Hue University Journal of Science: Techniques and Technology; pISSN 2588-1175 | eISSN 2615-9732

Vol. 132, No. 2B, 2023, 33–50; DOI: 10.26459/hueunijtt.v132i2B.7084

Boosting Prediction of Protein-Protein Interactions using Word Embedding Techniques

Tran Hoai Nhan[1], Bui Duc Hanh[1, 3], Nguyen Phuc Xuan Quynh [1], Truong Khanh Duy[2], Nguyen Tuong Tri[1, 2*]

[1] Department of Informatics, Hue University of Education, Hue University, Vietnam
[2] Institute of Open Education and Information Technology, Hue University, Vietnam
[3] Cam Thuy Secondary & High School, Cam Lo, Quang Tri, Vietnam

Abstract. Understanding protein-protein interactions (PPIs) helps to identify protein functions and develop other important applications such as drug preparation, protein-disease relationship identification. Machine learning methods have been developed for the PPI prediction task in order to reduce the cost and time of previous experimental methods. In this paper, we study a method for determining PPIs using deep learning and protein sequence representation learning. In our method, an word embedding technique is utilized for protein sequence representation learning. This technique captures the semantic relationship between amino acids in protein sequences. The semantic relationship is then used as the input information, which is fed into a neural network to help recognize the interaction signature of the input protein pair. Different from previous studies, we integrate the protein sequence embedding mechanism into a neural network model. Thereby, the protein sequence embedding is better controlled for PPI prediction by our neural network model. We evaluate our method on benchmark datasets including Yeast, Human, and eight different independent sets. In addition, we also conduct an extensive comparison with the other existing methods. Our results show that the proposed method is superior to other existing methods and achieves high efficiency in predicting cross-species PPIs. The dataset and our source code are available at https://github.com/thnhub/BoostPPIP.git.

Keywords: Protein-Protein interaction, Sequence analysis, Word embedding, Machine learning

1 Introduction

Determining protein-protein interactions (PPIs) is one of the important problems in the field of Bioinformatics. According to Li et al. [1, 3] understanding PPIs helps to identify protein functions and develop other important applications such as drug preparation, and identification of protein-disease relationships. In recent years, computational methods based on machine learning (ML) for PPI prediction are widely proposed and studied [4, 19]. ML methods are developed based on different biological information sources such as protein sequences, structural information of proteins, gene ontology annotation and semantic similarity of proteins

** Corresponding: ntuongtri@hueuni.edu.vn*

Received: 29–01–2023; Accepted: 13–03–2023

[2, 7]. In addition, protein sequence data is growing rapidly, which is creating an advantage over other sources of biological information [5]. To date, works using machine learning have produced various high-performance models to predict protein interactions from protein sequences alone. This success is mainly based on the representation of protein sequences and the selection of suitable learning models.

Among them, Shen [8] used the Conjoint Triad (CT) descriptors to encode physicochemical properties of amino acids in a protein sequence, and chose an SVM to learn classifying from those encoded physicochemical properties. Guo [6] used the Auto Covariance (AD) descriptors to extract features from the amino acid sequence in a protein, and then fed these features into a Support Vector Machine (SVM) to predict protein interactions. Because proteins bind to each other at certain regions of the protein, so the authors, You et al. [9, 20], Zhou et al. [11], Yang et al. [10], and Zhou et al. [12] suggested using Multi-scale Continuous and Discontinuous local descriptors to encode protein sequences. These authors then experimented with their ideas using SVM, Gradient Boosting Decision Tree (GBDT) as classifier in PPI prediction tasks. The physicochemical and sequence-order information can be used to describe amino acids sequence, Chen et al. [13] proposed using a combination of multiple descriptors including, Pseudo-Amino Acid Composition (PseAAC), Autocorrelation (AC), and CT to capture that information in encoding protein sequences. The authors then utilized a LightGBM algorithm to learn protein interactions from that extracted information. Besides, evolutionary information can be also mined for encoding protein sequences. In GTB-PPI model, Yu et al. [14] utilized Pseudo Position-Specific Scoring Matrix (PsePSSM) descriptors to extract the evolutionary information, which is stored in the Position-Specific Scoring Matrix (PSSM). To enhance PPI prediction performance for their model, Yu et al. [14] combined with sequence-order and physicochemical information using PseAAC, Pseudo Position-Specific Scoring Matrix (PsePSSM), Reduced Sequence Index-Vectors (RSIV) and AC descriptors.

The works mentioned above have shown that protein sequence descriptors have been widely applied to the PPI prediction problem. Features extracted in those ways can not only be used to feed traditional machine learning models but also for deep learning models, such as the work of Du et al. [5]. In addition, various techniques in feature engineering have also been proposed to build higher quality features for PPI prediction, such as Chen et al. [13], Yu et al. [14], and Yu et al. [18]. However, those feature extraction methods require a great human effort in feature engineering. To overcome this disadvantage, various works have attempted to design deep learning models capable of automatically learning protein sequence representation for the PPI prediction problem. For example, Hashemifar et al. [15] proposed the DPPI model, which is a Convolutional Neural Network (CNN), taking the evolutionary information as raw features to infer PPIs. However, Hashemifar's method runs extremely slow because it is required to run BLAST [40] against a huge protein the non-redundance database [41] to generate a PSSM matrix as its feature. Gonzalez-Lopez et al. [16] proposed the DeepSequencePPI model, which is based

on Recurrent Neural Networks (RNNs), learning embedding features from direct protein sequences without using any other feature extraction techniques. Since protein sequences can be very long, using RNNs to capture properties in protein sequences can lead to a vanishing gradient problem, which greatly affects the final prediction. Inspired by Natural Language Processing (NLP) techniques, Yao et al. [17] proposed the Res2vec framework, Tran and Nguyen [42] proposed the DeepCF-PPI model, which are capable to learn amino acid embeddings, providing a stepping stone for converting protein sequences into embedding vectors. But, both Res2vec and DeepCF-PPI were trained completely independently of their PPI predictors, leading to the fact that the generated embedding vector did not perform to its full potential in the PPI prediction task. To address the disadvantages of current methods and inherit the advantages of protein sequence representation learning, in this paper, we propose integrating the protein sequence embedding mechanism into a deep neural network, where an embedding matrix is learned by a Word2vec [23] model, is attached to a deep neural network to perform protein sequence embedding and the embedding process is controlled to perform PPI prediction. Through this work, we introduce a novel method to predict protein-protein interactions directly from sequence data and demonstrate the robustness of our method on a number of benchmark datasets.

