Mathematical Biosciences and Engineering

MBE, 19(8): 7719–7736.
DOI: 10.3934/mbe.2022362
Received: 11 March 2022
Revised: 03 May 2022
Accepted: 06 May 2022
Published: 24 May 2022

http://www.aimspress.com/journal/MBE

Research article

DNA-binding protein prediction based on deep transfer learning

Jun Yan[1], Tengsheng Jiang[1], Junkai Liu[1], Yaoyao Lu[1], Shixuan Guan[1], Haiou Li[1], Hongjie Wu[1,2,*] and Yijie Ding[3,*]

[1] College of Electronic and Information Engineering, Suzhou University of Science and Technology, Suzhou, China
[2] Suzhou Smart City Research Institute, Suzhou University of Science and Technology, Suzhou, China
[3] Yangtze Delta Region Institute (Quzhou), University of Electronic Science and Technology of China, Quzhou, China

* **Correspondence:** Email: wuxi_dyj@163.com, hongjie.wu@qq.com.

Abstract: The study of DNA binding proteins (DBPs) is of great importance in the biomedical field and plays a key role in this field. At present, many researchers are working on the prediction and detection of DBPs. Traditional DBP prediction mainly uses machine learning methods. Although these methods can obtain relatively high pre-diction accuracy, they consume large quantities of human effort and material resources. Transfer learning has certain advantages in dealing with such prediction problems. Therefore, in the present study, two features were extracted from a protein sequence, a transfer learning method was used, and two classical transfer learning algorithms were compared to transfer samples and construct data sets. In the final step, DBPs are detected by building a deep learning neural network model in a way that uses attention mechanisms.

Keywords: DNA-Binding protein; deep learning; transfer learning

1. Introduction

Protein is very important for the human body. Some of these proteins can interact with DNA and are called DNA-binding proteins (DBPs). These are very important for gene-related life activities. For example, in DNA replication and repair functions, origins of replication sites [1] is the location where genomic DNA replication begins, and is important for the study of the DNA replication process. In

transcription and regulatory functions, RNA is an important molecule in the cell. Messenger RNA passes genetic information to DNA and acts as a template for protein synthesis, while only 2% of RNA molecules in proteins act as templates, the rest being a molecule called MicroRNA, which plays an important regulatory role in biological processes. Identifying molecules of MicroRNA [2] helps to understand the whole regulatory process, while some other functions are single-stranded DNA binding and separation functions, chromatin formation functions and cell development functions [3,4]. In addition, research into drug target proteins [5,6] and DNA expression genetics are also quite popular, as drug target proteins are closely related to human diseases, while DNA expression genetics include DNA N4-methylcytosine [7,8], histone modification, RNA interference, etc. The main study in this paper is DNA binding proteins. Identification of DBPs can help us better understand how proteins interact with DNA, thus promoting the development of life science.

Although the traditional method based on biological experiments can obtain high-precision results, it needs large quantities of time and human effort. In addition, with the advent of the post-genome era, Web-lab methods cannot keep up with the growth rate of protein sequences. By contrast, computational approach reduces the resources and manpower required and enables simple and efficient identification of DBPs from many protein sequences. Thus, for the development of bioinformatics, the use of computational methods to predict DBPs is of great value.

In the past decade, machine learning based algorithms are already getting a lot of attention, and researchers have also proposed several research algorithms. In general, DNA-binding proteins can be identified by two computational methods, one based on structure and the other on sequence. Gao et al. [9] proposed a knowledge-based method called DBD-Hunter. This method uses protein structural alignment and statistical potential energy assessment to predict DBPs. Nimrod et al. [10] used the 3D structure of proteins to predict DBPs. They used a random forest classifier to determine whether a protein was a DBP based on features obtained from the protein's evolutionary profile. Zhao et al. [11] Identification of DBPs proteins using 3D structures generated based on HHblits [12]. However, structure-based approaches rely on predicted or natural 3D protein structures, and obtaining these structures is difficult. As a result, many sequence-based methods have been developed. Kumar et al. [13] developed a random forest approach called DNA-Prot to identify DBPs from protein sequences. Liu et al. [14] developed a predictor called iDNAPro-PseAAC, which relies only on protein sequence information. They applied PseAAC [15,16] to support vector machines to identify DBPs. Wei et al. [17] used the features extracted from the local PSE-PSSM (pseudo location-specific scoring matrix) in combination with a random forest classifier and to identify DBPs. Mishra et al. [18] proposed a method called StackDPPred, which uses features extracted from PSSM and residue-specific contact energy to help train a stacking-based machine learning method that can effectively predict DNA-binding proteins. Nanni et al. [19] in order to build an optimal and most general classification system for DNA-binding proteins, features were experimentally extracted from proteins and trained and evaluated in a separate support vector machine, while the matrix of proteins was fine-tuned using convolutional neural networks with different parameter settings, and the decisions were fused with the support vector machine using weights and rules for predicting DBPs. In recent years, deep learning has proven to be very effective in image and natural language processing. Therefore, researchers gradually began to apply deep learning in bioinformatics. Deep learning methods need only to input raw data and do not need to manually extract features, as does machine learning. For example, Qu et al. [20] used a combination of LSTM and CNN and extracted features from protein sequences to predict DBPs. Shadab et al. [21] proposed two methods, DeepDBP-ANN and DeepDBP-CNN, by using deep

learning methods, the first by generating a set of features through traditional neural network training and the second by means of pre-learned embedding and convolutional neural networks, both of which have fetched good results. Some other methods, such as DeepDRBP-2L [22], iDRBP_MMC [23], and PDBP-Fusion [24], also improved the predictive performance of DBP by using deep learning.

2. Methods

2.1. The framework of DBPs prediction based on deep transfer learning

In the experiments of this study, the main approach to prediction was to use a conjunction of transfer learning and deep learning. First, the transfer learning algorithm was used to extract the data set S, which was related to the target sample, but not completely distributed based on sample similarity. Then the sequence and PSSM [25] features of data set S were extracted, in a deep network with an attention mechanism, the features are input and trained.

In the deep learning part of this method, the sequence and PSSM features were entered into LSTM [26] and CNN [27] respectively. In subsequent improvements, ResNet [28] was used to replace CNN, and better results were obtained. The final prediction results of these two parts also need to go through the fully connected layer. Figure 1 shows an overall prediction framework, mainly based on the DBP [29,30] prediction framework of deep transfer learning.

2.2. Transfer learning used in this paper

Many machine learning and data mining algorithms can now achieve positive results, but this is based on data sets with the same distribution [31]. In practice, this is often not true. The performance of traditional machine learning is likely to degrade when the distribution of the datasets used is different. When researchers are interested in a data domain, it is very expensive to re-label new data, and the labeled data will become outdated over time. For example, the search data on a Web site will be updated every once in a while, and the labeled data will become outdated at that time [32].

