A Deep Learning Framework for Predicting Protein Functions With Co-Occurrence of GO Terms

Min Li, Wenbo Shi, Fuhao Zhang, Min Zeng, and Yaohang Li

Abstract—The understanding of protein functions is critical to many biological problems such as the development of new drugs and new crops. To reduce the huge gap between the increase of protein sequences and annotations of protein functions, many methods have been proposed to deal with this problem. These methods use Gene Ontology (GO) to classify the functions of proteins and consider one GO term as a class label. However, they ignore the co-occurrence of GO terms that is helpful for protein function prediction. We propose a new deep learning model, named DeepPFP-CO, which uses Graph Convolutional Network (GCN) to explore and capture the co-occurrence of GO terms to improve the protein function prediction performance. In this way, we can further deduce the protein functions by fusing the predicted propensity of the center function and its co-occurrence functions. We use Fmax and AUPR to evaluate the performance of DeepPFP-CO and compare DeepPFP-CO with state-of-the-art methods such as DeepGOPlus and DeepGOA. The computational results show that DeepPFP-CO outperforms DeepGOPlus and other methods. Moreover, we further analyze our model at the protein level. The results have demonstrated that DeepPFP-CO improves the performance of protein function prediction. DeepPFP-CO is available at https://csuligroup.com/DeepPFP/.

Index Terms—Protein function prediction, deep learning, graph convolutional network, co-occurrence

1 Introduction

Proteins play an important role in biological processes, e.g., gene regulation, metabolic regulation, and body movement [1]. The discovery of protein functions helps understand biological activities at the molecular level and the treatment of diseases. However, less than 1% of proteins are annotated by the biological experiments in UniProt [2] until January, 2021. Biological experiments in vitro and in vivo are expensive and time-consuming. They are unable to reduce the gap between annotations of protein functions and the increasing number of protein sequence [3]. This motivates the development of computational methods.

In the past decades, many computational methods have been proposed for predicting protein functions. Many computational methods predict protein functions by using machine learning algorithms with protein se-quences [4], [5]. [6], [7], protein-protein interaction (PPI) networks [8], [9], [10], [11], [12], protein structures [13], [14], [15], [16], biomedical literature [17], [18], [19], and other features [20]. These methods annotate protein functions with Gene Ontology (GO) including more than 40,000 terms [21]. In addition, the prediction of protein functions is a multi-class, multi-label problem [22]. Thus, these methods need to learn an individual model for each GO term. This means that machine learning methods have to train tens of thousands of prediction models, one for each GO term. Recently, deep learning techniques are used to improve the performance of protein function prediction [23], [24], [25], [26], [27], [28], [29], [30], [31], [32] by learning individual model for Biological Processes Ontology (BPO), Molecular Functions Ontology (MFO), and Cellular Components Ontology (CCO), instead of learning a model for each GO term. DeepGO [25] combined a PPI network and sequence data for predicting protein functions. DeepGOA [28] improved the performance of protein function prediction with sequences, sub-sequences, and PPI network data. DeepGOPlus [24] is a sequence-based method that used Convolutional Neural Network (CNN) to capture sequence local features for protein function prediction. DeepText2GO [29] retrieved protein-related citations and extracted deep semantic information to predict protein functions. DeepFRI [32] used protein structure and sequence to predict protein functions. GONET [30] constructed the PPI network of human and mouse and extracted protein sequence features by CNN-RNN-Attention model to predict protein functions.

Although compared with the machine learning methods, the existing deep learning methods improve the performance of protein function prediction, these methods ignore

• *Min Li, Wenbo Shi, Fuhao Zhang, and Min Zeng are with the School of Computer Science and Engineering, Central South University, Changsha, Hunan 410017, China. E-mail: limin@mail.csu.edu.cn, {wbshi_cs, fhzhang, zengmin}@csu.edu.cn.*
• *Yaohang Li is with the Department of Computer Science, Old Dominion University, Norfolk, VA 23529 USA.. E-mail: yaohang@cs.odu.edu.*

Manuscript received 7 September 2021; revised 3 March 2021; accepted 22 April 2022. Date of publication 27 April 2022; date of current version 3 April 2023.
This work was supported in part by the National Natural Science Foundation of China under Grant 61832019, in part by Hunan Provincial Science and Technology Program under Grants 2019CB1007 and 2021RC4008, in part by 111 Project under Grant B18059, in part by the Fundamental Research Funds for the Central Universities of Central South University under Grant 502221903, and in part by the High Performance Computing Center of Central South University.
(Corresponding author: Min Li.)
Digital Object Identifier no. 10.1109/TCBB.2022.3170719

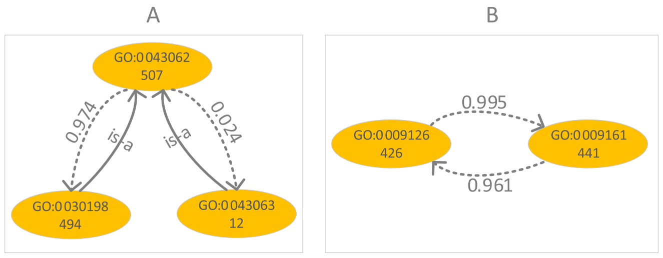


Fig. 1. The relationship of GO terms. Each cell in the picture has two lines, where the first line denotes the GO term id, and the second line is the count frequency of the GO term in training set. " $G_i \longrightarrow G_j$" means G_i has relationship with G_j. " $G_i \dashrightarrow G_j$" means the probability of G_j when G_i appears and the probability is on the dashed arrow.

the co-occurrence between GO terms at the training process. The structure of Gene Ontology is a hierarchical graph, where a term is a node, the relationships between child terms and parent terms are represented as the edges. In order to improve the prediction of protein functions, we analyze the training set and observe some interesting results. We take GO:0043062 (extracellular structure organization), GO:0030198 (extracellular matrix organization), and GO:0043063 (intercellular bridge organization) as an example. As shown in Fig. 1A, both GO:0030198 and GO:0043063 terms are the children of GO:0043062 term based on the is-a relationship. If protein P is annotated with GO:0030198 term or GO:0043063 term, P must also be annotated with GO:0043062 term. Moreover, we find that the numbers of proteins that are annotated with GO:0030198, GO:0043063, or GO:0043062 terms are 494, 12, and 507, respectively. It means that if GO:0043062 appears, GO:0030198 will also appear with a very high probability (0.974), but GO:0043063 does not (0.024). Then, we can deduce that if a protein has GO:0043062 term, this protein is also annotated by GO:0030198 term with high probability. Although the parent terms can be safely inferred by child terms based on is-a or part-of relationship, we can see parent terms may be helpful to predict child terms according to the probability from the phenomenon. And the appearance probabilities of child terms are obviously different when parent terms appear. Another example is GO:0009126 (purine nucleoside monophosphate metabolic process) and GO:0009161 (ribonucleoside monophosphate metabolic process). There is no is-a or part-of relationship between GO:0009126 and GO:0009161 terms. However, the count frequencies of GO:0009126 term, GO:0009161 term, and combination of GO:0009126 and GO:0009161 terms in training set are 426, 441, and 424, respectively. According to the Bayesian equation, the probability of GO:0009161 term when GO:0009126 term appears is 0.995. This means although two GO terms do not have any relationships, a term can be helpful to predict another one. From previous analysis, we see some GO terms normally co-occur on a protein. We call the combination of two GO terms as co-occurrence. Co-occurrence of GO terms is useful in protein function prediction, which can reveal the dependence between the two GO terms. However, previous deep learning studies ignore this.