This paper consists of 4 sections: Section 3 introduces the problem of PPI prediction and previous works, Section 4 is our proposed method, Section 5 is the experiment results and comparison with the other existing works on benchmark datasets, and final conclusion is introduced in Section 6.

2 Methods

Determining protein-protein interactions (PPIs) can be regarded as a binary classification problem. The objective is to classify a given protein pair as belonging to the interacting class (denoted 1) or to the non-interacting class (denoted 0). In this study, the input of the PPI prediction problem is a pair of protein sequences and its output is the probability of interaction. Based on the interaction probability, we can completely classify the given protein sequence pairs into class 0 or 1. In this section, we will detail our proposed method. First, we will describe the technique used to represent protein sequences into embedding features. Then, we will introduce our deep neural network (DNN) architecture for determining PPIs from the obtained embedding features.

2.1 Protein sequence representation learning

Nowadays, there are many word embedding learning techniques have been proposed, for example, Word2vec [23], GloVe [24], BERT [25], etc. In this work, we utilize the Word2vec

technique, Continuous bag of word (CBOW) model [23], because of its simple architecture and the ability to learn large amounts of data. To apply the Word2vec technique, we first consider the protein sequence is a sentence where each word is an amino acid. Inspired by the idea of the CBOW model, we then build an algorithm (named Amino Acid Encoding) to learn an embedding matrix, where each row of this matrix is a vector representing one of the 20 naturally occurring amino acids. Figure 1 illustrates the protein sequence translating into the corresponding sentence. Figure 3 describes and illustrates the architecture of the neural network used to learn the embedding matrix.

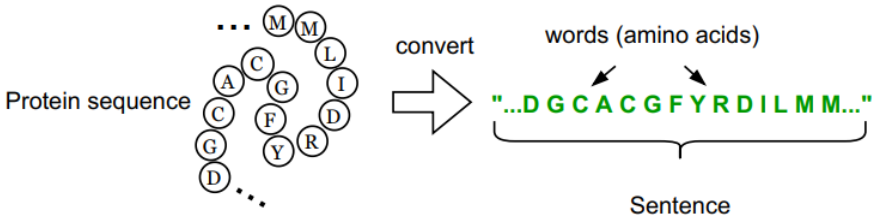

Fig. 1. An illustration of the translation of a protein sequence into a sentence in which an amino acid corresponds to a word

The Amino Acid Encoding algorithm has two stages, the T training dataset generating stage (steps 1-3) and the NN neural network training stage (steps 4-11). Let maxlen be the maximum length of the protein sequences in P, from that, the computational complexity of the algorithm is determined by the formula, $O(\text{maxlen} \times |P| + ep \times |T|)$. After running the Amino Acid Encoding algorithm we obtain the amino acid (word) embedding matrix W_1. This matrix is then used to produce embedding vectors for amino acid a and protein sequence p according to the formulas,

$$a_W = onehot(a) \cdot W_1, \tag{1}$$

$$p_W = onehot(p) \cdot W_1, \tag{2}$$

where $onehot(a) \in \mathbb{R}^{|V|}$ is a one-hot vector representing a, and $onehot(p) \in \mathbb{R}^{N,|V|}$ is one-hot matrix representing p with N is the sequence length.

On the other hand, W_1 is also used in the Embedding layer of the neural network used in the PPI prediction task that we will present in the next section. The $softmax(.)$ activation function is used in the *'Amino Acid Encoding'* to calculate probability distribution of words in the vocabulary V when knowing the center word is explained by the formula,

$$softmax(\cdot) = \frac{\exp(onehot(x) \cdot W_1 \cdot W_2)}{\sum_j^{|V|} \exp\left(onehot(y_j) \cdot W_1 \cdot W_2^{(j)}\right)} \tag{3}$$

where $onehot(x) \in \mathbb{R}^{|V|}$ maps x into an one-hot vector, $W_2^{(j)}$ is column j of matrix W_2.

Algorithm 1: *Amino Acid Encoding*

Input:

- The neural network NN has the architecture described as in Figure 2;
- The learning rate lr, the number of weights updating ep;
- The set of protein sequences P;
- The window w.

Output: The embedding matrix W_1.

Constructing the vocabulary V from P, where each unique amino acid is a word in V;
Building the corpus C from the set P by the following steps: Converting each protein sequence in P into the sentence in which each word is one amino acid;
Building the training set T from the copus C by the following steps: Selecting a sentence c in corpus C; moving w on over c; in w, getting the center word y and the set of context words x to put in T;
Initializing $W_1 \in \mathbb{R}^{|V|,h}$, $W_2 \in \mathbb{R}^{h,|V|}$ (h indicates the number of neurons of the Hidden layer of NN as well as embedding vector size);
Training the network NN to optimal matrices W_1 and W_2 using Stochastic Gradient Descent algorithm [26] by the following steps:
foreach ep, $(x, y) \in T$ **do**
    $\tilde{x} = \frac{1}{|x|} \sum_{x \in x} onehot(x)$, where $onehot(x) \in \mathbb{R}^{|V|}$ is one-hot vector of the word x;
    $\hat{y} = softmax(\tilde{x} \cdot W_1 \cdot W_2)$;
    $\text{error} = \hat{y} - onehot(y)$;
    $W_1 = W_1 - lr \times \tilde{x}^T \cdot (W_2 \cdot \text{error})^T$;
    $W_2 = W_2 - lr \times (\tilde{x} \cdot W_1 \cdot \text{error}^T)$;
return W_1 ;

To the network NN can capture the semantic relationship between amino acids, it is important to feed it a large set of sequences. In this study, we utilized the UniProtKB database [27]. Besides, the hyperparameters of NN including, the window w, the learning rate lr, and the number of steps in training ep are needed to choose carefully. To get those hyperparameters, we applied the grid search method. Finally, we select $w = 5$, $lr = 0.025$, and $ep = 10$, respectively.

