNN. Figure 7 shows the performance variation curves of AUC under different cardinality of datasets.

From Fig. 7, we find that the value of *AUC* increases with the cardinality of data and the performance of DeepSite wins CNN in most cases. Table 9 gives the values of *Sen*, *Spe*, *Acc*, *Pre* and *MCC* under different number of data. The results show that the our method achieve 0.713, 0.765, 0.770 and 0.783 for *MCC* on 10%, 30%, 50%, and 100% of the size of data, respectively, performing better than the CNN model with 0.008, 0.116, 0.131, 0.138 on 10%, 30%, 50%, and 100% of the size of data. This may be explained by the fact that the 100% of the size of dataset has more training data and DeepSite can make good use of the large number of training instances to improve its performance.

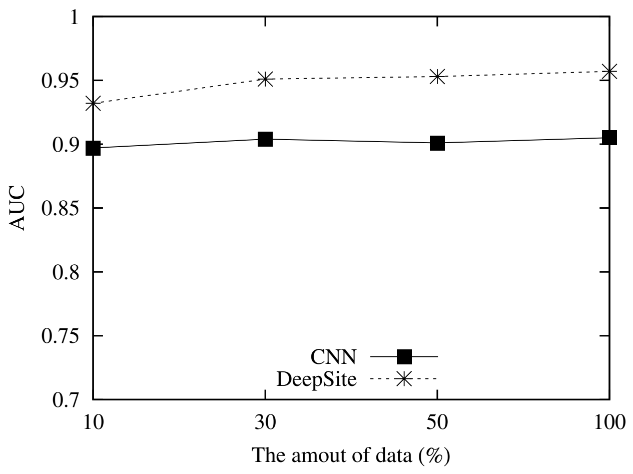

Fig. 7 *AUC* of CNN and DeepSite models under different sizes of data

Table 9 Performance comparison of DeepSite and CNN under different sizes of datasets

	Amount of data (%)	CNN	DeepSite
Sen (%)	10	76.20	80.09
	30	75.28	84.06
	50	76.33	87.30
	100	81.85	87.12
Spe (%)	10	86.63	90.60
	30	88.79	91.96
	50	87.06	89.69
	100	82.71	91.06
Acc (%)	10	81.69	85.62
	30	82.39	88.22
	50	81.98	88.56
	100	82.30	89.19
Pre (%)	10	83.69	88.47
	30	85.81	90.39
	50	84.15	88.40
	100	81.00	89.77
MCC	10	0.633	0.713
	30	0.649	0.765
	50	0.639	0.770
	100	0.645	0.783

4 Conclusions

In this study, we present a combined BLSTM and CNN framework to predict DNA–protein binding in DNA sequences, which is called DeepSite. DeepSite uses BLSTM to capture context dependency information of DNA subsequences, and then spreads it into the CNN layer to extract the discriminative features, and finally outputs these features to a full connection layer. Experimental results with a training dataset have demonstrated the efficacy of the proposed DeepSite. The DeepSite model proposed in this study can be applied to identify DNA–protein binding. For the ongoing work, we will further investigate the applicability of the proposed model to other types of molecules binding prediction problems, e.g., RNA-protein binding, which potentially can help scientists identify new DNA–protein binding sites in test sequences.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China under Grants nos. 61702058, 61772091, 61802035, 71701026; the China Postdoctoral Science Foundation funded project under Grant no. 2017M612948; the Scientific Research Foundation for Education Department of Sichuan Province under Grant no. 18ZA0098; the Sichuan Science and Technology Program under Grant nos. 2018JY0448, 2019YFG0106, 2019YFS0067, 2018GZ0307; the Natural Science Foundation of Guangxi under Grant no. 2018GXNSFDA138005; the Innovative Research Team Construction Plan in Universities of Sichuan Province under Grant no. 18TD0027; the Fund of Science and Technology Department of Guizhou Province under Grant no. J[2014]2134; the Scientific Research Foundation for Young Academic Leaders of Chengdu University of Information Technology under Grant nos. J201706, J201701; the Scientific Research Foundation for Advanced Talents of Chengdu University of Information Technology under Grant nos. KYTZ201717, KYTZ201715, KYTZ201750; Guangdong Key Laboratory Project under Grant no. 2017B030314073.