In this study, we present a deep learning framework, named DeepPFP-CO, which predicts protein functions with co-occurence of GO terms. DeepPFP-CO consists of two components: a feature combination component and a function prediction component. In the feature combination component, we extract and combine sequence/subsequence-based and PPI network data. Then, we use these data to score protein functions. In function prediction component, we use a Graph Convolutional Network (GCN) module to improve the accuracy of prediction with an effective correlation matrix based on the co-occurrences of GO terms.

In our evaluation, we first evaluate DeepPFP-CO in comparison with the other five methods (one official baseline method in CAFA3, one classical machine learning method, and three deep learning methods) on a test set. Then we further analyze the generalization of DeepPFP-CO and five competing methods on another dataset. Our computational results indicate that DeepPFP-CO outperforms all competing methods, demonstrating the effectiveness of features extracted from co-occurrence GO terms. In particular, we analyze the distribution of each method's predictive performance at the protein level. Finally, we present a case study to verify the effectiveness of DeepPFP-CO in positive annotations.

TABLE 1
Summary of Datasets

Dataset	BPO	CCO	MFO	ALL
Training set	51503	48204	33335	64810
2016 test set	1432	1094	658	1782
CAFA3 challenge set	2388	1092	1084	3323

2 Datasets

The datasets for training and evaluation are downloaded from UniProt. In our study, proteins are from SwissProt whose version is published on January, 2016 and October, 2016. The annotations with experimental evidence codes (EXP, MP, TAS, IGI, IDA, IEP, and IC) are considered to be experimental. We select proteins with experimental annotations and ignore proteins with ambiguous amino acid codes (J, U, X, B, O, Z) in their sequences. Inspired by Critical Assessment of Functional Annotation (CAFA) [33], the protein datas is divided into training dataset and test dataset according to the annotation time stamps. We use all proteins selected before January, 2016 as the training dataset. The 2016 test set includes all proteins collected between January, 2016 and October, 2016 and we use 23 target species that are in CAFA3 evaluation set to filter the 2016 test set. We also use the CAFA3 challenge dataset to evaluate the generalization performance of DeepPFP-CO. The number of proteins in each dataset is shown in Table 1. To validate the performance of the model on the most recent dataset, we collated the most recent data since January, 2022 and use it to train a new model. The predictive performance is shown in Supplementary Table S1 and Table S2, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2022.3170719.

In our experiments, the version of GO released on January, 2016 are used. This version of GO has 29,265 biological process classes, 4,035 cellular component classes, and 10,694 molecular function classes. While propagating protein functions, we only consider the relationships of is-a and part-of.

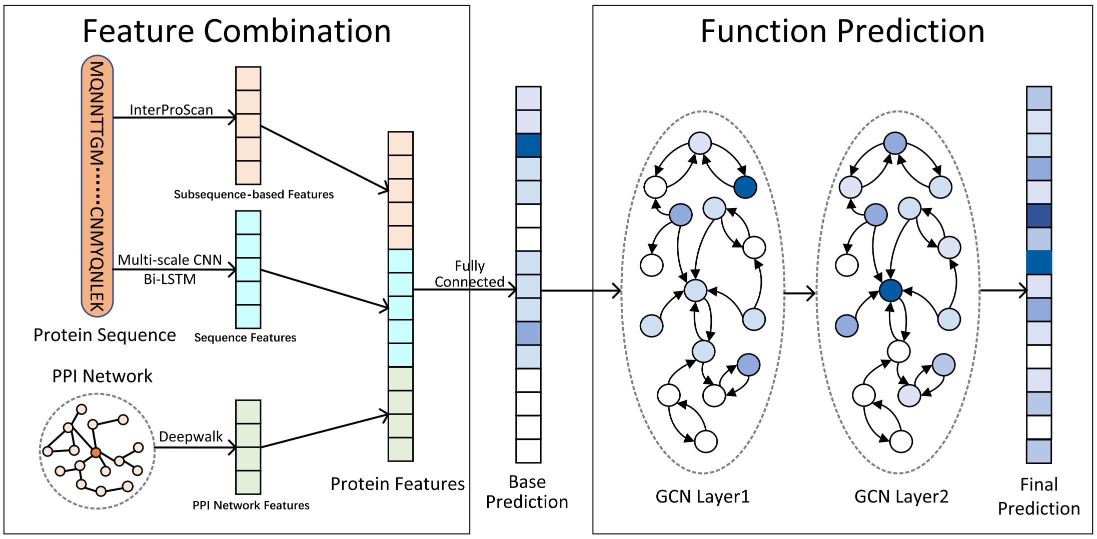


Fig. 2. The architecture of DeepPFP-CO. In feature combination component, we extract protein sequence features, subsequence-based features including motifs and domains, and PPI network features. Then we use these features to score the selected functions. In function prediction component, we use GCN module based on GO terms' co-occurrence to further deduce the protein functions.

Only these two relationships can reliably group annotations. For a protein, we look for the functions in UniProt and their ancestor nodes through these two relationships. All these found functions are annotated to the protein. After this step, we count the occurence of each GO term in the training dataset. We use the features from feature combination component to score all terms with numbers $\geq$50, $\geq$40, and $\geq$30 for BPO, MFO, and CCO, respectively. In function prediction component, we predict all terms, that appear in the training set.

We use the PPI data from STRING database [34] to construct a PPI network. In addition, we supplement the PPI network with orthology relations obtained from EggNOG database [35].

3 Methods

DeepPFP-CO predicts protein functions with a two-component design. The feature combination component integrates various protein features to predict protein functions. Firstly, we use multi-CNN [36], [37] and Bi-LSTM [38] to extract local and global sequence-based features. Then, we obtain PPI networks features and subsequence-based features including motifs and domains. The combination of the above three features is fed to two fully connected layers. In function prediction component, we use GCN module to further improve the predicted quality of protein function prediction with co-occurrence of GO terms. Fig. 2 shows the architecture of DeepPFP-CO.