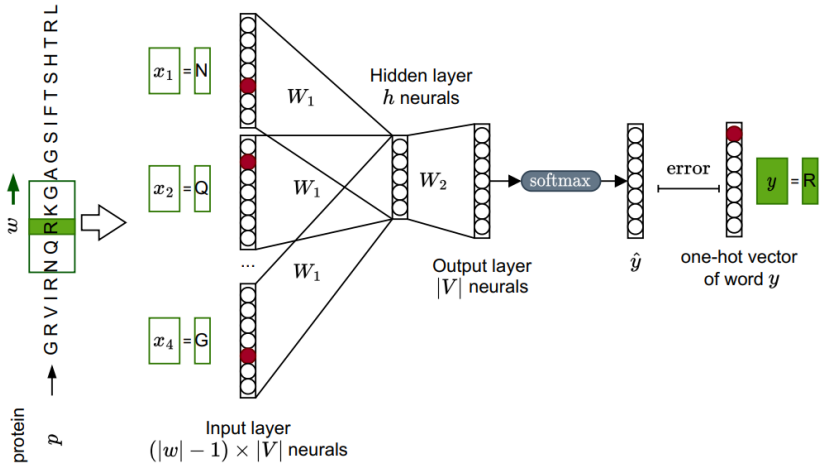

Fig. 2. CBOW (Continuous Bag of Words) model's architecture is used in the Amino Acid Encoding algorithm. In this figure, w is illustrated 5

2.2 Proposed model for PPI prediction

Deep neural networks (DNNs) [28] are a type of neural network with a high number of layers. The main role of DNN is to extract high-level abstraction features, remove noise, and reduce data size. We used DNN architectures to design a PPI prediction model with embedding features as input. The architecture of our proposed DNN model consists of 2 layers: feature extraction and classification. For convenience, we named this model BoostPPIP (Boost PPI Prediction). The general architecture of our PPI prediction model is illustrated in Figure 4.

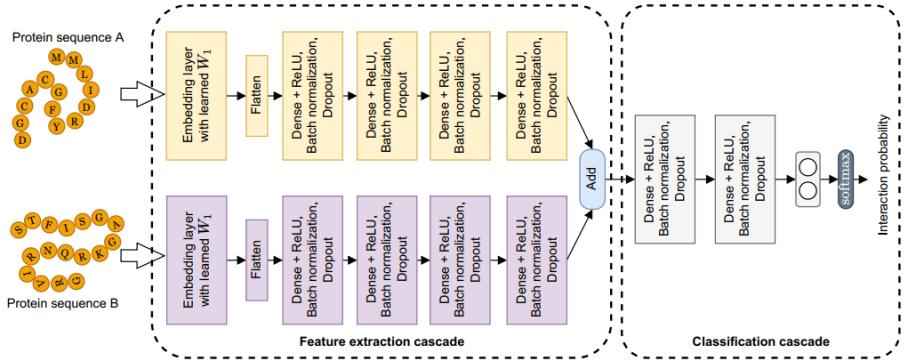

Fig. 3. The architecture of our proposed model, BoostPPIP (Boost PPI Prediction)

Since the BoostPPIP model needs to receive protein sequences of equal length, we fixed it with a value of N. Specifically, protein sequences are truncated on the right side if their length is greater than N, otherwise, "_" characters are added to the right side. Character "_" is represented by an embedding vector $\overleftarrow{0} \in \mathbb{R}^h$. The protein sequences are fixed after the Amino Acid Encoding algorithm performed. The Feature Extraction cascade consists of two branches and the architectures of the two branches are similar. Each branch is constructed by layers including, Embedding [28], Dense [28], Batch normalization [29], Dropout [30] and Flatten. The Embedding layer takes its input as a protein sequence, and its output is a matrix of size (N, h); where N indicates the fixed sequence length, h indicates the dimensional of amino acid embeddings. The embedding matrix W_1 is integrated into the Embedding layers to generate embedding vectors serving PPI prediction. The Flatten layers are tied after the Embedding layers in order to transform the output matrix of each Embedding layer into the vector with a size of $h * N$. Consequently, protein sequences after passing through the Embedding and Flaten layers, become embedding vectors. In this way, BoostPPIP can continue to adjust the weights of the matrix W_1 during training on PPI datasets. By this integration, the word embedding mechanism has been integrated into a deep neural network model, and the embedding matrix is fine-tuned to be more optimal for the task of PPI classification.

Mathematically, the embedding vector p sequence at the output of the Flatten layer is expressed by the formula,

$$v(p) = Flatten(Embedding(p)) = Flatten(p_W) = Flatten\left(w_{ij_{N,h}}\right) = w_{l_{h*N}}, \quad (4)$$

where p_W is defined as the formula (2). In the case, the model receives a $batch_n(v(p))$ – a set of n protein sequences, the formula (4) can be expressed by the following formula,

$$batch_n(v(p)) = w_{l_{h*N_n}}. \quad (5)$$

The Dense layer is the layer where each of its neurons receives input from all the neurons of the previous layer. We used the Dense class to learn the non-linear relationships of their inputs, transform high-dimensional space into low-dimensional space, and extract abstract features. To learn the non-linear relationship between the inputs, the activation ReLU [31] was added after the Dense layers, except the last one. To speed up training and avoid overfitting, Batch normalization [29] and Dropout [30] layers are added after each the Dense layer. So, if the input of Dense layer is $X = batch_n(v(p))$, its output vectors are calculated by the formula,

$$f(X) = Dropout_\alpha\left(batchnorm\big(ReLU(Dense(X))\big)\right) \quad (6)$$

where $Dense(X) = (X \cdot W + bias)$; $ReLU(x) = max(0, x)$; $batchnorm(X) = \frac{X - mean(X)}{\sqrt{std(X) + \epsilon}}$ with $\epsilon = 0.001$, $mean(X)$ and $std(X)$ are the mean and the standard deviation of column of X; $Dropout_\alpha(X)$ randomly assign with a rate α on a number of columns of X into values of 0; W is the learnable embedding matrix of the neural network BoostPPIP.

