Identification of All-to-All Protein-Protein Interactions Based on Deep Hash Learning

Yue Jiang
Fujian Normal University
Lin Shen
Fujian Normal University
Jie Lin (linjie891@163.com)
Fujian Normal University

Research Article

Keywords: Protein-protein interaction, Deep learning, Binary Hash code, Binary search, Hamming distance

Posted Date: September 2nd, 2021

DOI: https://doi.org/10.21203/rs.3.rs-778066/v1

RESEARCH

Identification of All-to-all Protein-protein Interactions Based on Deep Hash Learning

Yue Jiang, Lin Shen and Jie Lin*

*Correspondence:
linjie891@163.com
College of Computer and Cyber Security, Fujian Normal University, 350108 Fuzhou, PRChina
Full list of author information is available at the end of the article

Abstract

Background Protein-protein interaction (PPI) is vital for life processes, diseases treatment and new drugs discovery. The computational prediction of PPI is well accepted for its inexpensive and efficient nature comparing to the wet-lab experiment. When a new protein comes, one try to find whether there is any PPI relationship between this new protein and existing proteins, the current computational prediction methods usually compare this new protein to existing proteins one by one in pairwise. This is time comsuming.

Results We proposed an more efficient model, Deep Hash Learning Protein-and-Protein Interaction (DHL-PPI) model, to predict all-to-all PPI relationship on a database. First, DHL-PPI encodes a protein sequence into a binary Hash code based on the features extracted from sequences by using deep learning technique. This encoding scheme enables the PPI discrimination problem to be a much simpler searching problem. A protein with a binary code can be regarded as a number. In the prescreen of PPI prediction stage, the string match problem of searching a string against a database with M proteins can be turned into a much more simpler problem: to find a number inside an sorted array with length M. This prescreen process narrows down proteins inside the whole database into a much smaller candidate set for further confirmation. At last, DHL-PPI uses the Hamming distance to determine the final PPI relationship.

Conclusions The experimental results confirmed that DHL-PPI is feasible and effective. Using a dataset with strictly negative PPI examples of four species, DHL-PPI is superior or competitive to the other state-of-the-art methods in terms of precision, recall or F1 score. Furthermore, in the prediction stage, the proposed DHL-PPI decrease the usual time compexity of $O(M^2)$ to $O(MlogM)$ for predicting all-to-all PPI interactons between any pairs in M proteins on a database. A protein database can be stored in the proposed encoding scheme and waited to be searched, which is a potential novel encoding scheme to cope with current searching problem for a large volume of database.

Keywords: Protein-protein interaction; Deep learning; Binary Hash code; Binary search; Hamming distance

1 Background

Most vital molecular processes and biochemical reactions need to be completed by Protein-Protein Interaction (PPI) in biological cells, such as intracellular communications, signal transduction and gene regulation. The identification of PPI is important in life process research, diseases treatment and new medicines development [1].

To identify PPI in the wet-lab experiment is costly and time comsuming. Although current high throughput experiments are much more high-efficiency and

money-saving, for example, yeast 2-hybrid (Y2H) [2], mass spectrometric protein complex identification (MS-PCI) [3], Co-Immunoprecipitation (Co-IP) [4],and Trandem affinity purification-mass spectrometry (TAP-MS) [5]. Whereas these wet lab methods are still expensive and the results have many FPs and FNs [6]. The computational identification methods are usually used to prescreen and predict PPI before the wet-lab experiment due to its relatively convenient, cheap and efficient nature.

The protein sequence is essential for PPI prediction. The existing PPI recognization methods based on sequence can be divided into three kinds: co-occurrence based method, pattern matching based method and machine learning based method.

The method based on co-occurrence [7] judges the interaction between a pair of proteins by counting the number of times of their co-occurrence. Bunescu et al. [8] recognize the protein-protein interaction by extracting frequent patterns.