3.1 Notation

Let D be a training set with N_D proteins, P_i be the ith protein, G_i be the ith GO term, and N_{G_i} be the number of proteins annotated by G_i in D. Denote $I(G_i, P_j)$ as a binary indicator, showing if P_j is associated with G_i. That is, if protein P_j is annotated by G_i, $I(G_i, P_j)$ is 1, otherwise 0.

3.2 Sequence Features

In the past decades, a large number of protein sequences have been obtained [39]. Protein sequences are composed of various amino acids. We regard the sequence of protein as sentence where different amino acids represent various words. We use the word2vec algorithm [40] to represent amino acids. Fragments of sequences that contain functional information have different numbers of amino acids. It is difficult to design a specific convolution kernel that obtains the complete local protein sequences features. Multi-scale CNN [41], [42], [43] is one of the classic deep learning models that extracts various local features with different kernel sizes. Thus, we use multi-scale CNN to capture local features of protein sequences. The global information of protein sequences is important to identify protein functions. The Long and Short Term Memory (LSTM) is suitable for capturing the long-range information in a sequence. Bi-LSTM structure is composed of a forward LSTM and backward LSTM, and captures bi-directional contextual features. We use Bi-LSTM structure to extract global features of protein sequences.

3.3 Subsequence-Based Features

Different from GONET [30], we also use subsequence-based features to predict protein functions. Subsequence-based features are important for protein function prediction. We use InterPro to get subsequence-based features including protein domains and motifs. InterPro offers a tool, named InterProScan, which provides functional analysis of protein sequences [44]. The subsequence-based features created by InterProScan are encoded into a 35020-dimensional binary vector. Since this vector is sparse, it is put into two fully connected layers to obtain its low-dimensional representation.

3.4 PPI Network Topological Features

Many proteins perform biological functions by interacting with other protein partners [45]. Therefore, PPI network information plays an important role in protein function prediction [46], [47], [48]. We use the same datasets from our previous protein function prediction work [28]. We download these proteins and their interaction data from STRING. The proteins are mapped to SwissProt. We also add the orthologous relations from EggNOG to the network. The network consists of 354,687 nodes and 54,552,077 edges. We use Deepwalk [49] to capture the topological features of each protein in the PPI network.

3.5 Function Deduction

In function prediction component, we deduce the protein functions based on the co-occurrence of GO terms. From previous analysis, we know that the appearance of one term is often accompanied by another one, even they do not have the relationships of is-a or part-of. This provides useful information to further enhance protein function prediction. To improve the predicted quality of protein function prediction, we use the GCN module to explore and capture such important dependencies. GCN [50] performs semi-supervised classification based on a topological graph. The key idea of GCN is to update the representations of nodes through information dissemination between nodes. Different from the convolution operation in CNN, the goal of GCN is to learn the function f(·,·) on the graph G. It can be written as:

$$H^{l+1} = f\left(H^l, A\right) \tag{1}$$

where H^l represents the feature description of each GO term and A is the corresponding correlation matrix. After the convolutional operation, f(.,.) can be represented as:

$$H^{l+1} = h\left(\hat{A}H^lW^l\right) \tag{2}$$

where $\hat{A}$ normalizes correlation matrix A and W^l is a transformation matrix that updates weights in the training process.

We use GCN [50], [51] to improve the performance of protein function prediction based on co-occurrence relations of GO terms. We represent the co-occurrence of GO terms by an effective correlation matrix. The details of the correlation matrix are described below.

First of all, for a term G_i , we compute N_{G_i} that represents the number of proteins annotated with the G_i in the training dataset. We compute co-occurrence number of proteins both annotated with the functions G_i and G_j in the training dataset. And we represent all these co-occurrence numbers of G_i with the matrix $M_i \epsilon R^{1*C}$, where C is the number of functions in the training dataset. Then, we compute the conditional probability matrix with the following formulas as:

$$P_i = \; M_i/N_{G_i} \tag{3}$$

where $P_{ij} = P(G_j|G_i)$ and $P(G_j|G_i)$ denotes the probability of occurrence of function G_j when function G_i appears. Since P_{ij} != P_{ji}, this matrix is asymmetric. Constructing the matrix may result in a long-tailed distribution. We use a threshold t to filter noise, and the operation can be written as:

$$A_{ij} = \begin{cases} 0, & if\ P_{ij} < t \\ 1, & if\ P_{ij} \geq t \end{cases} \tag{4}$$

where A is a binary correlation matrix.

Through the above formula, we know that the score of a node will be added by the score of itself and its neighbors, which may result in over-smoothing. And to alleviate this problem, we use the following reweighted scheme:

$$A'_{ij} \;=\; \begin{cases} \frac{b}{\sum_{j=1\ \ i\,!=j}^{C} A_{ij}}, & if\ i\,!=j \\ 1-b, & if\ i=j \end{cases} \tag{5}$$

where A' is the reweighted correlation matrix and b determines the weights assigned to a node itself and other correlated nodes. When b approaches 1, the neighbor nodes of the center node contribute more. On the other hand, when b approaches 0, the features of the center node are more relevant.

3.6 Implemental Details

DeepPFP-CO has many hyperparameters such as convolutional kernel size and types of activation functions to be used. The loss function and optimizer of our model are binary cross-entropy loss and Adam, respectively. In the first part, we use multi-scale CNN to capture local features of protein sequences. Kernel sizes of multi-CNN are [3, 5, 7, 9, 11, ..., 31], the number of channels is 64. The stride is 1. And the activation function is ReLU, and the max-pooling layer size is 1,000. For Bi-LSTM structure, the hidden size is 256 and the number of recurrent layers is 2. Then, the output of Bi-LSTM is fed to the max-pooling layer with the size of 512*1. The subsequence-based features, represented as a 35,020-dimensional binary vector, are fed to a fully connected layer with 512 neurons. The dimension of the features of PPI network is 256. In the second part, the input contains all GO terms that appear in the training dataset. There are some unpredictable functions by feature combination component. To solve this cold start problem, we use the Diamond tool [52] to supplement. The output of two GCN layers is a 64-dimensional vector and is fed to the max-pooling layer that outputs the propensity of functions.

4 Results

4.1 Competing Methods

4.1.1 Naïve Method

The naïve method is the official baseline method in CAFA3. The naïve method annotates all proteins with the same terms based on the relative frequency of GO terms in training dataset D. The propensity of a protein P_j with G_i is calculated as follows:

$$S(G_i, P_j) = \; \frac{N_{G_i}}{N_D} \tag{6}$$

where N_{G_i} is the number of proteins with G_i in D and N_D is the number of proteins in the training set. $S(G_i, P_j)$ is the prediction scores.