To connect two branches together and also to produce a feature vector representing the input protein sequence pair, we use an Add layer, through which two output feature vectors of two branches are added to form a single vector. Specifically, assuming that the input of the Add layer are $f_1(X)$ and $f_2(X)$, the its output is as formular (7).

$$F(X) = f_1(X) + f_2(X) \quad (7)$$

$F(X)$ then passed to the Classification cascade. Here, the input protein sequence pair representation is transformed into a two-dimensional vector in the final layer to be used for interaction determination. Finally, assigning the interaction probability to the input protein sequence pair, we use the 2-class Softmax function whose input is vector two at the output of the final layer. Specifically, if the input of 2-class Softmax is a two-dimensional vector, its output is calculated by the formula,

$$\hat{y} = softmax(x) = \frac{\exp(x)}{\sum_{i=1}^{2} \exp(x_i)} \quad (8)$$

The interaction probability of the input protein sequence pair of BoostPPIP was determined by applying the formulas (6) to (8). To train our model, we choose the value N as the average length of the protein sequences in the training set. In the experiments, we utilized the Adam algorithm [32] to train BoostPPIP with the following loss function,

$$loss(X,Y) = \frac{1}{n}\sum_{i=1}^{n} (\hat{y} - y)^2 \quad (9)$$

where X, Y are sets of n samples, protein pairs and labels, respectively.

In our experiments, the learning rate used to optimize the neural network is set to 0.001 with a penalty for error function of 0.001. The implementation is done with support from Python libraries including Tensorflow [33] and Scikit-learning [34].

3 Results and Discussion

3.1 Evaluation criteria

To evaluate the performance of the models, we use various evaluation metrics including, accuracy (Acc), sensitivity (Sen), precision (Pre), F1-score (F1) and Matthew's correlation coefficient (MCC). These metrics are defined by the following formula,

$$Acc = \frac{tp+tn}{tp+fp+tn+fn}, \quad (10)$$

$$Sen = \frac{tp}{tp+fn}, \quad (11)$$

$$Pre = \frac{tp}{tp+fp}, \quad (12)$$

$$F1 = 2 \times \frac{Pre \times Sen}{Pre+Sen}, \quad (13)$$

$$MCC = \frac{tp \times tn - fp \times fn}{\sqrt{(tp+fp)(tn+fn)(tp+tn)(fp+fn)}}. \quad (14)$$

where tp, tn, fp, fn are the number of positive samples (interacting protein pairs) predicted to be positive, and the number of negative samples (pairs) respectively. non-interacting proteins) was predicted to be negative, the number of positive samples was predicted to be negative, and the number of negative samples was predicted to be positive.

In addition, we also use the area under the Receiver curve operating characteristic (AUROC) and the Precision-Recall curve (AUPRC) to evaluate the performance of models. The large area represents the high performance of the model.

3.2 Datasets

We use ten PPI datasets for experiments on the proposed model and for comparison with existing methods as well. We divide them into two groups, the first group is used for cross-validation and the second group is used for the independent tests. The first group consists of two datasets, Yeast and Human. The Yeast dataset was built and introduced in the paper [8]. The Yeast dataset contains interacting protein pairs selected from the DIP database [35], including 5,594 interacting pairs and 5,594 non-interacting pairs, after removing protein pairs

with sequence length less than 50 and sequence identification greater than or equal to 40% using CD-HIT tool [36]. The Human dataset was introduced by Huang et al. [37], which was collected from the HPRD database (https://www.hprd.org/), consisting of 3,899 interacting protein pairs and 4,262 non-interacting protein pairs. The second group consists of 5 cross-species PPI datasets and 3 PPIs network datasets. These datasets were downloaded from the DIP database [35] (version 2017-02-05) and retained only protein pairs identified as physically interacting. Five cross-species PPI datasets include Caenorhabditis Elegans (Celeg), Escherichia Coli (Ecoli), Homo Sapiens (Hsapi), Helicobacter Pylori (Hpylo) and Mus Musculus (Mmusc); the number of samples is respectively 4,013, 6,954, 1,412, 1,420, and 313. Three PPI network datasets include One-core Network (CD9), Wnt-related Network (Wnt) and Cancer-specified Network (Cancer) with the number of positive samples of these data sets was 16, 96 and 108, respectively.

3.3 Optimal hyperparameters

To choose the optimal configuration for the BoostPPI model, we need to observe the influence on PPI prediction through different combinations of two hyperparameters, the number of Dense layers at the Feature Extraction cascade and size *h* of the embedding vector. Observation is performed on the Yeast dataset by dividing it into three parts, 70% for the training set, 10% for the validation set, and 20% for the test set. The optimal configuration is selected by comparing the model's scores from the scales on the test set. Figure 5) shows the results obtained by the BoostPPIP model with different configurations.

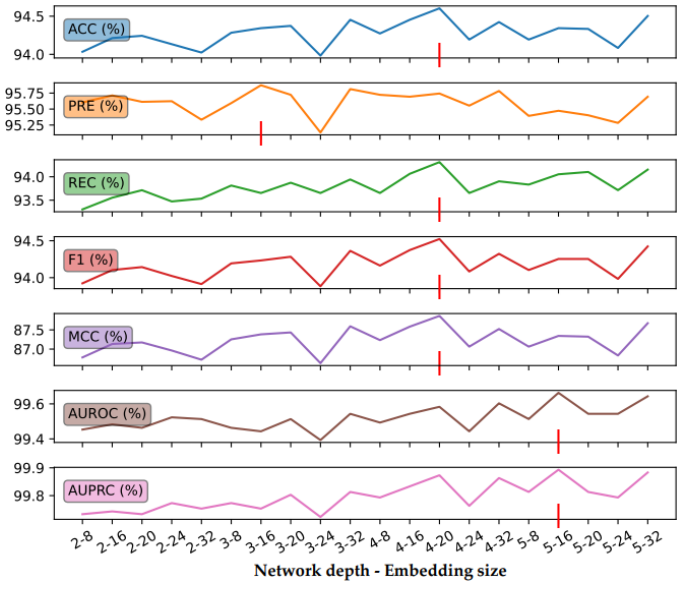

Fig. 4. Predictive performance of BoostPPIP model on the Yeast core dataset over different combinations of network depth and embedding size. The red vertical line at each metric represents the best combination

Our results (see Figure 5) show that the BoostPPIP model achieves the best performance on most measurements when the embedding vector size is 20 and the network depth is 4, with an Acc of 94.56%, Rec of 94.04%, F1 of 94.58% and MCC of 87.58%. Therefore, the embedding amino acid size of 20 and the number of Dense layers of 4 (in the Feature Extraction cascade) was selected as the optimal configuration of BoostPPIP. In addition, this experimental method is also used to find the optimal number of training times for BoostPPIP. Figure 6 shows that the model shows signs of overfitting at the 30th epoch, it becomes difficult to decrease the prediction error until the 50th epoch, after the model falls into a marked overfitting state. Therefore, we set up 50 times for training the BoostPPIP model.