The pattern matching based method searches the potential PPI by establishing certain pattern rules. Due to the limitation of frequent patterns, this kind of method generally achieves high recall but low precision and poor generalization [9]. Fundal [10] proposes the dependency relationship based on the structure of syntax rules of sentences. Temkin [11] distinguishes PPI through the sentence analyzer with the rules of grammar generation. This method needs to construct patterns manually, thus it is inefficient, time-consuming and labor-intensive [12]. Moreover, because of the diversity of relational patterns of PPI, the predefined rules cannot contain all PPI rational patterns. The machine learning based method mainly has two groups: kernel function based method and feature based method. Haussler et al. [13] present the convolution kernel which can be used in discrete structure. The string kernel is proposed by Lodhi H. et al. [14], which takes the inner product of the word substring with a specific length in the feature space as the calculation method of the kernel function. This kind of method analyzes PPI according to the grammar, syntax and dependency of a single sentence. However due to the complex grammar and the indirect discription of PPI, the conclusion it draws may not be accurate enough. This kind of method takes the various properties of protein as feature, for instance, physicochemical property such as hydrophobicity profiles, amino acid composition, and domain composition, genomic features such as gene neighbouring, and network topology-based features [15].

The machine learning based PPI prediction method can be classified into supervised and unsupervised depending on whether the used data has labels or not. The supervised method for PPI prediction learns mapping functions from labeled PPI data, then predicts the PPI. The frequently used machine learning methods are Decision Trees [16, 17], SVM [18, 19, 20, 21], ANN [22, 23, 24], KNN [25] and Naive Bayes [26] and so on. The unsupervised PPI prediction method learns the intrinsic feature representation of the unlabeled data, and then carry on deeper analysis. Clustering method such as K-means [27, 28] often is applied on it.

Deep learning, a typical machine learning methods, has been widely used for PPI prediction and has obtained remarkable achievements. Zhao et al. [29] take nine properties of amino acids, for example, Relative Exterior Solvent Accessible area (RESA) and Hydropathy Index (HI, two versions), as feature representation. And they train long-short term memory networks(LSTM) [30] to accomplish the correct

prediction of interface residue pairs from two monomer proteins. Li et al. [31] firstly substitute corresponding random numbers for amino acids in the protein sequence to complete sequence coding. Then they map amino acids to dense vector like word2vec through an embedding layer. Next, they mine the long term dependence between amino acids by using convolution neural network (CNN) [30] and LSTM. Finally the learnt features are inputted into the full connect layer to predict PPI. Somaye et al. [32] split the protein into subsequences, and use these subsequences to carry on multiple sequence alignment through PSI-BLAST to obtain protein profile, which can be learned by convolutional module and random projection module to identify PPI. Sun [33] combines Autocovariance(AC) and Conjoint triad(CT) as feature representation of protein, and learns PPI prediction based on sequence using stacked autoencoder(SAE) [30], where AC uses seven physicochemical properties of amino acids and CT describes the composition of amino acids. Similarly, Du et al. [34] express the composition of amino acids using three methods, Composition, Transition and Distribution, meanwhile using Quasi-Sequence-Order Descriptors, Amphiphilic Pseudoamino Acid Composition(APAAC) to represent the physical and chemical properties of protein. Then they learn protein representation through two different type of DNN schemes. Sunil [35] presents domain-based method, DeepInteract, which takes the protein domain frequency as feature, using 2,971 domains to represent protein sequence specifically, and predict PPI through DNN.

The other deep learning method, the graph based method constructs networks according to proteins and their relationships, where nodes mean proteins and edges are interactions. Huang [36] establishes protein adjacency matrix to represent the interaction between proteins in graph. He stimulates the evolution process of PPI network implicitly by using SAE, finally predicting PPI by regularized Laplacian kernel. But this kind of method has a problem that the network is too sparse because of the lackness of existing PPI. Fang et al. [37] represent the protein sequence by using Conjoint triad (CT). They further learn the embedding of protein sequence based on the feature of protein and local information of sequence through signed variational graph autoencoder (S-VGAE) to improve the robustness of model for sparse datasets. And finally they accomplish the PPI prediction based on graph.