4.1.2 BlastKnn

For a given protein P_j, we use BLAST [53] to find homologous proteins in the training dataset based on specified sequence similarity score. Then, we use all functions of homologous proteins to annotate P_j. H represents the set of similar proteins of P_j in D, and $bitscore(P_j, s)$ is the similarity score between proteins P_j and s, where s is one of similar proteins in set H. We set the evalue to 0.001. $S(G_i, P_j)$ is calculated as follows:

$$S(G_i, P_j) = \frac{\sum_{s\epsilon H} bitscore(P_j, s) * I(G_i, P_j)}{\sum_{s\epsilon H} bitscore(P_j, s)} \tag{7}$$

4.1.3 DeepGOA

Zhang et al. developed DeepGOA which fuses different features of proteins to predict protein functions. Firstly, DeepGOA utilizes Bi-LSTM and CNNs with different filters to extract sequence features. Secondly, DeepGOA combines sequence features with the protein subsequence-based features including protein domains and motifs and generates sequence-based features. Then, DeepGOA extracts topological features of the PPI network. Finally, subsequence-based features and topological features are concatenated together to predict protein functions.

4.1.4 DeepGO and DeepGOPlus

M. Kulmanov et al. developed two deep learning-based models: DeepGO and DeepGOPlus. DeepGO firstly uses deep learning techniques to predict protein functions. DeepGO combines sequence and topological features of PPI networks to predict protein functions by using hierarchical classifier. In the training process, DeepGO trained three individual models for BPO, CCO, and MFO. Compared with DeepGO, DeepGOPlus represents the input sequence with one-hot coding and uses a set of CNN layers with different filters to learn specific sequence features.

4.2 Evaluation Metrics

We use two evaluation metrics to evaluate the performance of DeepPFP-CO. F_{max} is maximum protein-centric F-measure computing overall prediction thresholds and $AUPR$ is a metric for evaluating predictions with imbalanced datasets [54]. The formulas are as follows:

$$pr_j(t) = \frac{\sum_i 1(S(G_i, P_j) \geq t) * I(G_i, P_j)}{\sum_i 1(S(G_i, P_j) \geq t)} \tag{8}$$

$$rc_j(t) = \frac{\sum_i 1(S(G_i, P_j) \geq t) * I(G_i, P_j)}{\sum_i I(G_i, P_j)} \tag{9}$$

$$AvgPr(t) = \frac{1}{m(t)} * \sum_{i=1}^{m(t)} pr_j(t) \tag{10}$$

$$AvgRc(t) = \frac{1}{n} * \sum_{i=1}^{n} rc_j(t) \tag{11}$$

where $m(t)$ is the quantity of predicted proteins with at least one GO term, n is the total quantity of proteins, and $1(.)$ is 1 if the condition is true, otherwise 0.

TABLE 2
The Comparison Performance of on 2016 Test Set Which Is Generated by a Time-Based Split

Method	F_{max}			$AUPR$		
	MFO	BPO	CCO	MFO	BPO	CCO
Naïve	0.274	0.29	0.588	0.147	0.192	0.503
BlastKnn	0.579	0.434	0.646	0.477	0.299	0.500
DeepGO	0.425	0.362	0.618	0.375	0.286	0.660
DeepGOA	0.559	0.410	0.684	0.531	0.343	0.697
DeepGOPlus	0.576	0.439	0.690	0.531	0.360	0.715
DeepPFP	0.569	0.422	0.680	0.557	0.367	0.726
DeepPFP-CO	**0.613**	**0.463**	**0.703**	**0.593**	**0.419**	**0.731**

$$F_{max} = \max_t \left\{ \frac{2 * AvgPr(t) * AvgRc(t)}{AvgPr(t) + AvgRc(t)} \right\} \tag{12}$$

4.3 Evaluation and Comparison

4.3.1 Comparison With Other Methods

We evaluate DeepPFP-CO and other computational methods on the two test sets: the 2016 test set and CAFA3 challenge dataset. The details of test sets are shown in Table 1. Naive is a baseline comparison method. BlastKnn is a machine learning method. DeepGO, DeepGOA, and DeepGOPlus predict protein functions by using deep learning techniques. As mentioned above, DeepGO and DeepGOA both combine protein sequences and PPI network information to predict protein functions. DeepGOPlus is a sequence-based method that uses the combination of predictions of deep learning techniques and predictions by using Diamond to predict protein functions. Compared with DeepPFP-CO, DeepPFP offers the prediction results by the features generated from feature combination component.

Table 2 shows the performance of DeepPFP-CO and other computational methods on the 2016 test set. We observe that DeepPFP-CO achieves the best performance when being quantified with F_{max} and $AUPR$. DeepPFP-CO achieves F_{max} of 0.463, 0.613, and 0.703 for BPO, MFO, and CCO, respectively, which is better than DeepGOPlus (0.439, 0.576, 0.69). Compared with DeepGO that is the first deep learning model of protein function prediction, DeepPFP-CO improves F_{max} about 44.2%(MFO), 27.9%(BPO), and 13.8% (CCO). From the table, we can see DeepPFP achieved F_{max} of 0.569 over MFO, 0.422 over BPO, and 0.68 over CCO, which performed a little better than DeepGOA in most cases and worse than DeepPFP-CO. It also indicates the advantage of DeepPFP-CO with the GCN component.

In order to evaluate the generalization performance of DeepPFP-CO, we use the CAFA3 challenge dataset to test. Table 3 compares the evaluation results of each method. The table shows that DeepPFP-CO performs better than other comparison methods in most cases. For example, in the MFO, F_{max} and $AUPR$ of DeepPFP-CO are 0.57 and 0.542, respectively. It improves about 4.8% (F_{max}) and 12.2% ($AUPR$) than DeepGOPlus. But in terms of $AUPR$, DeepPFP-CO achieves 0.603 in the CCO, which is slightly lower than that of DeepGOPlus (0.605).