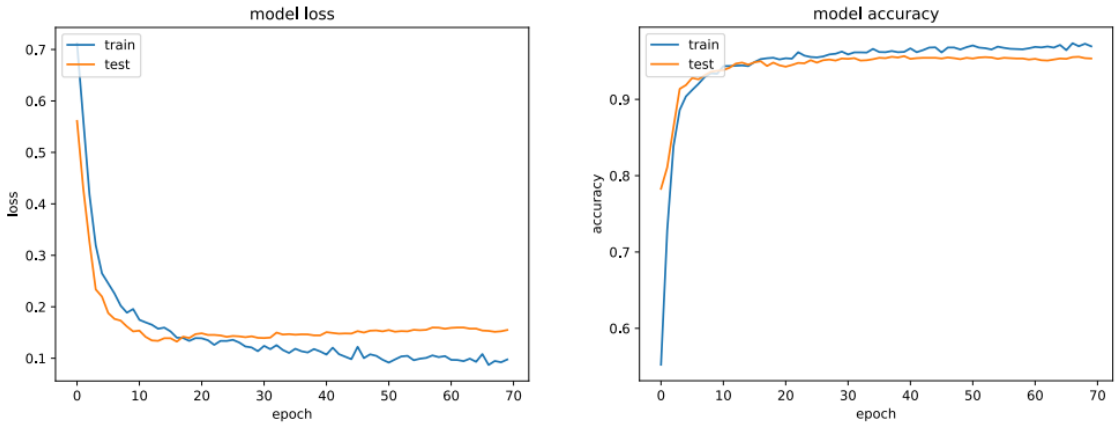

Fig. 5. The correlation between the prediction error and the number of training times of the BoostPPIP model. The experiment was performed on the set Yeast dataset. The blue line indicates the training error/training accuracy error (left side/right side), and the orange line indicates the test error/test accuracy (left side/right side)

3.4 Comparison with traditional machine learning models

The BoostPPIP model is designed based on DNNs, which can be regarded as a classifier with protein sequences as the input. To determine if using a neural network yields better performance than classifiers based on traditional ML, we compare prediction results obtained by BoostPPIP and 6 different well-known traditional ML models through 5-fold cross-validation on the Yeast dataset. The compared traditional ML classifiers including, Naive Bayes (NB), AdaBoost (Ada), SVM, Decision Tree (DT), K-Nearest Neighbors (KNN), and Random Forest (RF). In this experiment, the training of traditional ML models is processed in steps: First, the length of protein sequences in the Yeast dataset was fixed according to the same way as mentioned in 4.2, the fixed sequences are then converted to feature vectors according to the formula 2, and finally these feature vectors are used to train traditional ML models. Figure 7 shows the comparison of models on AUROC and AUPRC measurements.

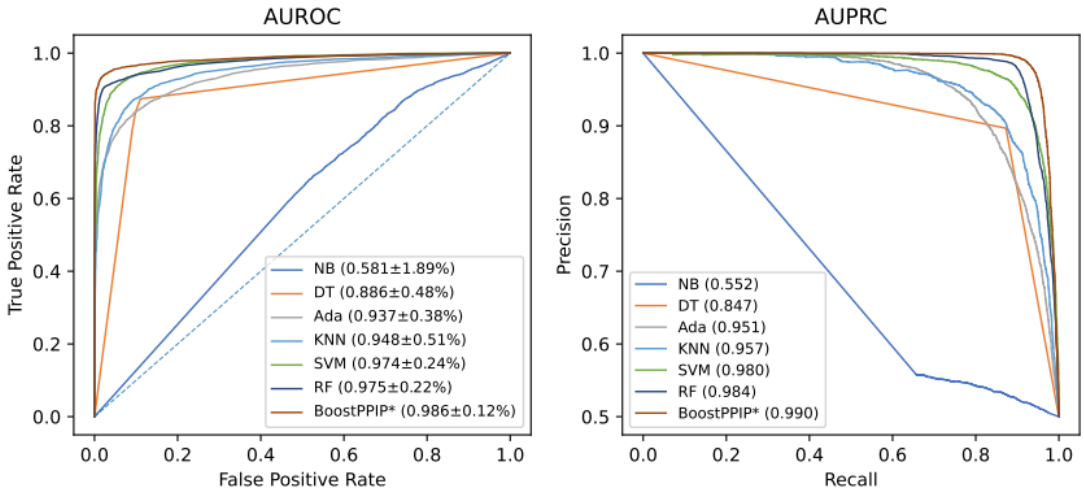

Fig. 6. Comparing the proposed model with traditional models, including NB (Naive Bayes), DT (Decision Tree), Ada (AdaBoost), KNN (K-Nearest Neighbors), SVM (Support Vector Machine), and RF (Random Forest)

We can see that BoostPPIP predicted better than NB, DT, Ada, KNN, SVM and RF. The AUROC and AUPRC values of BoostPPIP are 0.9%–40.4% and 0.7%–44.4% higher than the compaired classifiers. These results demonstrate that the use of neural networks in building classifiers is appropriate when combined with embedded features. We continue to perform other experiments to compare the performance of the proposed method with existing methods. Subsequent comparison experiments are performed on the same benchmark datasets, the same sampling method, and on the same predictive performance measurements.

3.5 Performance of methods on the Yeast dataset

Most of the methods that have been proposed for PPI prediction use the Yeast dataset to experiment and measure the prediction performance. In this experiment, we use 5-fold cross-validation on the Yeast dataset, where the mean and the standard deviation values of metrics are obtained to measure the robustness of the methods. Table 1 lists methods' prediction results. It can be seen that BoostPPIP achieved the highest prediction results with Acc, Sen, Pre, and MCC are 95.79%, 93.33%, 98.16%, and 91.7%, respectively. The BoostPPIP model helped to increase the performance prediction on the Acc, Sen, Pre, and MCC by 0.39%–8.39%, 0.63%–6.3%, 0.11%–10.34% and 0.69%–7.49%, respectively. These results show that our method is achieving good results in comparison to other existing methods.