Moreover, there are numerous protein sequences in the natural environment and it is too expensive to verify all interactions among all pairs between proteins, which cause the incompleteness of current known PPI networks [38]. Given a dataset with M protein sequences, in order to predict the potential PPI among them, the current available methods compare each protein to the other proteins one by one in pairwise, which need to compare $\frac{M(M-1)}{2}$ times, that is, with $O(M^2)$ time complexity. Our proposed DHL-PPI method only needs $O(Mlog_2M)$ time complexity.

The proposed DHL-PPI method transforms a protein sequence into a binarized Hash code by using deep learning techniques. In the prescreen stage, DHL-PPI takes part of a binarized Hash query code which can be regarded as a number, to search against the same part of binarized Hash ontic codes in a database with M proteins, which can be regarded as an array with M numbers inside. Then the string matching problem is turned into a problem of finding exact number inside an array to filtering out irrelevant proteins. After this prescreen stage, the whole database can be narrowed down to a much smaller candidate set. This smaller candidate

set is further processed by calculating the Hamming distance between the query Hash code and ontic Hash codes to determine whether there is PPI relationship or not. In average, the proposed DHL-PPI only needs $O(Mlog_2M)$ time complexity to predict all PPI in a database with M proteins. This reduces the time compexity and improves the search performance greatly.

The experimental results confirm that DHL-PPI is feasible and effective. In the dataset with strictly negative PPI examples of four species data, DHL-PPI is superior or competitive to the other methods no matter in terms of precision, recall or F1 score.

Our experimental results demonstrate that the proposed DHL-PPI is suitable to predict PPI accurately in much simpler, faster and efficient way. A database can be stored in the proposed encoding method and wait to be searched which is able to decrease searching time greatly for a large volume of database.

The remaining of the paper is organized as follows. In Section 2, we show our experiments and results, including the introduction of data sets, evaluation criteria, results, comparison with the other state-of-the-arts methods. In Section 3, we present the discussion of experimental results. In Section 4, we draw our conclusions. In Section 5, the detail methods with algorithms and analysis are presented.

2 Experiments and Results

The experimental data set contains strictly known positive PPI and strictly negative PPI samples. In this benchmark data set, we verify the validation of the proposed DHL-PPI method.

2.1 Dataset

The data set is from the Human Protein References Database (HPRD) [39] (http://www.hprd.org/, release 7_20070901). Pan et al. [40] use it for PPI prediction, where all positive PPI relationship and negative PPI relationship are confirmed by wet lab experiment, also called Pan's dataset.

The dataset contains four species which are (1) C.elegan dataset; (2) Drosophila dataset; (3) E.coli dataset; and (4) Human dataset. The negative set of the dataset is generated based on the hypothesis that two proteins in different cellular compartments do not have any interaction, more detail criteria are described in Yang [37]. Therefore, the negative set of the dataset [40] is regarded as strictly negative samples in the field. The number of proteins and positive samples and negative samples of these four species in the dataset are shown in Table 1.

Dataset	Number of Proteins	Number of Positive Samples	Number of Negative Samples
C.elegan dataset	1734	2877	1670
Drosophila dataset	5624	19712	14900
E.coli dataset	1528	5576	4031
Human dataset	7803	31761	25203

Table 1 Four species in the dataset.

In this work, we divided the dataset containing positive samples and negative samples into training set and testing set with ratio of 9 : 1. We double the dataset through switching the order of input protein pairs, that is, by transforming the sequence A-sequence B-label to sequence B-sequence A-label. Thus the used samples are doubled in all four species.

2.2 Evaluation Criteria

In this work, we use precision, recall and F1 to evaluate our models. The definitions are presented as follows respectively:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 * precision * recall}{precision + recall}$$

where TP stands for True Positive; FP is False Positive; and FN denotes False Negative.

2.3 Experimental Results and Analysis

In this section, we evaluated the performance of DHL-PPI over the above introduced dataset by using recall, precision and F1 score.

The experimental results on the dataset [40] are shown in Table 2. Observe Table 2, one can find that the precision, recall and F1 are all very good.