TABLE 3
The Comparison Performance of on CAFA3 Challenge Dataset

Method	F_{max}			$AUPR$		
	MFO	BPO	CCO	MFO	BPO	CCO
Naïve	0.293	0.332	0.542	0.153	0.229	0.448
BlastKnn	0.549	0.423	0.580	0.430	0.299	0.463
DeepGO	0.375	0.416	0.564	0.278	0.327	0.578
DeepGOA	0.470	0.467	0.604	0.390	0.397	0.594
DeepGOPlus	0.544	0.431	0.606	0.483	0.336	**0.605**
DeepPFP	0.544	0.486	0.605	0.508	0.421	0.588
DeepPFP-CO	**0.570**	**0.503**	**0.616**	**0.542**	**0.425**	0.603

To validate the performance of DeepPFP-CO on the most recent dataset, we collated the most recent data since January, 2022 to train the model. All results are shown in Supplementary Table S2, available online. We observe that DeepPFP-CO achieves F_{max} of 0.486, 0.679, and 0.643 for BPO, MFO, and CCO, respectively, which is better than other compared methods. The observation is the same as the results in Table 2. This indicates that DeepPFP-CO performs better across different datasets. In addition, we compared the DeepPFP-CO with the ablation model (DeepPFP-CO_R) trained without is-a and part-of relationships. This aims to analyze influence of nodes in GO term graph without the edges which the co-occurrences equal 1. The results are shown in Supplementary Table S2, available online. From the table, we can see that the Fmax and AUPR of DeepPFP-CO_R are lower than that of DeepPFP-CO which indicates that the relationships of is-a and part-of maintain the connectivity and information which is helpful to the final prediction. Supplementary Fig. S3, available online, shows the precision-recall curves of DeepPFP-CO_R and DeepFPF-CO. The results on most recent dataset indicate that DeepPFP-CO is an effective method.

4.3.2 Comparison at the Protein Level

We further analyze the poor, median, and good performance of DeepPFP-CO and other compared methods at the protein level. The F_{max} content that is binned into three ranges. The first range where the F_{max} is between 0.0 and 0.4 denotes that methods perform poorly. The second ($0.4<=F_{max}<0.7$) and third ranges ($0.7<=F_{max}<=1$) indicate that methods achieve median and good predictive performance, r espectively. Fig. 3 shows the number of proteins for which a given method obtains poor, median, good predictive performance in MFO. We also report the distributions of each method in BPO and CCO in Supplementary Figs. S1 and S2, available online. The F_{max} performance of predictions with the use of Naïve method is less than 0.7 for the whole 2016 test set. For DeepGO, the number of proteins with the best predictive performance is lower than poor and median predictive performance. DeepPFP-CO, DeepGOA, DeepPFP, and DeepGOPlus demonstrate good F_{max} performance for more than 50% of proteins. Moreover, DeepPFP-CO shows good predictive performance ($0.7<=F_{max}<=1$) for 379 proteins, which is about 14.5%, 8%, and 12.1% higher than DeepGOPlus, DeepPFP, and DeepGOA, respectively. When quantifying the number of proteins with poor predictive performance ($0<=F_{max}<0.4$), 134 predictions of DeepPFP-CO fall into this category, which is also less than DeepGOPlus, DeepPFP, and DeepGOA.

In addition to comparing the distribution of each method's predictive performance, we count the number of the methods predicting best for each protein according to F_{max} on the whole 2016 test set. First, we calculate F_{max} of each method on a protein. Then, we rank the best method on this protein and calculate the statistics in all proteins in the 2016 test set, which is shown in Fig. 4. In BPO, CCO, and MFO, DeepPFP-CO is the best prediction method in more protein cases than the other methods. Specifically, DeepPFP-CO is about 26.5%, 26.3%, and 14.4% higher than DeepGOPlus about 52.4%, 35.2%, and 47.1% higher than DeepPFP for BPO, CCO and MFO, respectively. This indicates that DeepPFP-CO is an effective method.

4.3.3 Case Study

To gain further insight into why DeepPFP-CO is able to improve protein function prediction, we choose one protein

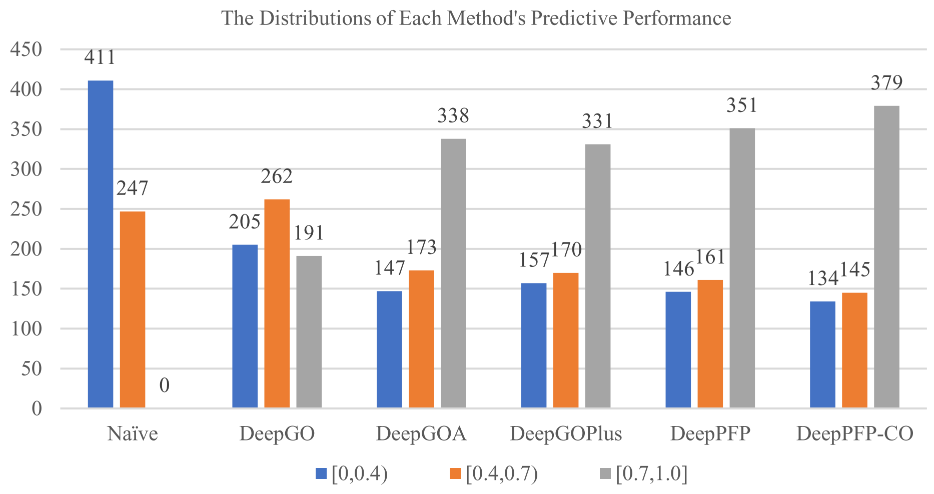


Fig. 3. The distributions of predictive performance of DeepPFP-CO and other compared methods at the protein level when quantified with F_{max}. The blue bar, orange bar and gray bar denote the number of proteins with bad ($0<=F_{max}<0.4$), median ($0.4<=F_{max}<0.7$), best ($0.7<=F_{max}<=1$) predictive performance, respectively.

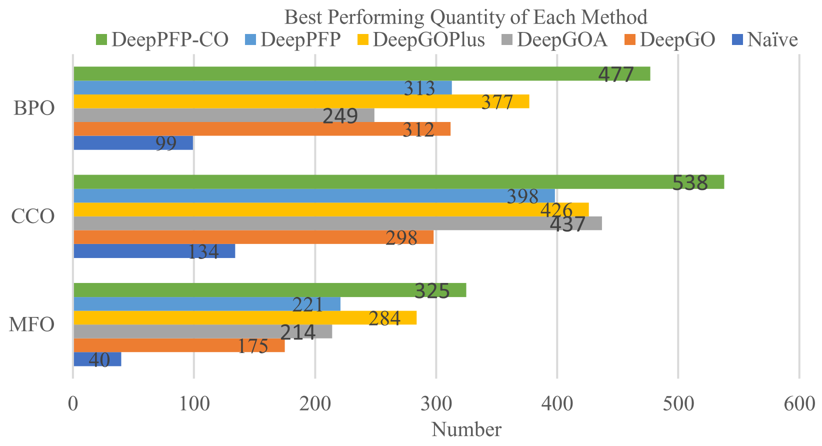


Fig. 4. The quantity of each method which performances best on single protein in BPO, CCO, and MFO.