Table 1. Performance comparison on Yeast set using 5-fold cross-validation

Methods	Acc(%)	Sen(%)	Pre(%)	MCC(%)	Ref.
Guo et al.	89.3 2.67	89.9 3.68	88.87 6.16	N/A	[6]
Guo et al.	87.4 1.38	87.3 4.68	87.82 4.33	N/A	[6]
You et al.	91.3 0.36	90.7 0.69	91.94 0.62	84.2 0.59	[9]
You et al.	93.9 0.36	91.1 0.31	96.45 0.45	88.6 0.63	[38]
Ding et al.	95.0 0.46	92.7 0.50	97.31 0.55	90.1 0.92	[21]
DeepPPI	94.4 0.30	92.1 0.36	96.65 0.59	89.00 0.62	[5]
DPPI	94.6	92.2	96.68	N/A	[15]
DeepFE-PPI	94.8 0.61	93.0 0.66	96.45 0.87	89.6 1.23	[17]
DeepSeqPPI	94.7 0.43	92.4 1.34	96.76 0.68	89.4 0.8	[16]
GTB-PPI	95.2 0.25	92.2 0.36	97.97 0.60	90.5 0.53	[14]
GcForest-PPI	95.4 0.18	92.7 0.44	98.05 0.25	91.0 0.35	[18]
BoostPPIP	**95.79 0.45**	**93.33 0.59**	**98.16 0.34**	**91.69 0.89**	**This work**

Note: The results are taken from the author's report. N/A means not reported. Bold font represents the highest value.

3.6 Performance of the methods on the Human dataset

We further compare our method with other existing methods on the Human. In this test, we also evaluate methods through 5-fold cross-validation. The prediction results of the methods are listed in Table 2. As shown in Table 2, the highest performance on Acc, Sen, and MCC metrics achieved by BoostPPIP, respectively, 99.33%, 99.74% and 98.65%. Our method hepled to increase the prediction accuracy from 0.63% to 3.73% comparing to the other methods. The sensitivity was also increased from 0.17% to 5.64%, and the MCC was increased from 1.25% to 7.45%. However, the precision achieved by BoostPPIP was 98.86%, ranking second after DeepPPI [5]. However, MCC achieved by BoostPPIP was 2.35% higher than that of DeepPPI, which shows that our method has better results in predicting both interacting and non-interacting classes. Moreover, the accuracy achieved by BoostPPIP was higher than DeepPPI's in independent testing.

Table 2. Performance comparison on Human set using 5-fold cross-validation

Methods	Acc(%)	Sen(%)	Pre(%)	MCC(%)	Ref.
Pan et al.	96.4	94.2	N/A	92.8	[22]
Pan et al.	95.7	97.6	N/A	91.8	[22]

Methods	Acc(%)	Sen(%)	Pre(%)	MCC(%)	Ref.
Ding et al.	97.6	96.6	98.3	95.1	[21]
Ding et al.	96.1	95.1	96.97	92.2	[21]
Ding et al.	95.6	94.1	96.9	91.2	[21]
Huang et al.	96.3	99.56	92.56	92.8	[37]
DeepPPI	98.1	96.95	99.13	96.3	[5]
DeepSeqPPI	97.98	97.98	97.98	96	[16]
DeepFE-PPI	98.7	98.5	98.8	97.4	[17]
Li et al.	96.1	95.2	96.6	92.5	[39]
BoostPPIP	**99.33 0.13**	**99.74 0.14**	**98.86 0.25**	**98.65 0.27**	**This work**

Note: The results are taken from the author's report. N/A means not reported. Bold fold represents the highest value.

3.7 Independent testing

Testing on cross-species PPI prediction is very important, a classifier learned on the PPI dataset of one species (e.g. Saccharomyces cerevisiae) should be applied to identify PPIs in another (e.g. Homo sapiens), meanwhile, the PPI network dataset provides some reference information for identifying PPIs from the PPIs network which has not been identified yet [13, 14]. In this test, we use all 11,188 samples of Yeast dataset as training set and 8 independent data sets as test set. The accuracy of predictions across 5 cross-species PPI datasets and 3 PPIs network datasets were used to test the generality of the methods. The Figure 4 is an illustration of a PPI network, the Cancer-specific network. The prediction results of the methods are summarized in Table 3.

Experimenting on different interactive datasets, the BoostPPIP model achieved 100% accuracy on four datasets, respectively, Celeg, Hsapi, Hpylo, Mmusc. On the Ecoli dataset, BoostPPIP achieved an accuracy of 99.88% (correct prediction of 6,946 samples out of a total of 6,954 samples), 0.12% lower than the accuracy achieved by DeepFE-PPI [17]. However, the DeepFE-PPI model was not tested on the PPIs network datasets. Compared with the other existing methods, our method and DeepFE-PPI obtained the highest accuracies. Experiment results on the PPIs networks listed in Table 3 These results indicate that our proposed model has high generalizability.

Table 3. Independent test results of the methods.

Methods	Ecoli	Celeg	Hsapi	Hpylo	Mmusc	CD9	Wnt	Cancer
Shen et al. [8]	N/A	N/A	N/A	N/A	N/A	81.25	76.04	N/A
Zhou et al. [12]	71.24	75.73	76.27	N/A	76.68	N/A	N/A	N/A
Ding et al. [21]	92.80	92.16	94.33	N/A	95.85	87.50	94.79	N/A
You et al. [20]	89.30	87.71	94.19	N/A	91.96	N/A	N/A	N/A
Huang et al. [37]	66.08	81.19	82.22	82.18	79.87	N/A	N/A	N/A
DeepPPI [5]	93.87	92.92	93.13	93.77	93.93	N/A	N/A	N/A
DPPI [15]	96.66	95.51	96.24	N/A	95.84	N/A	N/A	N/A
LightGBM-PPI [13]	92.16	90.16	94.83	N/A	94.57	93.75	92.70	N/A
DeepFE-PPI [17]	**100**	**100**	**100**	**100**	**100**	N/A	N/A	N/A
GTB-PPI [14]	94.06	92.42	97.38	N/A	98.08	93.75	95.83	N/A
GcForest-PPI [18]	96.30	96.01	98.58	N/A	99.04	**100**	97.92	**100**
Li et al. [39]	N/A	90.93	92.21	92.54	91.37	N/A	N/A	N/A
BoostPPIP	99.88	**100**	**100**	**100**	**100**	**100**	**100**	**100**

Note: The results (accuracy – %) are taken from the author's report. N/A means not reported. Bold fold represents the highest value.