Dataset	Recall	Precision	F1-score
C.elegan dataset	0.981	1.000	0.990
Drosophila dataset	0.981	0.998	0.990
E.coli dataset	0.962	0.987	0.975
Human dataset	0.963	0.984	0.973

Table 2 The performance of DHL-PPI on the dataset

2.3.1 Comparison with Other Methods

On the dataset, DHL-PPI is tested on four species data sets respectively containing the strictly negative examples. Table 3 show the results by comparing our method to Yang's work [37] and DNN-PPI [31] on these four sepecies, C.elegan, Drosophila, E.coli and Human respectively.

At the first glance of Table 3, one can claims that all methods are very impressive. In these four species, the smallest value is 0.942 which is the recall value of DNN-PPI method in Table 3. Neverthless, the performance of this recall value of 0.942 is very good in common sense.

The experimental results show that the performance of our proposed DHL-PPI method is competitive to that of Yang's work [37]. And the proposed DHL-PPI is better than DNN-PPI proposed by Hang in all species sets [31]. These results show that our proposed DHL-PPI is suitable to predict PPI in this benchmark data set.

3 Discussion

More effective computational prediction of all-to-all protein-protein interaction (PPI) relationship methods are wanted to avoid costly wet-lab experimental efforts. We proposed an effective and feasible method, deep Hash learning PPI prediction

Dataset	Method	Recall	Precision	F-score
C.Elegan	DHL-PPI	0.981	1	0.99
C.Elegan	Yang [37]	0.992	0.993	0.993
C.Elegan	DNN-PPI [31]	0.981	0.992	0.986
Drosophila	DHL-PPI	0.981	0.998	0.99
Drosophila	Yang	0.996	1	0.998
Drosophila	DNN-PPI	0.9686	0.9995	0.9837
E.coli	DHL-PPI	0.962	0.987	0.975
E.coli	Yang	0.984	0.994	0.989
E.coli	DNN-PPI	0.942	0.975	0.958
Human	DHL-PPI	0.963	0.984	0.973
Human	Yang	0.98	0.995	0.988
Human	DNN-PPI	-	-	-

Table 3 The performance comparisons with different methods on the dataset

(DHL-PPI) to predict PPI. Our experimental results suggest that DHL-PPI is not only an effective but also faster than the other state-of-the-arts methods.

In a benchmark dataset with strictly negative and positive PPI examples of four species data, DHL-PPI is superior or competitive to the other methods no matter in terms of precision, recall or F1 score.

One might argue that it is hard to draw conclusions based on only one data. However, there are four species inside this dataset which contain more than thousands of samples which can be regarded as four sub data sets. Furthermore, this data set is a benchmark data set in the field.

The proposed encoding scheme of DHL-PPI by transferring a protein sequence into a binary Hash code turned a complicated sequence matching problem into a much simpler and faster problem of finding number in an array. This process excludes the irrelevant proteins out in the prescreen stage quickly. This also suggsets that DHL-PPI is a potential novel encoding scheme to cope with current searching problem in large volume of databases.

The proposed DHL-PPI turned a complex sequence matching problem with time complexity of $O(M^2)$ into a much lower time complexity $O(MlogM)$. This suggests that the proposed DHL-PPI is feasible in a database with large volume of data.

Together, The experimental results suggests that DHL-PPI is feasible and effective in predicting all-to-all PPI relationships on a database in faster time.

4 Conclusion

In this paper, we proposed a protein interaction prediction model, Deep Hash Learning Protein to Protein Interaction (DHL-PPI) prediction which includes an encoding scheme to transform a protein sequence into a binary Hash code and a prediction scheme. Firstly we encoded a protein sequence into an integer sequence. Secondly, the deep learning technique is applied to generate a better embedding representation for each amino acid. Thirdly, the embedding representation is turned into binary Hash codes, namely, ontic Hash and query Hash codes respectively. In the prescreen process of PPI discrimination stage, the string matching problem is turned into a much more faster and simpler problem of finding exact number inside an array. This prescreen process filters out irrelevant proteins inside a database resulting a much smaller candidate set for further confirmation. In the final step of PPI discrimination, DHL-PPI calculates Hamming distance between query Hash code and ontic Hash codes to determine the final PPI relationship set.