(name: PLPR4_RAT) from the 2016 test set to explore effects in the CCO. As shown in Table 4, PLPR4_RAT is annotated with 14 native GO terms. Although the Naive method predicts 11 GO terms, there are only three correctly predictive GO terms. In contrast, the predictive GO terms of DeepPFP-CO and DeepGOPlus are all correct. Moreover, we have the following interesting observations in Table 4. First, although the GO:0005622 and GO:0044424 terms are not native annotated GO terms, they are annotated as positive GO terms by Naïve, DeepGO, DeepGOA, and DeepPFP. Second, GO:0005623 and GO:0044464 terms are correctly annotated by DeepPFP-CO as well as the competing methods. Finally, we observe that DeepPFP-CO is the only method that correctly annotates the GO:0016021 and GO:0031224 terms. The above observations indicate that DeepPFP-CO provides accurate predictions and offers correct annotations for specific terms that are not easily annotated by other methods.

We further explore the important aspect by comparing the difference of the predictive propensity between DeepPFP and DeepPFP-CO in Fig. 5. DeepPFP offers the predictions without GCN component of DeepPFP-CO. Fig. 5A not only shows the predicted score of GO:0005622 term and GO:0044424 term by DeepPFP but also the relations with their neighborhood terms. And Fig. 5A shows that DeepPFP correctly annotates the GO:0005623 and the GO:0044464 terms. Compared Figs. 5A with 5B, we observe that DeepPFP-CO eliminates these wrong annotations (GO:0005622 term, GO:0044424 term). From Figs. 5C and 5D, we can find that DeepPFP-CO correctly annotates the GO:0016021 and GO:0031224 terms. Although DeepPFP-CO incorrectly annotates the GO:0005887 and GO:0031226 terms, DeepPFP-CO improves the predictive propensity. The above observations support that DeepPFP-CO significantly improves the performance of protein function prediction based on co-occurrence information.

TABLE 4
The Prediction of the Protein (PLPR4_RAT) With Different Methods

Labels	Naive	DeepGO	DeepGOA	DeepGOPlus	DeepPFP	DeepPFP-CO
GO:0005623	GO:0005623	GO:0005623	GO:0005623	GO:0005623	GO:0005623	GO:0005623
GO:0044464	GO:0044464	GO:0044464	GO:0044464	GO:0044464	GO:0044464	GO:0044464
GO:0016020	GO:0016020	GO:0016020	*	GO:0016020	GO:0016020	GO:0016020
GO:0005886	*	*	*	GO:0005886	GO:0005886	GO:0005886
GO:0044425	*	*	*	GO:0044425	GO:0044425	GO:0044425
GO:0071944	*	*	*	GO:0071944	GO:0071944	GO:0071944
GO:0044459	*	*	*	GO:0044459	*	GO:0044459
GO:0016021	*	*	*	*	*	GO:0016021
GO:0031224	*	*	*	*	*	GO:0031224
GO:0005887	*	*	*	*	*	*
GO:0009897	*	*	*	*	*	*
GO:0009986	*	*	*	*	*	*
GO:0031226	*	*	*	*	*	*
GO:0098552	*	*	*	*	*	*
	GO:0005622	GO:0005622	GO:0005622		GO:0005622	
	GO:0044424	GO:0044424	GO:0044424		GO:0044424	
	GO:0005737	GO:0005737	GO:0005737			
	GO:0044444	GO:0044444	GO:0044444			
	GO:0043229					
	GO:0043231					
	GO:0043226					
	GO:0043227					

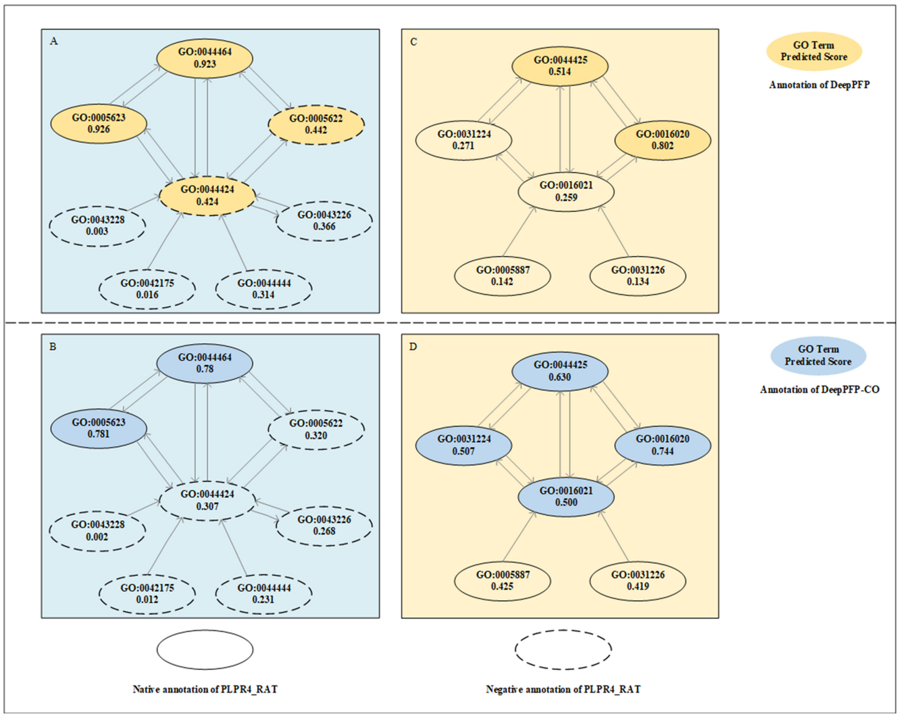


Fig. 5. Figures A, B, C and D show the relationships of GO terms. "$G_i \rightarrow G_j$" describes the used co-occurrence of G_i to G_j. Each cell in the picture has two lines, where the first line is GO term id, the second line is the predicted score by each method. Figures A and C show the predicted score of GO terms by DeepPFP. Figure B and D show the predicted scores of GO terms by DeepPFP-CO.

5 Conclusion

Protein function prediction is a multi-label problem. In this study, we propose DeepPFP-CO to predict protein functions by taking advantage of co-occurrence of GO terms. DeepPFP-CO consists of two parts. In feature combination component, DeepPFP-CO uses multi-source protein features to score GO terms. In function prediction component, DeepPFP-CO constructs an effective correlation matrix based on the co-occurrence of GO terms to improve the prediction performance. With the help of co-occurrence GO terms, our computational results show that DeepPFP-CO is an effective method in predicting protein functions. A web-server implementation is freely available at https://csuligroup.com/DeepPFP/. Since GO annotations and UniProt data will be updated regularly, models trained on older data set have limited predictions on the new data set. We will train models using the latest datasets every few months and publish them on our website.