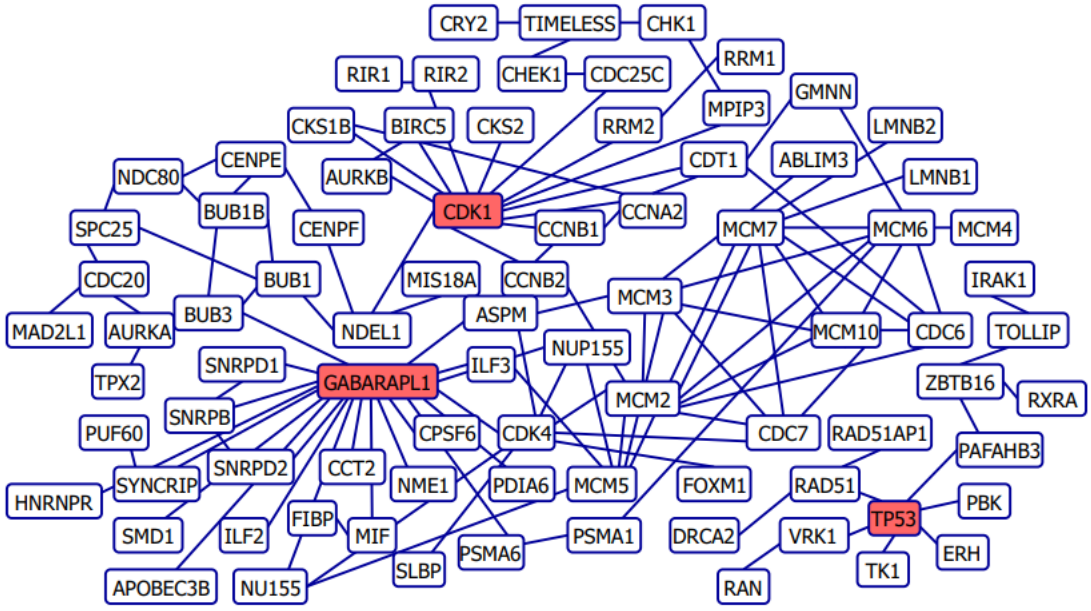

Fig. 7. Illustration of Cancer-specific network. The node represents a protein, the edge represents a prediction

4 Conclusion

In this study, we proposed a novel method for predicting PPIs directly from protein sequences data. In our method, protein sequences are converted into embeddings by a model learning the semantic relationship between amino acids. Our results have shown that the embedding features are effective in predicting protein interactions. In particular, this type of feature enhances the generality of our model in the task of determining across-species PPIs. Using the softmax function to calculate the probability distribution is not really beneficial in the case of a large vocabulary. However, since our proposed method considers only one amino acid as a word, the size of the generated vocabulary is small, specifically with only 25 elements (including 20 amino acids that have been identified in nature and 5 amino acids that have not been identified). Therefore, the softmax function is still efficient for our proposed method. In future work, we intend to incorporate our proposed model with other representation learning methods such as Doc2vec, Glove, and BERT.

Acknowledgment

This work was supported by Hue University under Grant No. DHH2023-19-03.

References

1. Y. Li, C. Huang, L. Ding, Z. Li, Y. Pan, and X. Gao, "Deep learning in bioinformatics: Introduction, application, and perspective in the big data era," Methods, vol. 166, pp. 4--21, Aug. 2019, doi: 10.1016/j.ymeth.2019.04.008.
2. Z. G. Gao, L. Wang, S. X. Xia, Z. H. You, X. Yan, and Y. Zhou, "Ens-PPI: A Novel Ensemble Classifier for Predicting the Interactions of Proteins Using Autocovariance Transformation from PSSM," Biomed Res Int, vol. 2016, 2016, doi: 10.1155/2016/4563524.
3. J. de Las Rivas and C. Fontanillo, "Protein-protein interactions essentials: Key concepts to building and analyzing interactome networks," PLoS Comput Biol, vol. 6, no. 6, pp. 1--8, Jun. 2010, doi: 10.1371/JOURNAL.PCBI.1000807.
4. L. Skrabanek, H. K. Saini, G. D. Bader, and A. J. Enright, "Computational prediction of protein-protein interactions," Mol Biotechnol, vol. 38, no. 1, pp. 1--17, 2008, doi: 10.1007/s12033-007-0069-2.
5. X. Du, S. Sun, C. Hu, Y. Yao, Y. Yan, and Y. Zhang, "DeepPPI: Boosting Prediction of Protein-Protein Interactions with Deep Neural Networks," J Chem Inf Model, vol. 57, no. 6, pp. 1499--1510, Jun. 2017, doi: 10.1021/acs.jcim.7b00028.
6. Y. Guo, L. Yu, Z. Wen, and M. Li, "Using support vector machine combined with auto covariance to predict protein-protein interactions from protein sequences," Nucleic Acids Res, vol. 36, no. 9, pp. 3025--3030, May 2008, doi: 10.1093/nar/gkn159.
7. K. H. Chen, T. F. Wang, and Y. J. Hu, "Protein-protein interaction prediction using a hybrid feature representation and a stacked generalization scheme," BMC Bioinformatics, vol. 20, no. 1, pp. 1--17, Jun. 2019, doi: 10.1186/s12859-019-2907-1.
8. J. Shen et al., "Predicting protein-protein interactions based only on sequences information," Proc Natl Acad Sci U S A, vol. 104, no. 11, pp. 4337--4341, Mar. 2007, doi: 10.1073/pnas.0607879104.
9. Z. H. You, L. Zhu, C. H. Zheng, H. J. Yu, S. P. Deng, and Z. Ji, "Prediction of protein-protein interactions from amino acid sequences using a novel multi-scale continuous and discontinuous feature set," BMC Bioinformatics, vol. 15, no. Suppl 15, pp. 1--9, 2014, doi: 10.1186/1471-2105-15-S15-S9.
10. L. Yang, J.-F. Xia, and J. Gui, "Prediction of Protein-Protein Interactions from Protein Sequence Using Local Descriptors," Protein Pept Lett, vol. 17, no. 9, pp. 1085--1090, 2010, doi: 10.2174/092986610791760306.
11. Y. Z. Zhou, Y. Gao, and Y. Y. Zheng, "Prediction of protein-protein interactions using local description of amino acid sequence," in Communications in Computer and Information Science, 2011, vol. 202 CCIS, no. PART 2, pp. 254--262. doi: 10.1007/978-3-642-22456-0_37.
12. C. Zhou, H. Yu, Y. Ding, F. Guo, and X. J. Gong, "Multi-scale encoding of amino acid sequences for predicting protein interactions using gradient boosting decision tree," PLoS One, vol. 12, no. 8, Aug. 2017, doi: 10.1371/journal.pone.0181426.
13. C. Chen, Q. Zhang, Q. Ma, and B. Yu, "LightGBM-PPI: Predicting protein-protein interactions through LightGBM with multi-information fusion," Chemometrics and Intelligent Laboratory Systems, vol. 191, pp. 54--64, 2019, doi: 10.1016/j.chemolab.2019.06.003.