We verify the proposed DHL-PPI on a benchmark data set with strictly negative and positive examples of four species. The experimental results with high recall, precision and F1 score confirmed that DHL-PPI is an effective method to predict PPI relationship. Furthermore, in the prediction stage, the proposed DHL-PPI decrease the usual time compexity of $O(M^2)$ to $O(MlogM)$ for predicting all-to-all pairs of PPI relationships for M proteins on a database. The encoding scheme which turns a string matching problem into a much simpler problem of finding a number inside an array saves a lot of searching time. It is also a potential encoding scheme for a database and waited to be searched. This encoding scheme can be applied to the other sequences, i.e., DNA sequences and other texts also.

5 Methods

We proposed a protein-protein interaction relationship prediction model, deep Hash learning PPI (DHL-PPI), which contain an encoding scheme and a prediction scheme. There are three main steps in the method: preprocessing (Section 5.1), DHL encoding model(Section 5.2), and PPI prediction model(Section 5.4). Beside these three steps, in this section, we also explain two other important knowledges in DHL-PPI model: loss functions (Section 5.3) and associated prediction algorithms (Section 5.5).

5.1 Data Preprocessing

In data preprocessing stage, first, an amino acid in protein sequence is transferred into an interger as shown in Table 4 by using Tokenizer API which is introduced in Keras [41]. And then, the sequence consisting of integers is processed by zero-pad method [41] into a sequence with fixed length. The reason is that the neural convolution in the next step only accept sequences with fixed length. In this study, the protein sequences with less than 5000 amino acids in the dataset are zero-padded into 5000 amino acids. These sequences are fed into CNN in the next DHL model.

Ala	Gly	VaL	Ile	Leu	Phe	Pro	Tyr	Met	Thr
(A)	(G)	(V)	(I)	(L)	(F)	(P)	(Y)	(M)	(T)
1	2	3	4	5	6	7	8	9	10
Ser	His	Asn	Gln	Trp	Arg	Lys	Asp	Glu E	Cys
(S)	(H)	(N)	(Q)	(W)	(R)	(K)	(D)	(E)	(C)
11	12	13	14	15	16	17	18	19	20

Table 4 The Integer encoding of amino acids

5.2 The DHL Encoding Model

The DHL-PPI contains two main parts: DHL encoding model and PPI prediction model. The goal of DHL model is to encode a pair of protein A and B into query code and ontic code respectively. Then the PPI prediction model determines whether there is PPI relationship between two proteins A and B by calculating the Hamming distance between a query code and an ontic code.

Inside DHL encoding model, there are four parts: the embedding layer, convolution blocks, random projection and range control module, as shown in Figure 1. The used DHL model makes an improvement over a basic network proposed in [42]. First, the embedding layer encodes a protein sequence into a vector. Then, the convolution