References

[1] A. Shehu, D. Barbará, and K. Molloy, "A survey of computational methods for protein function prediction," in *Big Data Analytics Genomics*. Berlin, Germany: Springer, 2016, pp. 225–298.
[2] E. Boutet et al., "UniProtKB/Swiss-Prot, the manually annotated section of the uniprot knowledgebase: How to use the entry view," in *Proc. Plant Bioinformatics*. Berlin, Germany: Springer, 2016, pp. 23–54.
[3] P. Radivojac et al., "A large-scale evaluation of computational protein function prediction," *Nature Methods*, vol. 10, no. 3, pp. 221–227, 2013.
[4] C. Cai, L. Han, Z. L. Ji, X. Chen, and Y. Z. Chen, "SVM-Prot: Web-based support vector machine software for functional classification of a protein from its primary sequence," *Nucleic Acids Res.*, vol. 31, no. 13, pp. 3692–3697, 2003.
[5] C. E. Jones, J. Schwerdt, T. A. Bretag, U. Baumann, and A. L. Brown, "GOSLING: A rule-based protein annotator using BLAST and gO," *Bioinformatics*, vol. 24, no. 22, pp. 2628–2629, 2008.
[6] N. Kaplan et al., "ProtoNet 4.0: A hierarchical classification of one million protein sequences," *Nucleic Acids Res.*, vol. 33, no. suppl_1, pp. D216–D218, 2005.
[7] R. You, Z. Zhang, Y. Xiong, F. Sun, H. Mamitsuka, and S. Zhu, "GOLabeler: Improving sequence-based large-scale protein function prediction by learning to rank," *Bioinformatics*, vol. 34, no. 14, pp. 2465–2473, 2018.
[8] J. Hou and X. Chi, "Predicting protein functions from PPI networks using functional aggregation," *Math. Biosciences*, vol. 240, no. 1, pp. 63–69, 2012.
[9] D. Piovesan, M. Giollo, C. Ferrari, and S. C. Tosatto, "Protein function prediction using guilty by association from interaction networks," *Amino Acids*, vol. 47, no. 12, pp. 2583–2592, 2015.
[10] R. You et al., "NetGO: Improving large-scale protein function prediction with massive network information," *Nucleic Acids Res.*, vol. 47, no. W1, pp. W379–W387, 2019.
[11] W. Peng, M. Li, L. Chen, and L. Wang, "Predicting protein functions by using unbalanced random walk algorithm on three biological networks," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 14, no. 2, pp. 360–369, Mar./Apr. 2017.
[12] B. Zhao et al., "A new method for predicting protein functions from dynamic weighted interactome networks," *IEEE Trans. Nanobiosci.*, vol. 15, no. 2, pp. 131–139, Feb. 2016.
[13] D. Lee, O. Redfern, and C. Orengo, "Predicting protein function from sequence and structure," *Nature Rev. Mol. Cell Biol.*, vol. 8, no. 12, pp. 995–1005, 2007.
[14] A. Martino, A. Rizzi, and F. M. F. Mascioli, "Supervised approaches for protein function prediction by topological data analysis," in *Proc. Int. Joint Conf. Neural Netw.*, 2018, pp. 1–8, doi: 10.1109/IJCNN.2018.8489307.
[15] D. Pal and D. Eisenberg, "Inference of protein function from protein structure," *Structure*, vol. 13, no. 1, pp. 121–130, 2005.
[16] J. Yang, R. Yan, A. Roy, D. Xu, J. Poisson, and Y. Zhang, "The I-TASSER suite: Protein structure and function prediction," *Nature Methods*, vol. 12, no. 1, pp. 7–8, 2015.
[17] A. Koike, Y. Niwa, and T. Takagi, "Automatic extraction of gene/protein biological functions from biomedical text," *Bioinformatics*, vol. 21, no. 7, pp. 1227–1236, 2005.

[18] G. Nenadic, S. Rice, I. Spasić, S. Ananiadou, and B. Stapley, "Selecting text features for gene name classification: From documents to terms," in *Proc. ACL Workshop Natural Lang. Process. Biomedicine*, 2003, pp. 121–128.

[19] A. Wong and H. Shatkay, "Protein function prediction using text-based features extracted from the biomedical literature: The CAFA challenge," *BMC Bioinf.*, vol. 14, 2013, Art. no. S14.

[20] G. Pandey, C. L. Myers, and V. Kumar, "Incorporating functional inter-relationships into protein function prediction algorithms," *BMC Bioinf.*, vol. 10, no. 1, pp. 1–22, 2009.

[21] M. Ashburner et al., "Gene ontology: Tool for the unification of biology," *Nature Genet.*, vol. 25, no. 1, pp. 25–29, 2000.

[22] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1819–1837, Aug. 2014.

[23] V. Gligorijević, M. Barot, and R. Bonneau, "deepNF: Deep network fusion for protein function prediction," *Bioinformatics*, vol. 34, no. 22, pp. 3873–3881, 2018.

[24] M. Kulmanov and R. Hoehndorf, "DeepGOPlus: Improved protein function prediction from sequence," *Bioinformatics*, vol. 36, no. 2, pp. 422–429, 2020.

[25] M. Kulmanov, M. A. Khan, and R. Hoehndorf, "DeepGO: Predicting protein functions from sequence and interactions using a deep ontology-aware classifier," *Bioinformatics*, vol. 34, no. 4, pp. 660–668, 2018.

[26] A. S. Rifaioglu, T. Doğan, M. J. Martin, R. Cetin-Atalay, and V. Atalay, "DEEPred: Automated protein function prediction with multi-task feed-forward deep neural networks," *Sci. Rep.s*, vol. 9, no. 1, pp. 1–16, 2019.

[27] F. Zhang, H. Song, M. Zeng, Y. Li, L. Kurgan, and M. Li, "DeepFunc: A deep learning framework for accurate prediction of protein functions from protein sequences and interactions," *Proteomics*, vol. 19, no. 12, 2019, Art. no. 1900019.

[28] F. Zhang et al., "A deep learning framework for gene ontology annotations with sequence-and network-based information," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 18, no. 6, pp. 2208–2217, Nov./Dec. 2021.

[29] R. You, X. Huang, and S. Zhu, "DeepText2GO: Improving large-scale protein function prediction with deep semantic text representation," *Methods*, vol. 145, pp. 82–90, 2018.

[30] J. Li, L. Wang, X. Zhang, B. Liu, and Y. Wang, "GONET: A deep network to annotate proteins via recurrent convolution networks," in *Proc. IEEE Int. Conf. Bioinf. Biomed.*, 2020, pp. 29–34.