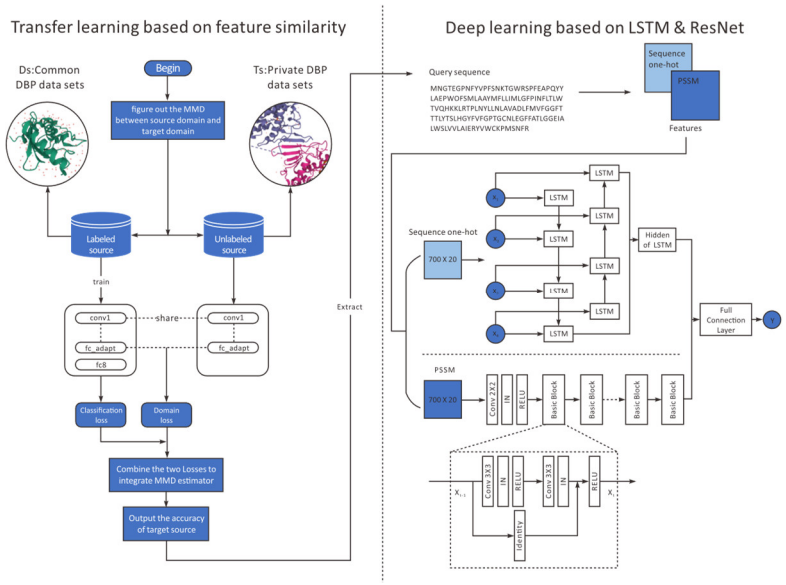


Figure 1. Flowchart of DBP prediction based on deep transfer learning.

The situation described above makes it necessary to train a powerful learning classifier from the relevant data domain. In processing related data domains and test data, the classifier can divide data with the same distribution. This is the principle of transfer learning.

In fact, machine transfer learning is closely related to human behavior. For example, once we learned to ride a bike when we were young, it was much easier to use an electric bike or a motorcycle [33]. After we learn the knowledge needed for riding a bi-cycle, when riding electric bikes and motorcycles, part of the technical knowledge can be shared. This means that we can quickly master the technology of riding electric bikes and motorcycles. This is human transfer learning, and machines can also master this learning mode. By using this model, machines can achieve faster learning and better results when faced with differently distributed data sets.

In transfer learning, there are generally two pairs of concepts, collectively referred to as two domains and two tasks, with two domains referring to the source and target domains and two tasks referring to the source and target tasks [33]. A domain can generally be thought of as a data set consisting mainly of a feature space and an edge probability distribution, which can be expressed as $D = \{x, P(X)\}$, where $x = \{x_1, ..., x_n\} \in x$. The two domains can be represented using Ds (source domain) and Dt (target domain) respectively. A task is simply the work to be performed and consists mainly of a label space and a target prediction function, usually denoted by $T = \{y, f(X)\}$. The two tasks are represented by Ts (source task) and Tt (target task) respectively.

With defined Ds and Ts and Dt and Ts, the main goal of using transfer learning is to acquire knowledge in Ds and Ts and finally learn the prediction function $ft(\bullet)$, but with the requirement that $Ds \neq Dt$ or $Ts \neq Tt$ [33].

Users of transfer learning must be clear about three things: 1) what transfer actually means; 2) how users make the transfer; and 3) when the transfer occurs [33].

To address the first point, "what to transfer", the first step is to clarify what kinds of problems need to be solved. The effect of transfer learning is better in classification and regression problems. As for the second point, "how to transfer", the only caveat is to choose simple and effective methods. There is no need to stick to a fixed algorithm because the algorithm is constantly changing and improving. As long as the algorithm has good results, it can be used. As for the third point, "when to transfer", although many studies have discussed the first two points, the third point is often more important. The aim of transfer learning is to optimize Tt. However, in real transfers, researchers often encounter a negative transfer effect, where the effect of transfer learning is not as good as that of learning without transfer. This requires a weighing and evaluation of how to avoid negative transfer.

There are four main approaches to achieving transfer learning: sample transfer, feature transfer, model transfer and relationship transfer. In sample-based transfer, the main task is to find data that are akin to the target domain in the Ds, and then to match the obtained data with the Dt data after corresponding operations. In feature-based transfer, the main task is to find similarities in the data from the two domains and to quantify these similarities using features. In model-based transfer, the trained model is directly migrated to the new domain. The advantage of this transfer is that deep learning can be used in conjunction with it. In the relational model, applying the network of logical relations learned in Ds to Dt is the main task.

There are many classical transfer-learning algorithms, which include learning to learn, knowledge transfer, lifelong learning, multi-task learning, meta learning, and context-sensitive learning. The experiments in [34] used TrAdaBoost [35], the pioneering work based on sample transfer in transfer learning. In addition, there are many ways deep learning can be integrated with transfer learning. Large

amounts of annotated data are needed by deep learning, but pre-training + fine-tuning a training model and parameter sharing require only a small fraction of labeled data. In the experiments in this study, deep domain confusion [36] (an algorithm combining deep learning and transfer learning) was also used to migrate source domain samples based on features.

2.3. Introduction to transfer learning algorithm

Considering that only slightly more than 1000 internationals standard DBP samples for deep learning training are available, too few data sets will have an impact on the effectiveness of deep learning. To increase the number of training samples, two migration algorithms, DDC and TrAdaBoost, were used in this study to migrate data sets, and appropriate data were selected for experimental testing and comparison.

2.3.1. Deep domain confusion

The general strategy for using deep learning with an insufficient number of data samples is to use fine-tuned networks, but testing with this paper's own samples did not work well. More layers may need to be fine-tuned to achieve better results, which requires many more samples. With little or no label data, there is no way to identify new samples through fine-tuning networks.

This study used the deep domain confusion approach. For each new sample, the dispersion of the data in the two domains is calculated and a layer is added between their adaptive layers and a domain is added to the confusion function. Next, the distance distribution is optimized using a convolutional neural network to reduce the distance between these scattered parts of the source and target domains. Finally, the problem was resolved un-der the condition of small samples of labels or no labels to identify problems. Table 1 shows the specific algorithm process.

The DDC algorithm in its essence involves looking for similar characteristics in two fields to maximize similar characteristics [36] by means of optimization loss. There are two main parts to the loss, the classification of the source domain is one part of the loss and the other part is used for the confusion loss of the two domains. To optimize the two losses, it is necessary to make the two domains close enough to the characteristics of the distribution [37]. In the end, the two domains will be indistinguishable.