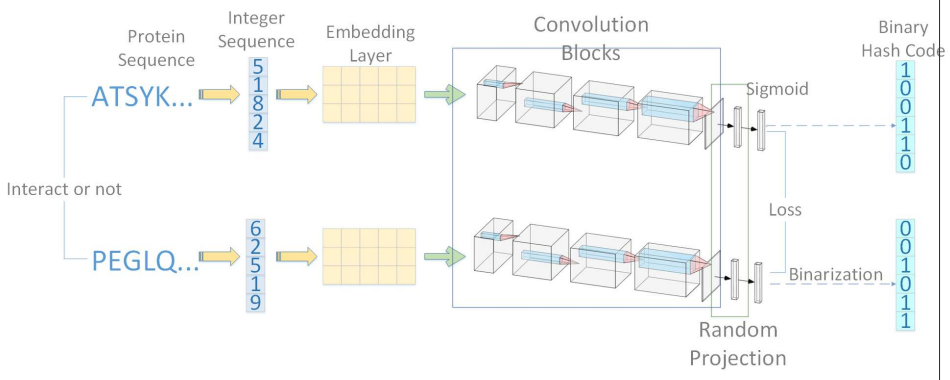

Figure 1 The architecture of DHL-PPI

blocks contain four CNN blocks with pairwise inputs. And each CNN contains an embedding layer which trying to generate a better representation for features of protein sequences. Thirdly, the random projection module transfers a vector representaion into an ontic code and a query code respectively. These two random projections for input proteins A and B are untrainable and they do not share any single same parameters. Lastly, the range control block is a sigmoid function which limit the output bits inside the range of [0,1]. The first two parts and the last part use siamese networks which share the same network parameters for input proteins A and B. After these four steps, a protein sequence is turned into two different codes, binarization Hash query code and binarization Hash ontic code respectively. This DHL model is implemented by using Keras API in the tensorflow2.0 framework.

5.2.1 The Embedding Layer for Amino Acid

The goal of the embedding layer is to learn a better vectorization representation of an amino acid, that is, an optimal way to encoding a protein sequence. This embedding layer maps each protein sequence into a $L * dim$ matrix, where L denotes the length of sequence and dim is the number of dimension of embedding representation of amino acid. Each row of the matrix indicates the embedding representation of each amino acid. Before training, L is initialized to the length of a protein sequence, and the embedding representation of amino acid is initialized randomly. In the experiment, L is initialized to 5000. In the training process, the embedding layer and convolution blocks are trained together to generate an effective vector to represent a protein sequence. The final value of embedding representation of amino acid is determined by the model automatically.

5.2.2 Convolution Blocks

The convolution blocks contain four CNN blocks with pairwise inputs. And each CNN contains an embedding layer trying to generate a better representation for features of protein sequences.

Each CNN block conducts four steps. Firstly, it carries out one-dimensional convolutions. Secondly, it activates the output through ReLU function. Thirdly, it uses batch normalization to reduce the training difficulty. And the final part is pooling. The first three convolution blocks adopt one-dimensional average pooling, while the last convolution block uses global average pooling. The parameters and settings of each block are shown in Table 5.

Block	Number of Convolution kernel	Kernel size	Pooling	Stride
ConvBlock1	64	5	1D Average Pooling	1
ConvBlock2	128	7	1D Average Pooling	1
ConvBlock3	256	9	1D AveragePooling	1
ConvBlock4	512	15	1D Global Average Pooling	1

Table 5 The parameters and settings of Convolution Blocks

5.2.3 Random Projection Module

The DHL model contains two different random projection modules. These two random projections are untrainable and they do not share any single parameter. One random projection module transfers a sequence into an ontic code, the other random projection module transfers the same sequence into a query code. Each random projection module contains a sub-network of full connected layer with 64 neurons. These two random projection modules accept the same input from previous convolution module. And then this same input are mapped into an ontic code and a query code respectively. The ontic and query codes are 64-dimension vectors. In order to do this, the weights of these two random projection modules are set to be different and untrainable. These untrainable characteristics reduce the number of trained parameters and further speed up the training process and finally avoid the risk of over-fitting.

5.2.4 Range Control Module

The range control module, which is the last part of DHL, uses a sigmoid function to activate the 64-dimension vector representations from the previous step. The sigmoid function is used to limit the output inside the range of [0,1]. It combines with the Hash constraint loss to minimize the quantization loss $x - sign(x)$. The finally output is binarized by the sign function (Equation 1), that is, the sigmoid is used to approximate the sign function.

$$sign(x) = \begin{cases} 1 & \text{if x} \geq 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

5.3 Loss Functions of the DHL-PPI Model

There are three kinds of losses in the DHL-PPI model, which are discrimination error loss, Hash constraint loss and bit balance loss.

Discrimination error loss occurs when using query Hash code to search against candidate ontic Hash codes to determine whether there is interaction relationship between two proteins or not. Hash constraint loss occurs in the stage of binarization of Hash codes. Bit balance loss occurs in the stage of optimizing the generalized Hash codes to obtain a balance 0 and 1 distribution.