Compliance with ethical standards

Conflict of interest There is no conflict of interest.

References

1. Jolma A, Yan J, Whitington T, Toivonen J, Nitta KR, Rastas R, Morgunova E, Enge M, Taipale M, Wei G (2013) DNA-binding specificities of human transcription factors. Cell 152(1):327–339
2. Zhou TY, Shen N, Yang L, Abe N, Horton J, Mann RS, Bussemaker HJ, Gordân R, Rohs R (2015) Quantitative modeling of transcription factor binding specificities using DNA shape. Proc Natl Acad Sci 112(15):4654–4659
3. Slattery M, Zhou T, Yang L, Dantas AC, Gordan R, Rohs R (2014) Absence of a simple code: how transcription factors read the genome. Trends Biochem Sci 39(9):381–399
4. Zhang YQ, Cao XY, Zhong S (2016) Genemo: a search engine for web-based functional genomic data. Nucleic Acids Res 44(W1):W122–W127
5. Fan S, Huang K, Ai R, Wang M, Wang W (2016) Predicting CPG methylation levels by integrating infinium humanmethylation 450 beadchip array data. Genomics 107(4):132–137
6. Furey TS (2012) Chip-seq and beyond: new and improved methodologies to detect and characterize protein–DNA interactions. Nat Rev Genet 13(12):840–52
7. Wang L, Chen J, Wang C, Uusküläreimand L, Chen K, Medinarivera A, Young EJ, Zimmermann MT, Yan H, Sun Z (2014) Mace: model based analysis of chip-exo. Nucleic Acids Res 42(20):e156