In the DDC experiment, the data from the two domains are first mapped into a reproducing kernel Hilbert space, in which the average difference between the two domains is calculated as the distance between their distributions. According to Eq (2), the empirical estimate of MMD [38] is calculated, and the value of MMD obtained is used as the test statistic, which is entered into the deep network for training.

This network is an improved CNN architecture. Traditionally, for monitoring a source domain (labeled data), the loss is trained through the network. However, for a target domain without labels, data loss cannot be used for monitoring. Therefore, the model parameters used for source domain training are shared with the target domain, and an adaptive layer is added to one of the architecture layers.

It is well known that a deep network is gradually more proprietary from the bottom to the top. However, in order to share the characteristics of the two domains, there is no need to be proprietary, and therefore this layer is usually selected at the top. In deep domain obfuscation, the next step is to improve domain invariance as much as possible by choosing the adaptation layer as the seventh layer

of the deep network.

Finally, classification loss and domain loss are combined according to Eq (4). Regularization hyper-parameters should be set to make the target mainly weighted classification and avoid over-fitting. Finally, the model trained from the source do-main must also achieve good results in the target domain through deep network back-tracking.

Table 1. Deep Domain confusion Algorithm.

Algorithm 1: Deep domain confusion

Input: Labeled data Xs from the Ds X_S, Unlabeled data Xt from the Dt X_T

1: Given two distributions s and t, the MMD is defined as:

$$\mathrm{MMD}^2(s,t) = ||\phi||_H^{Sup} \leq 1^{||E\chi^S \sim S^{[\phi(\chi^S)]} - E\chi^t \sim t^{[\phi(\chi^t)]}||_H^2} \tag{1}$$

2: $X_S = \{\chi_i^S\}_{i=1}^M$ and $X_T = \{\chi_i^t\}_{i=1}^N$, figure out their MMD:

$$MMD^2(X_S, X_T) = ||\frac{1}{M}\sum_{i=1}^M \Phi(\chi_i^S) - \frac{1}{N}\sum_{i=1}^N \Phi(\chi_i^t)||_H^2 \tag{2}$$

3: Train X_S with Fine-tuned Alex Net

4: Use the same parameters to train X_S with DDC to get the classification loss:

$$L = L_C(X_L, y) \tag{3}$$

5: Integrate this MMD estimator:

$$L = L_C(X_L, y) + \lambda \sum_{\ell \in L} L_M(D_S^l, D_t^l) \tag{4}$$

6: Output the Average_Loss and Accuracy by using the MMD estimator to test X_T

2.3.2. TrAdaBoost

The distribution of training and test data is the same in traditional machine learning, but in practice, the two data sets are usually distributed differently. Old data may be out of date, and the cost of re-labeling new data to learn new data is huge. Often, old data have parts worth using.

To meet this need, the TrAdaBoost algorithm was used. Identically distributed training data refers to a small number of new data points, and differentially distributed training data refers to data points where the distribution of the training set differs from the distribution of the test set. With each iteration of the algorithm, the weights of the training data are thus adjusted. If a different distribution of training data points is incorrectly classified, then the target data is different from this sample, and the data weights in the sample will be reduced as a result. On the contrary, if a sample with the same distribution is misclassified, this will result in an increase in the weight of that sample and the algorithm will next focus on training a weak classifier, in line with the idea of the AdaBoost algorithm. Table 2 shows the specific algorithm process.

Table 2. TrAdaBoost Algorithm.

Algorithm 2: TrAdaBoost

Input: labeled training data sets Ta and Tb (merged data set T = Ta ∪ Tb), unlabeled data set S, Algorithm Learner, number of iterations N.

1: Initializing the weight vector w: w^1= (w_1^1,w_2^1, ...,w_m^1, ...,w_n^1), among them, $w_i^1 = \frac{1}{n}$

2: Set initialization error β: $\beta = 1/(1 + \sqrt{2\ln n / N})$
For=1,2,3, ..., N-1, N

3: Normalize P^t: $P^t = \frac{w^t}{\sum_{i=n}^{n} w_i^t}$

4: Change the weight of the data in T

5: Train the new weak classifier H_t: H_t: $Y = f(x)$

6: Compute the error of H_t on T_b as $\in_t$:

$$\in_t = \sum_{m}^{n} \frac{w_i^t \cdot |H_t(x_i) - c(x_i)|}{\sum_{i=m}^{n} w_i^t}$$

7: Update the error β_t: $\beta_t = \in_t/(1 - \in_t)$

8: To obtain the new weight vector:

$$w_i^{t+1} = \begin{cases} w_i^t \beta^{|H_t(x_i) - c(x_i)|}, 1 \le i \le m \\ w_i^t \beta_t^{-|H_t(x_i) - c(x_i)|}, m+1 \le i \le n \end{cases}$$

Output the trained classifier H.

Table 2 clearly shows that during each iteration, the data weights are updated. The weight of the last data point is multiplied by a value that is greater than 0 and less than 1, so that the weight of wrong samples in the auxiliary training samples will be reduced in the next training round, and the weight of the wrong samples will be in-creased. Finally, a strong classifier is obtained by voting with a 1/2 base learner.

2.4. Attention mechanism to improve prediction effect

To improve prediction accuracy, an attention mechanism was added to the model [39]. As information increases, the model also becomes complex. However, for the time being, the strength of the computational power remains an important constraint on the development of neural networks. Meanwhile, LSTM can only alleviate the problem of long-range dependence in recurrent neural networks (RNNs) to a certain extent, but its information memory capacity remains weak. Therefore, an attention mechanism was added to the model. In natural language processing tasks, attention mechanisms are often used, especially sequence to sequence tasks [40].

The main purpose of the attention mechanism is to let machines learn to have the same focus as humans. The nature of a query mapping to a series of key-value pairs can be referred to as an attention mechanism function. If attention needs to be computed, then three main steps are required. The first step is to calculate the similarity between each key from one query to another, and getting the similarity is equivalent to getting the weights. Then, the second step is to use a SoftMax [41] function to

normalize each weight. The final step is attention, which focuses on weighting and summing the weights and the corresponding key values.

Compared with CNN and RNN, the complexity and the number of parameters when using an attention mechanism are smaller, and therefore the computing power requirements are smaller. In addition, the attention mechanism can solve parallel computing problems that cannot be solved by RNN and can obtain extremely fast computing speeds. Therefore, adding an attention mechanism can greatly improve the efficiency and accuracy of predictions in this field.

2.5. Feature construction

At present, in the field of biology, the main methods for feature extraction are those based on feature representation and those based on sequence information [42], both of which are relatively common extraction methods, and there are many other methods. For DNA-binding protein feature extraction methods, there are also many methods to draw on. For example, PsePSSM, PSSM-DWT, PSSM-AB and other methods [43,44]. In this paper, the One-Hot coding and PSSM methods are mainly used for protein feature extraction.