5.3.1 Discrimination Error Loss

If there is an interaction between two proteins, the distance between their corresponding Hash codes should be closer, otherwise, the distance between Hash codes should be larger. Specifically, in DHL-PPI model, when training the model, if two proteins have an known interaction, the distance threshold in_{dist} between them is set to 2 or smaller (in the range of [0, 2]). If two proteins do not have any known interaction (or strictly no interaction is confirmed), the distance threshold $unin_{dist}$ should be greater or equal to 12. The definition of the discrimination error loss function is shown in Equation 2:

$$discrimLoss = label \times max(dist - in_{dist}, 0) + (1 - label) \times max(12 - unin_{dist}, 0) \quad (2)$$

$$dist = \sum_{i=1}^{N} |hashcode_i^{ont} - hashcode_i^{que}| \quad (3)$$

where *label* is the label of a protein pair and N represents the dimension of Hash code. If there is interaction between two proteins, then $label = 1$, otherwise $label = 0$. The in_{dist} and $unin_{dist}$ represent the minimum and the maximum distance between protein Hash codes, which is set to 2 and 12 respectively. Equation 3 gives the way to calculate *dist* in Equation 2. In Equation 3, the *dist* is the Manhattan distance between two proteins' Hash codes, query code and ontic code. When each bit of Hash code is close to 0 or 1, *dist* is close to Hamming distance. N is set to 64 in this study. $hashcode_i^{ont}$ and $hashcode_i^{que}$ are the i-th bit of the ontic code corresponding to protein A and the query code corresponding to protein B before binarization respectively.

5.3.2 Hash Constraint Loss

Hash constraint loss occurs in the stage of binarization of Hash codes. In some related study [43, 44, 45], the tahn(tangent) function is used to binarize the Hash code in training phase. The proposed DHL-PPI uses the sigmoid function to limit the model output to [0,1] first. And then it introduces the Hash constraint loss shown in Equation 4 to minimize the constraint loss. Specifically, it makes each Hash code as close as possible to 0 or 1 before binarization. In this way, the total Hash code loss function is minimized.

$$hashLoss = hashLoss_{ont} + hashLoss_{que} \quad (4)$$

$$hashLoss_{ont} = max(quan_{thresh}^2 \times N - \sum_{i=1}^{N} (hashcode_i^{ont} - quan_{thresh})^2, 0)$$

$$hashLoss_{que} = max(quan_{thresh}^2 \times N - \sum_{i=1}^{N} (hashcode_i^{que} - quan_{thresh})^2, 0)$$

where $hashLoss_{ont}$ and $hashLoss_{que}$ denote the Hash constraint loss of ontic code and query code respectively. The total Hash constraint loss $hashLoss$ is the sum of these two values. $quan_{thresh}$ is the quantization threshold, which is set to 0.5 in this case. When each bit of the Hash code is close to 0 or 1, the loss is smaller. Otherwise, when each bit is 0.5, the loss will be the largest.

5.3.3 Bit Balance Loss

When encoding a protein sequence into a binary Hash code in DHL-PPI, the ideal situation is that the probability for each bit turned to be 0 or 1 is the same. However, the real situation seldom shows up. The bit balance loss is used to capture loss between ideal and real state. The bit balance loss function is shown in Equation 5.

$$bitLoss = bitLoss_{ont} + bitLoss_{que} \tag{5}$$

$$bitLoss_{ont} = (mean(hashcode^{ont}) - 0.5)^2$$

$$bitLoss_{que} = (mean(hashcode^{que}) - 0.5)^2$$

where $bitLoss_{ont}$ and $bitLoss_{que}$ represent the bit balance loss of ontic code and query code respectively; $hashcode^{ont}$ and $hashcode^{que}$ represent a binary Hash ontic code and a binary Hash query code respectively. The total bit balance loss is the sum of these two values. When the average of the Hash code is $\frac{0+1}{2} = 0.5$, the bit balance loss is the minimum. In the other way, when the generated binary Hash codes are balanced , that is, the possibility of binarized Hash code turned up to 1 or 0 is the same, the total bit loss is minimal. Otherwise, when the average is far away from 0.5, the loss will be the largest.