14. B. Yu, C. Chen, H. Zhou, B. Liu, and Q. Ma, "GTB-PPI: Predict Protein--protein Interactions Based on L1-regularized Logistic Regression and Gradient Tree Boosting," Genomics Proteomics Bioinformatics, vol. 18, no. 5, pp. 582--592, 2020, doi: 10.1016/j.gpb.2021.01.001.

15. S. Hashemifar, B. Neyshabur, A. A. Khan, and J. Xu, "Predicting protein-protein interactions through sequence-based deep learning," Bioinformatics, vol. 34, no. 17, pp. i802--i810, Sep. 2018, doi: 10.1093/bioinformatics/bty573.

16. F. Gonzalez-Lopez, J. A. Morales-Cordovilla, A. Villegas-Morcillo, A. M. Gomez, and V. Sanchez, "End-to-end prediction of protein-protein interaction based on embedding and recurrent neural networks," in 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Dec. 2018, pp. 2344--2350. doi: 10.1109/BIBM.2018.8621328.

17. Y. Yao, X. Du, Y. Diao, and H. Zhu, "An integration of deep learning with feature embedding for protein--protein interaction prediction," PeerJ, vol. 2019, no. 6, 2019, doi: 10.7717/peerj.7126.

18. B. Yu, C. Chen, X. Wang, Z. Yu, A. Ma, and B. Liu, "Prediction of protein--protein interactions based on elastic net and deep forest," Expert Syst Appl, vol. 176, p. 114876, Aug. 2021, doi: 10.1016/J.ESWA.2021.114876.

19. C. Xu, L. Jiang, Z. Zhang, X. Yu, R. Chen, and J. Xu, "An Integrated Prediction Method for Identifying Protein-Protein Interactions," Curr Proteomics, vol. 17, no. 4, pp. 271--286, Jun. 2020, doi: 10.2174/1570164616666190306152318.

20. Z. H. You, K. C. C. Chan, and P. Hu, "Predicting protein-protein interactions from primary protein sequences using a novel multi-scale local feature representation scheme and the random forest," PLoS One, vol. 10, no. 5, pp. 1--19, May 2015, doi: 10.1371/journal.pone.0125811.

21. Y. Ding, J. Tang, and F. Guo, "Predicting protein-protein interactions via multivariate mutual information of protein sequences," BMC Bioinformatics, vol. 17, no. 1, p. 398, Dec. 2016, doi: 10.1186/s12859-016-1253-9.

22. X. Y. Pan, Y. N. Zhang, and H. bin Shen, "Large-scale prediction of human protein-protein interactions from amino acid sequence based on latent topic features," J Proteome Res, vol. 9, no. 10, pp. 4992--5001, 2010, doi: 10.1021/pr100618t.

23. T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," Jan. 2013, {[}Online{]}. Available: http://arxiv.org/abs/1301.3781

24. J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532--1543. doi: 10.3115/v1/D14-1162.

25. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Oct. 2018.

26. S. Ruder, "An overview of gradient descent optimization algorithms," Sep. 2016.

27. A. MacDougall et al., "UniRule: a unified rule resource for automatic annotation in the UniProt Knowledgebase," Bioinformatics, May 2020, doi: 10.1093/bioinformatics/btaa485.

28. S. Skansi, Introduction to Deep Learning. Cham: Springer International Publishing, 2018. doi: 10.1007/978-3-319-73004-2.

29. S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," Feb. 2015.

30. N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," 2014.

31. A. M. Fred Agarap, "Deep Learning using Rectified Linear Units (ReLU)." {[}Online{]}. Available: https://github.com/AFAgarap/relu-classifier.

32. D. P. Kingma and J. Lei Ba, "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION."

33. M. Abadi et al., "TensorFlow: A system for large-scale machine learning," May 2016, {[}Online{]}. Available: http://arxiv.org/abs/1605.08695

34. F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825--2830, 2011.

35. I. Xenarios, "DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions," Nucleic Acids Res, vol. 30, no. 1, pp. 303--305, Jan. 2002, doi: 10.1093/nar/30.1.303.

36. W. Li and A. Godzik, "Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences," Bioinformatics, vol. 22, no. 13, pp. 1658--1659, Jul. 2006, doi: 10.1093/bioinformatics/btl158.

37. Y. A. Huang, Z. H. You, X. Gao, L. Wong, and L. Wang, "Using Weighted Sparse Representation Model Combined with Discrete Cosine Transformation to Predict Protein-Protein Interactions from Protein Sequence," Biomed Res Int, vol. 2015, 2015, doi: 10.1155/2015/902198.

38. Z. H. You, X. Li, and K. C. Chan, "An improved sequence-based prediction protocol for protein-protein interactions using amino acids substitution matrix and rotation forest ensemble classifiers," Neurocomputing, vol. 228, no. October, pp. 277--282, 2017, doi: 10.1016/j.neucom.2016.10.042.

39. Y. Li et al., "Robust and accurate prediction of protein--protein interactions by exploiting evolutionary information," Sci Rep, vol. 11, no. 1, pp. 1--12, 2021, doi: 10.1038/s41598-021-96265-z.

40. S. Altschul, "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs," Nucleic Acids Res, vol. 25, no. 17, pp. 3389–3402, Sep. 1997, doi: 10.1093/nar/25.17.3389.

41. E. W. Sayers et al., "Database resources of the national center for biotechnology information," Nucleic Acids Res, vol. 50, no. D1, pp. D20–D26, Jan. 2022, doi: 10.1093/nar/gkab1112.

42. H.-N. Tran, Q. N. P. Xuan, T.-T. Nguyen, "DeepCF-PPI: improved prediction of protein-protein interactions by combining learned and handcrafted features based on attention mechanisms," Applied Intelligence, Dec. 2022, doi: https://doi.org/10.1007/s10489-022-04387-2