2.5.1. One-hot

One-hot coding was used in this study to process the original protein sequence [45]. The one-hot code can be used to represent each residue in the protein sequence. Due to the fact that there are only 20 amino acids in nature, a 20-dimensional vector is used to represent all amino acid residues. A 20-dimensional vector consists of 19 zeros and 1 one, and then, a 20-dimensional vector is corresponded to a one-hot encoding, so a 20-dimensional vector can be used to represent an amino acid in the protein sequence index.

2.5.2. PSSM

The PSSM is an extremely important spectrum of information in the evolution of proteins [30]. It is generated by a scoring strategy with different amino acid occurrence frequencies at the same locus and their background frequency information in the results of protein multi-sequence alignment [46], and therefore contains information on protein evolution. It is the most commonly used evolutionary information spectrum in protein structure and function recognition. In a defined protein sequence, The $L \times 20$ matrix can be used to identify the PSSM matrix and the length of the protein sequence can be represented by L. When the matrix is a positive integer, it means that the amino acids at the corresponding site in the protein sequence are more likely to mutate to the 20 amino acids on the corresponding horizontal coordinate during the substitution process, and the larger the value, the higher the probability of substitution here. The opposite is true when the value is a negative integer, where a larger value indicates that it is less likely to change.

2.6. Training process

2.6.1. Neural network architecture

The paper for this research initially chose to use a neural network architecture combining LSTM and CNN, with ResNet added later as an improvement. LSTM, as an improved recurrent neural network, is not only able to solve the problem that RNNs cannot handle long-range dependencies, but also solves some common problems associated with gradients in neural networks. LSTM usually performs better than RNNs and hidden Markov models (HMM). In addition, LSTM has excellent performance in handwriting and speech recognition. In the method proposed here, LSTM is mainly used to deal with one-hot sequences.

In general, the depth in a neural network has a significant impact on model performance. Networks that need to extract more complex feature patterns need to increase the number of network layers, and when the appropriate number of layers is reached, theoretically better results can be obtained. However, it has been found that increasing the network depth may lead to the degradation problem: in other words, as the depth of the network increases and reaches a certain point, this will lead to saturation or even a decrease in accuracy. He et al. [47] proposed ResNets, which use residual learning to solve this degradation issue. ResNet uses a residual network structure, which allows the network layers to be added very deeply and ultimately results in better classification. Each output of the residual block can be represented by Eq (5).

$$x_t = f(x_{t-1} + F(x_{t-1}, W_t)) \tag{5}$$

The weight of the basic block of the t-th residual can be expressed as W_t. The function f is the activation function, and in the method proposed here, ReLU [48] is chosen as the activation function. In this study, ResNet was used to process the PSSM matrix. Use of ResNet can make the gradient flow more smoothly, which makes it possible to train the neural network with great depth.

2.6.2. Implementation of the method

The proposed method was trained under the Adam optimizer and implemented in Pytorch [49,50], along with a learning rate of 1e-3 and a cross-entropy loss for 40 epochs. Due to the limitations of GPU memory, when using LSTM and ResNet with relatively few layers, a batch size of 64 was used. For ResNet with very many layers, a batch size of 32 was used.

2.6.3. Evaluation indexes

The method has four main evaluation metrics, the first being Accuracy (ACC), the second being Matthew's Correlation Coefficient (MCC), the third being Sensitivity (SN) and the fourth being Specificity (Spec). The calculation formulas for the ACC, MCC, SN, and Spec indicators are shown as Eqs (6) to (9):

$$ACC = \frac{TN+TP}{TP+FP+FN+TN} \tag{6}$$

$$MCC = \frac{TN \times TP - FN \times FP}{\sqrt{(FN+TP)(FP+TN)(FP+TP)(FN+TN)}} \tag{7}$$

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

$$\text{Spec} = \frac{\text{TN}}{TN+FP} \tag{9}$$

Where TP refers to a positive case identified correctly, TN refers to a negative case identified correctly, FN refers to a negative case identified incorrectly and FP refers to a positive case identified incorrectly.

3. Experiments

3.1. Introduction to the dataset

In this work, baseline data sets (PDB186 and PDB1075) that recognize DNA binding proteins were used [14,51]. Each protein sequence in the benchmark data sets was derived from the PDB [52] (http://www.rcsb.org/pdb/home/home.do). The training set PDB1075 was originally extracted by Liu et al. in 2014 and the test set PDB186 was compiled by Lou in 2014. Each sequence was no more than 25% similar to other sequences in the dataset and did not contain irregular amino acids ('X'). The positive and negative sample counts for PDB1075 and PDB186 are shown in Table 3.

To carry out transfer learning, additional datasets PDB14189 and PDB2272 were used as source domains. These two datasets are from Du et al. [53] on the article published in 2019, while the two datasets also come from PDB Bank. The sequence similarity in PDB14189 was no more than 40%, the sequence similarity in PDB2272 was no more than 25%, and neither contained irregular amino acids ('X'). The positive and negative sample sizes for PDB14189 and PDB2272 are shown in Table 3.

Table 3. Information on the size of the data sets.

Data set	Total size	Negative size	Positive size
PDB1075	1075	525	550
PDB186	186	93	93
PDB14189	14,189	7129	7060
PDB2272	2272	1153	1119

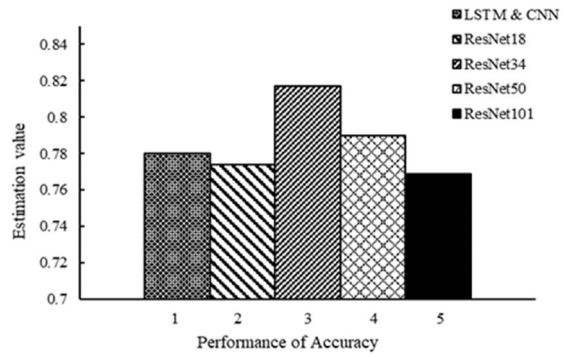
3.2. Experimental comparison of different network models for deep learning

The deep learning experiments reported in this paper mainly used a network model combining CNN and LSTM. To examine the effects of different depths and types of models on experimental performance, a comparison of the results using several common deep learning models is presented in Figure 2 and Table 4. The neural net-work models involved in the comparison included different kinds of ResNet. All these methods used the Adam optimizer with cross-entropy loss.

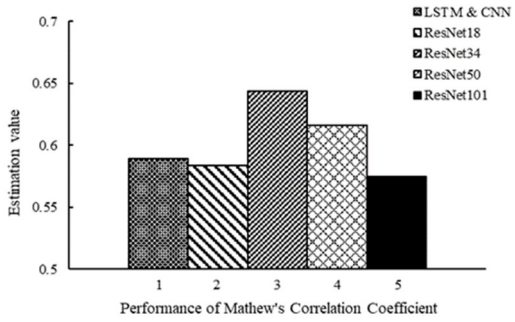
The data show that different depths and types of models affected the experimental results. When using ResNet18 (a network with 18 convolution and linear layers), the evaluation was slightly lower than the original. However, when using ResNet with more layers, the experimental performance was significantly improved.