8. He QY, Johnston J, Zeitlinger JL (2015) Chip-nexus: a novel chip-exo protocol for improved detection of in vivo transcription factor binding footprints. Nat Biotechnol 33(4):395–401
9. Cirillo D, Bottaorfila T, Tartaglia GG (2015) By the company they keep: interaction networks define the binding ability of transcription factors. Nucleic Acids Res 43(19):e125
10. Zhang HB, Lin Z, Huang DS (2016) Discmla: an efficient discriminative motif learning algorithm over high-throughput datasets. IEEE ACM Trans Comput Biol Bioinform 15(6):1810–1820
11. Zhu L, Guo WL, Lu CY, Huang DS (2017) Collaborative completion of transcription factor binding profiles via local sensitive unified embedding. IEEE Trans Nanobiosci 15(8):946–958
12. Schmidt F, Kern F, Ebert P, Baumgarten N, Schulz MH (2018) Tepic 2—an extended framework for transcription factor binding prediction and integrative epigenomic analysis. Bioinformatics 35(9):1608–1619
13. Huang DS (2004) A constructive approach for finding arbitrary roots of polynomials by neural networks. IEEE Trans Neural Netw 15(2):477–491
14. Zhang YQ, Zhang DL, Mi G, Ma DC, Li GB, Guo YZ, Li ML, Zhu M (2012) Using ensemble methods to deal with imbalanced data in predicting protein–protein interactions. Comput Biol Chem 36:36–41
15. Min S, Lee B, Yoon S (2017) Deep learning in bioinformatics. Brief Bioinform 18(5):851–869
16. Zhang YQ, Qiao SJ, Ji SJ, Zhou JL (2018) Ensemble-cnn: Predicting dna binding sites in protein sequences by an ensemble deep learning method. In: Proceedings of 2018 international conference on intelligent computing. Springer, Wuhan, China, pp 301–306
17. Spencer M, Eickholt J, Cheng JL (2015) A deep learning network approach to ab initio protein secondary structure prediction. IEEE ACM Trans Comput Biol Bioinform 12(1):103–112
18. Chen YF, Li Y, Narayan R, Subramanian A, Xie XH (2016) Gene expression inference with deep learning. Bioinformatics 32(12):1–8
19. Zhang Y, Qiao S, Ji S, Han N, Liu D, Zhou J (2019) Identification of DNA–protein binding sites by bootstrap multiple convolutional neural networks on sequence information. Eng Appl Artif Intell 79:58–66
20. Asgari E, Mofrad MRK (2015) Continuous distributed representation of biological sequences for deep proteomics and genomics. PLoS One 10(11):1–15
21. Alipanahi B, Delong A, Weirauch MT, Frey BJ (2015) Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. Nat Biotechnol 33(8):831–839
22. Zhou J, Troyanskaya OG (2015) Predicting effects of noncoding variants with deep learning-based sequence model. Nat Methods 12(10):931–934
23. Zeng H, Edwards MD, Liu G, Gifford DK (2016) Convolutional neural network architectures for predicting DNA–protein binding. Bioinformatics 32(12):i121–i127
24. Cao Z, Zhang SH (2018) Simple tricks of convolutional neural network architectures improve DNA–protein binding prediction. Bioinformatics 35(11):1837–1843
25. Harrow J, Frankish A, Gonzalez JM, Tapanari E, Diekhans M, Kokocinski F, Aken BL, Barrell D, Zadissa A, Searle S (2012) Gencode: the reference human genome annotation for the encode project. Genome Res 22(9):1760–1774
26. Wang X, Wang R, Chen X (2018) Discovering the relationship between generalization and uncertainty by incorporating complexity of classification. IEEE Trans Cybern 48(2):703–715
27. Wang R, Wang X, Kwong S, Chen X (2017) Incorporating diversity and informativeness in multiple-instance active learning. IEEE Trans Fuzzy Syst 25(6):1460–1475
28. Graves A, Mohamed AR, Hinton G (2013) Speech recognition with deep recurrent neural networks. In: IEEE international conference on acoustics, speech and signal processing. IEEE, Vancouver, BC, Canada, pp 6645–6649
29. Zhu L, Deng SP, Huang S (2015) A two-stage geometric method for pruning unreliable links in protein–protein networks. IEEE Trans Nanobiosci 14(5):528–534
30. Klaus G, Rupesh KS, Jan K, Bas RS, Jürgen S (2015) LSTM: a search space odyssey. IEEE Trans Neural Netw Learn Syst 28(10):2222–2232
31. Krizhevsky A, Sutskever T, Hinton G (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems 25: 26th annual conference on neural information processing systems. Lake Tahoe, Nevada, USA, pp 1097–1105
32. Abdel-Hamid O, Mohamed AR, Jiang H, Penn G (2012) Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. In: 2012 IEEE international conference on acoustics, speech and signal processing. IEEE, Kyoto, Japan, pp 4277–4280
33. Karpathy A, Toderici G, Shetty S, Leung T, Sukthankar R, Li FF (2014) Large-scale video classification with convolutional neural networks. In: 2014 IEEE conference on computer vision and pattern recognition. IEEE, Columbus, OH, USA, pp 1725–1732
34. Wang T, Wu DJ, Coates A, Ng AY (2012) End-to-end text recognition with convolutional neural networks. In: Proceedings of the 21st international conference on pattern recognition. IEEE, Tsukuba, Japan, pp 3304–3308
35. Cecotti H, Graser A (2011) Convolutional neural networks for p300 detection with application to brain–computer interfaces. IEEE Trans Pattern Anal Mach Intell 33(3):433–445
36. Ouyang WL, Wang XG, Zeng XY, Qiu S, Luo P, Tian YL, Li HS, Yang S, Wang Z, Loy CC (2015) Deepid-net: deformable deep convolutional neural networks for object detection. In: IEEE conference on computer vision and pattern recognition. IEEE, Boston, MA, USA, pp 2403–2412
37. Wang X, Xing H, Li Y, Hua Q, Dong C, Pedrycz W (2015) A study on relationship between generalization abilities and fuzziness of base classifiers in ensemble learning. IEEE Trans Fuzzy Syst 23(5):1638–1654
38. Kingma D, Ba J (2014) ADAM: a method for stochastic optimization. In: Proceedings of 3rd international conference on learning representations. San Diego, CA, USA, pp 1–15
39. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323(6088):533–536
40. Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res 12(7):257–269
41. Wang X, Zhang T, Wang R (2019) Non-iterative deep learning: incorporating restricted Boltzmann machine into multilayer random weight neural networks. IEEE Trans Syst Man Cybern Syst 49(7):1299–1380
42. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958