[31] K. Hippe, S. Gbenro, and R. Cao, "ProLanGO2: Protein function prediction with ensemble of encoder-decoder networks," in *Proc. 11th ACM Int. Conf. Bioinf. Comput. Biol. Health Inform.*, 2020, pp. 1–6.

[32] V. Gligorijević et al., "Structure-based protein function prediction using graph convolutional networks," *Nature Commun.*, vol. 12, no. 1, pp. 1–14, 2021.

[33] N. Zhou et al., "The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens," *Genome Biol.*, vol. 20, no. 1, pp. 1–23, 2019.

[34] D. Szklarczyk et al., "STRING v10: Protein–protein interaction networks, integrated over the tree of life," *Nucleic Acids Res.*, vol. 43, no. D1, pp. D447–D452, 2015.

[35] J. Huerta-Cepas et al., "eggNOG 4.5: A hierarchical orthology framework with improved functional annotations for eukaryotic, prokaryotic and viral sequences," *Nucleic Acids Res.*, vol. 44, no. D1, pp. D286–D293, 2016.

[36] J. Gu et al., "Recent advances in convolutional neural networks," *Pattern Recognit.*, vol. 77, pp. 354–377, 2018.

[37] M. Zeng, M. Li, Z. Fei, Y. Yu, Y. Pan, and J. Wang, "Automatic ICD-9 coding via deep transfer learning," *Neurocomputing*, vol. 324, pp. 43–50, 2019.

[38] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Netw.*, vol. 18, no. 5-6, pp. 602–610, 2005.

[39] N. Yuvaraj, K. Srihari, S. Chandragandhi, R. A. Raja, G. Dhiman, and A. Kaur, "Analysis of protein-ligand interactions of SARS-Cov-2 against selective drug using deep neural networks," *Big Data Mining Analytics*, vol. 4, no. 2, pp. 76–83, 2021.

[40] S. M. Yusuf, F. Zhang, M. Zeng, and M. Li, "DeepPPF: A deep learning framework for predicting protein family," *Neurocomputing*, vol. 428, pp. 19–29, 2021.

[41] R. Xin, J. Zhang, and Y. Shao, "Complex network classification with convolutional neural network," *Tsinghua Sci. Technol.*, vol. 25, no. 4, pp. 447–457, 2020.

[42] M. Li, Z. Lu, Y. Wu, and Y. Li, "BACPI: A bi-directional attention neural network for compound-protein interaction and binding affinity prediction," *Bioinformatics*, vol. 38, pp. 1995–2002, 2022.

[43] M. Zeng, Y. Wu, C. Lu, F. Zhang, F.-X. Wu, and M. Li, "DeepLncLoc: A deep learning framework for long non-coding RNA subcellular localization prediction based on subsequence embedding," *Brief. Bioinf.*, vol. 23, no. 1, 2022, Art. no. bbab360.

[44] A. Mitchell et al., "The interpro protein families database: The classification resource after 15 years," *Nucleic Acids Res.*, vol. 43, no. D1, pp. D213–D221, 2015.

[45] X. Meng, J. Xiang, R. Zheng, F. Wu, and M. Li, "DPCMNE: Detecting protein complexes from protein-protein interaction networks via multi-level network embedding," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 19, no. 3, pp. 1592–1602, 2022.

[46] M. Zeng et al., "A deep learning framework for identifying essential proteins by integrating multiple types of biological information," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 18, no. 1, pp. 296–305, Jan. 2021.

[47] M. Zeng, F. Zhang, F.-X. Wu, Y. Li, J. Wang, and M. Li, "Protein–protein interaction site prediction through combining local and global features with deep neural networks," *Bioinformatics*, vol. 36, no. 4, pp. 1114–1120, 2020.

[48] F. Zhang, W. Shi, J. Zhang, M. Zeng, M. Li, and L. Kurgan, "PROBselect: Accurate prediction of protein-binding residues from proteins sequences via dynamic predictor selection," *Bioinformatics*, vol. 36, no. Supplement_2, pp. i735–i744, 2020.

[49] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 701–710.

[50] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

[51] Z.-M. Chen, X.-S. Wei, P. Wang, and Y. Guo, "Multi-label image recognition with graph convolutional networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5177–5186.

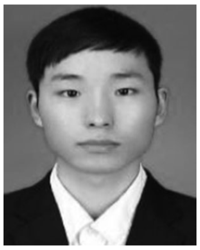
[52] B. Buchfink, C. Xie, and D. H. Huson, "Fast and sensitive protein alignment using DIAMOND," *Nature Methods*, vol. 12, no. 1, pp. 59–60, 2015.

[53] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *J. Mol. Biol.*, vol. 215, no. 3, pp. 403–410, 1990.

[54] W. A. Abbasi, F. U. A. A. Minhas, and C. Biology, "Issues in performance evaluation for host–pathogen protein interaction prediction," *J. Bioinform. Comput. Biol.*, vol. 14, no. 3, 2016, Art. no. 1650011.

Min Li received the PhD degree in computer science from Central South University, China, in 2008. She is currently a professor and vice dean with the School of Computer Science and Engineering, Central South University, Changsha, Hunan, China. Her research interests include computational biology, systems biology and bioinformatics. She has published more than 100 technical papers in refereed journals such as *Genome Research*, *Genome Biology*, *Bioinformatics*, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, and conference proceedings. According to Google scholar, her paper citations are more than 7800 and H-index is 47.

Wenbo Shi received the BSc degree from Northeast Agricultural University, China, in 2019. He is currently working toward the postgraduate degree in bioinformatics with Central South University. His current research interests include bioinformatics, machine learning, and deep learning.



Fuhao Zhang received the BSc degree from the Chongqing University of Posts and Telecommunications, China, in 2014. He is currently working toward the PhD degree with the School of Computer Science and Engineering, Central South University, China. His current research interests include bioinformatics, network representation learning, and deep learning.

Min Zeng received the BS degree from Lanzhou University, in 2013, and the MS and PhD degrees in system science and computer science from Central South University, in 2016 and 2020, respectively. He is currently a lecturer with the School of Computer Science and Engineering, Central South University, Changsha, Hunan, China. His research interests include machine learning and deep learning techniques for bioinformatics and computational biology.

Yaohang Li received the MS and PhD degrees in computer science from Florida State University, Tallahassee, Florida, in 2000 and 2003, respectively. He is currently an associate professor with the Department of Computer Science, Old Dominion University, Norfolk, VA. His research interests include computational biology, monte carlo methods, and scientific computing. After graduation, he worked with Oak Ridge National Laboratory as a research associate for a short period. Before joining ODU, he was an associate professor with the Computer Science Department, North Carolina A&T State University.