Table 4. Performance comparisons between neural network models on PDB186.

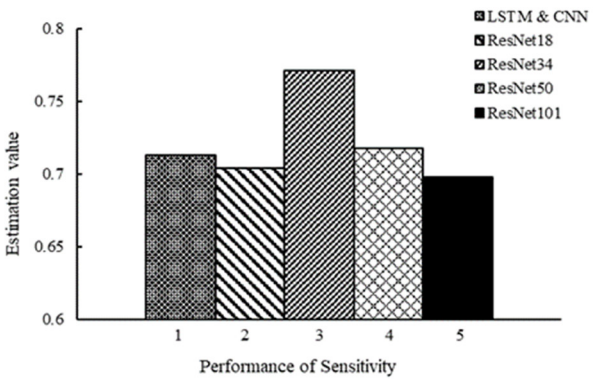
Methods	Model	ACC	MCC	SN	Spec
Deep learning	LSTM & CNN	0.780	0.589	0.713	0.906
	ResNet18	0.774	0.584	0.704	0.918
	ResNet34	0.817	0.644	0.771	0.883
	ResNet50	0.790	0.616	0.718	0.936
	ResNet101	0.769	0.575	0.698	0.917
TrAdaBoost + Deep learning	LSTM & CNN	0.871	0.751	0.822	0.937
	ResNet18	0.833	0.688	0.767	0.943
	ResNet34	0.812	0.637	0.759	0.892
	ResNet50	0.801	0.639	0.753	0.952
	ResNet101	0.785	0.587	0.730	0.873



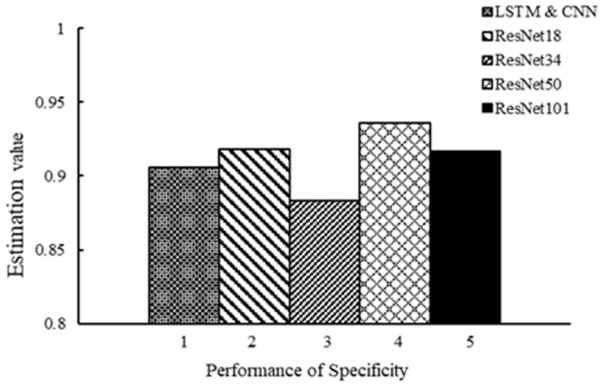
(a) Performance of Accuracy



(b) Performance of Mathew's Correlation Coefficient

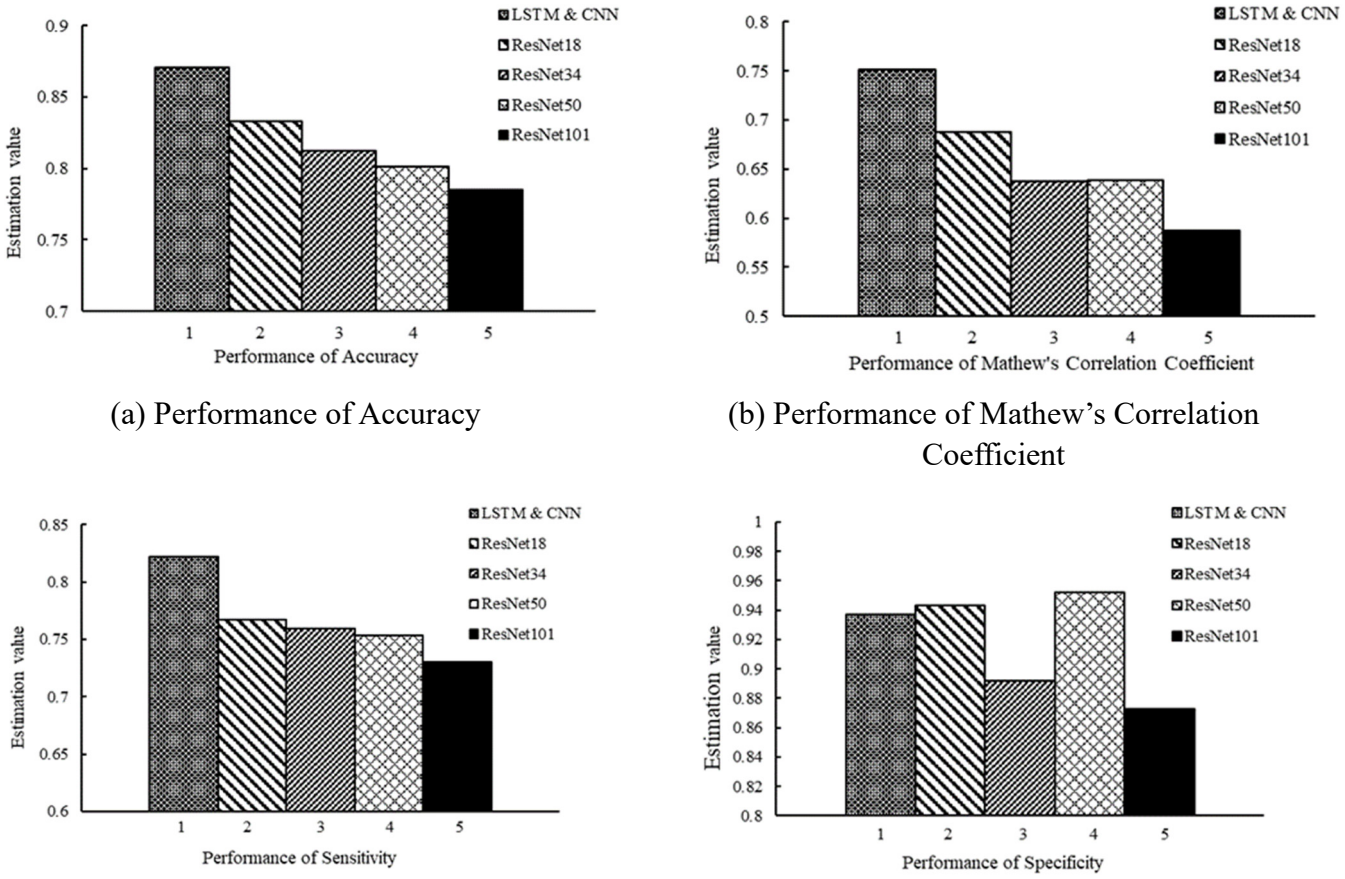


(c) Performance of Sensitivity



(d) Performance of Specificity

Figure 2. Performance comparison of different neural network models for deep learning on PDB186. (a), (b), (c) and (d) are the performance of Accuracy, MCC, Sensitivity and Specificity, respectively.



(a) Performance of Accuracy

(b) Performance of Mathew's Correlation Coefficient

(c) Performance of Sensitivity

(d) Performance of Specificity

Figure 3. Performance comparison of different neural network models for transfer learning on PDB186. (a), (b), (c) and (d) are the performance of Accuracy, MCC, Sensitivity and Specificity, respectively.

3.3. Experimental comparison of different network models for transfer learning

Similarly, the different model depths and types used above were applied to the PDB186 data set obtained through the transfer learning algorithm. Comparison through extensive experimental data, the performance obtained by adding the transfer data to the training sample was greatly improved. Specific experimental data are shown in Figure 3, and detailed results are given in Table 4.

3.3.1. DDC Experiment

In the experiments using DDC, datasets from the PDB14189 source domain and PDB2272 target domain were used, as transfer learning presupposes similarity between the two domains, which would otherwise lead to negative transfer.

According to the results of the DDC experiment, after the 20th training epoch, the supervised training accuracy of the data in the source domain was only 88.24%, whereas the accuracy of the classifier obtained when acting on the target domain was 71.55%. However, after the 100th training epoch, the classification accuracy of the source do-main data was 99.82%, and that of the target domain

data was 74.24%.

In this paper the classifier is considered good enough for both domains and the results of the 100th round of classification are used.

3.3.2. TrAdaBoost Experiment

Tradaboost as a transfer learning algorithm, we used PDB14189 as the source domain and PDB2272 as the target domain for migration learning. The features used in the migration learning process were PSSM matrices, after learning using a decision tree classifier, and 50 iterations were selected.

It is known that TrAdaBoost algorithm adjusts the weight of each data in each iteration. In the last iteration, the classification weight result of the classifier was recorded, and the average weight obtained was 7.047713017125943E-05. Through weight comparison, 12,541 migration results that met the classification were selected.

In the actual experiment, the DDC algorithm trains the network by supervised learning from the Ds and uses unsupervised learning from the Dt to train the network, and then fine-tunes the network model. The implementation cycle of this process is relatively long, and the trained model is too complex, leading to a model accuracy of only 74.24% on the test sample after 100 iterations, which is not ideal. The TrAdaBoost algorithm can quickly distinguish sample classification through sample-based migration, achieving faster execution and better effect.

3.4. Comparison of deep transfer learning methods and traditional methods

In traditional machine learning, the training and test sets are equally distributed and the datasets are already labelled, in which case they tend to show better performance. However, the number of samples available for the PDB186 dataset is relatively small and the data accuracy is not satisfactory. Therefore, the model was changed by adding transfer learning to determine the best model. The transfer learning model using LSTM and CNN demonstrated the best performance.

In order to achieve an objective and fair evaluation, experiments will be conducted using several other up-to-date methods and the results obtained will be compared with the methods in this paper. The proposed method is mainly covered in comparative experiments with other advanced methods. The results of various predictors on PDB186 are shown in Table 5 and Figure 4. In Table 5, the ACC, MCC, and Spec values from the proposed method on PDB186 exceed those from the other prediction methods. The experiments conclusively show that the approach used achieves excellent results in terms of performance and model robustness on the PDB186 independent dataset.

Experimental comparisons show that by using transfer learning, the number of training samples can be supplemented to some extent and fewer labelled samples can be used. This means that the method used here has better results than other DBP predictors.

Table 5. Comparison with other existing methods on PDB186.

Methods	ACC	MCC	SN	Spec
DNA-Prot	0.618	0.240	0.699	0.538
iDNAPro-PseAAC	0.715	0.442	0.828	0.602
Local-DPP	0.790	0.625	0.925	0.656
FKRR-MVSF	0.817	0.676	0.989	0.645
MSFBinder	0.796	0.616	0.936	0.656
DeepDRBP-2L	0.608	0.221	0.639	0.588
iDRBP_MMC	0.715	0.474	0.870	0.652
StackDPPred	0.8655	0.7363	0.9247	0.8064
IND1(PP(FUS)+eCNN)	0.8495	-	-	-
DeepDBP-CNN	0.8431	0.986	0.83	0.75
XGBoost	0.8548	0.713	0.903	0.806
Adilina's work	0.823	0.670	0.950	0.699
Our method	0.871	0.751	0.822	0.937

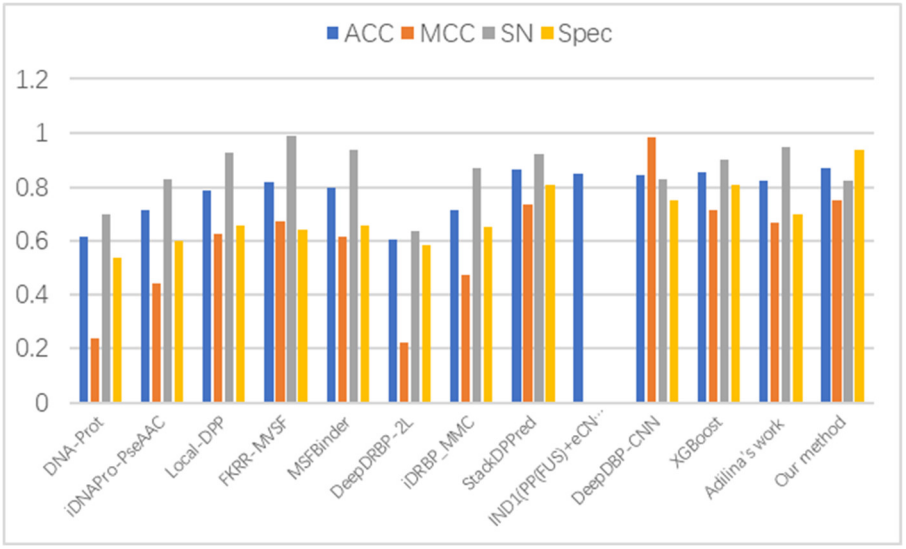


Figure 4. Comparison with other existing methods on PDB186.

4. Conclusions

The prediction of DNA-Binding Proteins has a long history of development in the field of structural biology and is currently a popular research topic, while the prediction of DNA-Binding Proteins continues to drive the development of the pharmaceutical industry. However, as far as the current traditional method is concerned, it is a serious drain on time and resources. In this paper, we excluded irregular amino acids ('X') in the samples, adopted transfer learning, and added attention mechanism to the deep neural network, which achieved better performance than traditional machine learning methods. But we didn't take into account the noise samples in the experiment. We will continue to investigate this in future research to continuously increase the predictive accuracy of the DBP.

Acknowledgments

This paper is supported by the National Natural Science Foundation of China (61902272,62073231,62176175,61876217,61902271), National Research Project (2020YFC2006602), Provincial Key Laboratory for Computer Information Processing Technology, Soochow University (KJS2166), Opening Topic Fund of Big Data Intelligent Engineering Laboratory of Jiangsu Province (SDGC2157), the Municipal Government of Quzhou (Grant Number 2020D003 and 2021D004).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. L. Wei, W. He, A. Malik, R. Su, L. Cui, B. Manavalan, Computational prediction and interpretation of cell-specific replication origin sites from multiple eukaryotes by exploiting stacking framework, *Briefings Bioinf.*, **22** (2021). https://doi.org/10.1093/bib/bbaa275
2. L. Wei, M. Liao, Y. Gao, R. Ji, Z. He, Q. Zou, Improved and promising identification of human MicroRNAs by incorporating a high-quality negative set, *IEEE/ACM Trans. Comput. Biol. Bioinf.*, **11** (2014), 192–201. https://doi.org/10.1109/TCBB.2013.146
3. D. H. Ohlendorf, W. F. Anderson, R. G. Fisher, Y. Takeda, B.W. Matthews, The molecular basis of DNA-protein recognition inferred from the structure of cro repressor, *Nature*, **298** (1982), 718–23. https://doi.org/10.1038/298718a0
4. W. H. Hudson, E. A. Ortlund, The structure, function and evolution of proteins that bind DNA and RNA, *Nat. Rev. Mol. Cell Biol.*, **15** (2014), 749–760. https://doi.org/10.1038/nrm3884
5. Y. Ding, J. Tang, F. Guo, Q. Zou, Identification of drug-target interactions via multiple kernel-based triple collaborative matrix factorization, *Briefings Bioinf.*, **23** (2022), bbab582. https://doi.org/10.1093/bib/bbab582
6. Y. Ding, J. Tang, F. Guo, Identification of drug–target interactions via dual laplacian regularized least squares with multiple kernel fusion, *Knowl.-Based Syst.*, **204** (2020), 106254. https://doi.org/10.1016/j.knosys.2020.106254
7. Y. Ding, P. Tiwari, Q. Zou, F. Guo, H. M. Pandey, C-loss based Higher-order Fuzzy Inference Systems for identifying DNA N4-methylcytosine Sites, *IEEE Trans. Fuzzy Syst.*, 2022. https://doi.org/10.1109/TFUZZ.2022.3159103
8. Y. Ding, W. He, J. Tang, Q. Zou, F. Guo, Laplacian regularized sparse representation based classifier for identifying DNA N4-methylcytosine Sites via L2,1/2-matrix norm, *IEEE/ACM Trans. Comput. Biol. Bioinf.*, 2021. https://doi.org/10.1109/TCBB.2021.3133309
9. M. Gao, J. Skolnick, DBD-Hunter: a knowledge-based method for the prediction of DNA-protein interactions, *Nucleic Acids Res.*, **36** (2008), 3978–3992. https://doi.org/10.1093/nar/gkn332
10. G. Nimrod, M. Schushan, A. Szilagyi, C. Leslie, N. Ben-Tal, iDBPs: a web server for the identification of DNA binding proteins, *Bioinformatics*, **26** (2010), 692–693. https://doi.org/10.1093/bioinformatics/btq019

11. H. Zhao, J. Wang, Y. Zhou, Y. Yang, Predicting DNA-binding proteins and binding residues by complex structure prediction and application to human proteome, *PLoS One*, (2014), e96694. https://doi.org/10.1371/journal.pone.0096694
12. M. Remmert, A. Biegert, A. Hauser, J. Soding, HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment, *Nat. Methods*, **9** (2011), 173–175. https://doi.org/10.1038/nmeth.1818
13. K. K. Kumar, G. Pugalenthi, P. N. Suganthan, DNA-Prot: identification of DNA binding proteins from protein sequence information using random forest, *J. Biomol. Struct. Dyn.*, **26** (2009), 679–686. https://doi.org/10.1080/07391102.2009.10507281
14. B. Liu, S. Wang, X. Wang, DNA binding protein identification by combining pseudo amino acid composition and profile-based protein representation, *Sci. Rep.*, **5** (2015), 15479. https://doi.org/10.1038/srep15479
15. K. C. Chou, Some remarks on protein attribute prediction and pseudo amino acid composition, *J. Theor. Biol.*, **273** (2011), 236–247. https://doi.org/10.1016/j.jtbi.2010.12.024
16. K. C. Chou, Prediction of protein cellular attributes using pseudo-amino acid composition, *Proteins*, **43** (2001), 246–255. https://doi.org/10.1002/prot.1035
17. L. Wei, J. Tang, Q. Zou, Local-DPP: an improved DNA-binding protein prediction method by exploring local evolutionary information, *Inf. Sci.*, **384** (2017), 135–144. https://doi.org/10.1016/j.ins.2016.06.026
18. A. Mishra, P. Pokhrel, M. T. Hoque, StackDPPred: a stacking based prediction of DNA-binding protein from sequence, *Bioinformatics*, **35** (2019), 433–441. https://doi.org/10.1093/bioinformatics/bty653
19. L. Nanni, S. Brahnam, Robust ensemble of handcrafted and learned approaches for DNA-binding proteins, *Appl. Comput. Inf.*, 2021. https://doi.org/10.1108/ACI-03-2021-0051
20. Y. H. Qu, H. Yu, X. J. Gong, J. H. Xu, H. S. Lee, On the prediction of DNA-binding proteins only from primary sequences: a deep learning approach, *PLoS One*, (2017), e0188129. https://doi.org/10.1371/journal.pone.0188129
21. S. Shadab, T. A. Khan, N. A. Neezi, S. Adilina, S. Shatabda, DeepDBP: deep neural networks for identification of DNA-binding proteins, *Inf. Med. Unlocked*, **19** (2020), 100318. https://doi.org/10.1016/j.imu.2020.100318
22. S. Ahmad, A. Sarai, PSSM-based prediction of DNA binding sites in proteins, *BMC Bioinf.*, **6** (2005), 33. https://doi.org/10.1186/1471-2105-6-33
23. J. Zhang, Q. Chen, B. Liu, DeepDRBP-2L: a new genome annotation predictor for identifying DNA-binding proteins and RNA-binding proteins using convolutional neural network and long short-term memory, *IEEE/ACM Trans. Comput. Biol. Bioinf.*, **18** (2021), 1451–1463. https://doi.org/10.1109/TCBB.2019.2952338
24. J. Zhang, Q. Chen, B. Liu, iDRBP_MMC: identifying DNA-binding proteins and RNA-binding proteins based on multi-label learning model and motif-based convolutional neural network, *J. Mol. Biol.*, **432** (2020), 5860–5875. https://doi.org/10.1016/j.jmb.2020.09.008
25. G. Li, X. Du, X. Li, L. Zou, G. Zhang, Z. Wu, Prediction of DNA binding proteins using local features and long-term dependencies with primary sequences based on deep learning, *PeerJ*, **9** (2021), e11262. https://doi.org/10.7717/peerj.11262

26. K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, J. Schmidhuber, LSTM: a search space odyssey, *IEEE Trans. Neural Networks Learn. Syst.*, **28** (2017), 2222–2232. https://doi.org/10.1109/TNNLS.2016.2582924
27. T. Roska, L. O. Chua, The CNN universal machine: an analogic array computer, *IEEE Trans. Circuits Syst. II*, **40** (1993), 163–173. https://doi.org/10.1109/82.222815
28. C. Szegedy, S. Ioffe, V. Vanhoucke, A. A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, (2017), 4278–4284. Available from: https://dl.acm.org/doi/10.5555/3298023.3298188.
29. B. Liu, J. Xu, X. Lan, R. Xu, J. Zhou, X. Wang, et al., iDNA-Prot|dis: identifying DNA-binding proteins by incorporating amino acid distance-pairs and reduced alphabet profile into the general pseudo amino acid composition, *PLoS One*, (2014), e106691. https://doi.org/10.1371/journal.pone.0106691
30. Y. Wang, Y. Ding, F. Guo, L. Wei, J. Tang, Improved detection of DNA-binding proteins via compression technology on PSSM information, *PLoS One*, (2017), e0185587. https://doi.org/10.1371/journal.pone.0185587
31. R. Caruana, A. Niculescu-Mizil, An empirical comparison of supervised learning algorithms, in *Proceedings of the 23rd International Conference on Machine Learning*, (2006), 161–168. https://doi.org/10.1145/1143844.1143865
32. K. Weiss, T. M. Khoshgoftaar, D. Wang, A survey of transfer learning, *J. Big Data*, **3** (2016), 9. https://doi.org/10.1186/s40537-016-0043-6
33. S. J. Pan, Q. Yang, A survey on transfer learning, *IEEE Trans. Knowl. Data Eng.*, **22** (2010), 1345–1359. https://doi.org/10.1109/TKDE.2009.191
34. M. Oquab, L. Bottou, I. Laptev, J. Sivic, Learning and transferring mid-level image representations using convolutional neural networks, in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, (2014), 1717–1724. https://doi.org/10.1109/CVPR.2014.222
35. W. Dai, Q. Yang, G. Xue, Y. Yu, Boosting for transfer learning, Machine Learning, in *Proceedings of the 24th International Conference on Machine Learning*, (2007), 193–200. https://doi.org/10.1145/1273496.1273521
36. S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, S. Bengio, Generating sentences from a continuous space, in *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, (2016), 10–21. https://doi.org/10.18653/v1/K16-1002
37. E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, T. Darrell, Deep domain confusion: Maximizing for domain invariance, preprint, arXiv:1412.3474.
38. H. Yan, Y. Ding, P. Li, Q. Wang, Y. Xu, W. Zuo, Mind the class weight bias: weighted maximum mean discrepancy for unsupervised domain adaptation, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (2017), 945–954. https://doi.org/10.1109/CVPR.2017.107
39. W. Qin, X. Cui, C. A. Yuan, X. Qin, L. Shang, Z. K. Huang, et al., Flower species recognition system combining object detection and attention mechanism, in *International Conference on Intelligent Computing, Springer*, 2019. https://doi.org/10.1007/978-3-030-26766-7_1

40. K. Cho, B. V. Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, et al., Learning phrase representations using RNN encoder-decoder for statistical machine translation, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (2014), 1724–1734. https://doi.org/10.3115/v1/D14-1179
41. T. Mikolov, S. Kombrink, L. Burget, J. Černocký, S. Khudanpur, Extensions of recurrent neural network language model, in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (2011), 5528–5531. https://doi.org/10.1109/ICASSP.2011.5947611
42. L. Wei, C. Zhou, H. Chen, J. Song, R. Su, ACPred-FL: a sequence-based predictor using effective feature representation to improve the prediction of anti-cancer peptides, *Bioinformatics*, **34** (2018), 4007–4016. https://doi.org/10.1093/bioinformatics/bty451
43. Y. Ding, J. Tang, F. Guo, Protein crystallization identification via fuzzy model on linear neighborhood representation, *IEEE/ACM Trans. Comput. Biol. Bioinf.*, **18** (2021), 1986–1995. https://doi.org/10.1109/TCBB.2019.2954826
44. Y. Ding, J. Tang, F. Guo, Human protein subcellular localization identification via fuzzy model on kernelized neighborhood representation, *Appl. Soft Comput.*, **96** (2020), 106596. https://doi.org/10.1016/j.asoc.2020.106596
45. S. K. Knapp, Accelerate FPGA macros with one-hot approach, *Electron. Des.*, 1990.
46. J. Soding, Protein homology detection by HMM-HMM comparison, *Bioinformatics*, **21** (2005), 951–960. https://doi.org/10.1093/bioinformatics/bti125
47. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (2016), 770–778. https://doi.org/10.1109/CVPR.2016.90
48. V. Nair, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, (2010), 807–814. Available from: https://dl.acm.org/doi/10.5555/3104322.3104425.
49. A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, et al., Automatic differentiation in pytorch, 2017. Available from: https://paperswithcode.com/paper/automatic-differentiation-in-pytorch.
50. D. P. Kingma, J. Ba, Adam: a method for stochastic optimization, *CoRR*, 2015. Available from: https://www.semanticscholar.org/paper/Adam%3A-A-Method-for-Stochastic-Optimization-Kingma-Ba/a6cb366736791bcccc5c8639de5a8f9636bf87e8.
51. W. Lou, X. Wang, F. Chen, Y. Chen, B. Jiang, H. Zhang, Sequence based prediction of DNA-binding proteins based on hybrid feature selection using random forest and Gaussian naive Bayes, *PLoS One*, (2014), e86703. https://doi.org/10.1371/journal.pone.0086703
52. P. W. Rose, A. Prlic, C. Bi, W. F. Bluhm, C. H. Christie, S. Dutta, et al., The RCSB Protein Data Bank: views of structural biology for basic and applied research and education, *Nucleic Acids Res.*, **43** (2015), D345–D356. https://doi.org/10.1093/nar/gku1214
53. X. Du, Y. Diao, H. Liu, S. Li, MsDBP: Exploring DNA-binding proteins by integrating multiscale sequence information via Chou's five-step rule, *J. Proteome Res.*, **18** (2019), 3119–3132. https://doi.org/10.1021/acs.jproteome.9b00226