5.3.4 Total Loss Function

The total loss is the sum of the above three mentioned loss functions, namely, the discrimination loss, the Hash constraint loss and the bit balance loss, which is shown in Equation 6. This loss function is incorporated into DHL-PPI to train the model iteratively to optimize the final output, binarized Hash codes.

$$Loss = discrimLoss + hashLoss \times W + bitLoss \tag{6}$$

where $discrimLoss$ is the discrimination error loss; $hashLoss$ is the Hash constraint loss; $bitLoss$ is the bit balance loss; and W is the weight of $hashLoss$.

The form of the sigmoid function is $f(z) = \frac{1}{1+exp(-z)}$. Its derivative is $f'(z) = f(z)(1 - f(z))$. When $f(z)$ is close to 0 or 1, its derivative tends to 0. DHL-PPI intends to reduce the total errors during training. It would be ideal if the output $f(z)$

is close to 0 or 1. This would lead to the disappearance of the gradient. Therefore we multiply the *hashLoss* to the weight W, where W is set to $2/N$, and N is the dimension of the Hash code during training ($N = 64$ in this study) . In the earlier stage of the training, the discrimination loss and bit balance loss play more important role in model fitting. In the later training period, with the decreasing of the discrimination loss, the Hash constraint loss come into play and constraint the Hash code to 0 or 1.

5.4 PPI Prediction Model

The prediction of PPI relationship is also called PPI discrimination or PPI identification in general. The state-of-the-arts deep learning methods usually contain two parts, one is encoding and the other is discriminating. The proposed DHL-PPI only utilizes encoding part and it does not use the usual discriminater in the deep learning model. In the other word, the deep learning techniques in DHL-PPI is only use to generate the reasonable binarization Hash codes (query code and ontic code) for a protein sequence. In the prediction phase, DHL-PPI first conducts a prescreen process to narrow down the whole database. And then it compares a query code to an ontic code by calculating the Hamming distance between them.

In the prescreen stage, the proposed DHL-PPI searches a binarized Hash query code against all binarized Hash ontic codes in a database. Because they are all binarized codes, these digital bits and partial of them can be regarded as a binary number. The problem of searching a protein sequence against all proteins (i.e, M number of protein sequence) in a database is turned into a problem of finding exact binary number inside an array with length of M. This process greatly narrows down the waited to be confirmed candidates in a quick way.

In the final step of DHL-PPI in PPI prediction, it compares a binarized Hash query code to an ontic code by calculating Hamming distance as measurement. This Hamming distance measurement is used as criteria to determine whether there is a realtionship between two proteins or not. This is another advantage of the proposed DHL-PPI which is faster and simpler than the other methods. Generally, calculating a Hamming distance is much faster than a discriminater in deep learning model.

Let's examine the Hamming distance comparison and the threshold used in this study. In the information theory, the Hamming distance between two strings of equal length is the number of different characters at the corresponding positions of these two strings. In the other words, it's the number of characters needed to be replaced by converting one string to another string. In this step, the most important thing is to define the distance threshold, d, which determine whether there is interaction between two proteins or not. Specifically, if a Hamming distance is greater than d, it is defined that there is no interaction between two proteins, otherwise, there is. Here in this study, the distance threshold d is set to be 4 ($d = 4$), that is, when the distance between two codes is greater than 4, we regard that there is no interaction between them, otherwise, there is.

Given a protein sequence P and a protein database D which contain M number of proteins. The problem is to determine whether the given P has any interaction (relationship) with any other protein inside the database D. The intuitive way is to compare a given protein sequence P with all proteins inside the database D one by

one which requires M number of comparison (the number of proteins inside D). If one want to determine whether there is any interaction among all proteins inside a database D, it usually needs to compare $M(M-1)$ times because all M proteins inside database D needed to be compared $M-1$ times.

In the prescreen stage of PPI, we transfer the sequence comparison problem into a much simpler problem of finding a number inside an array. Next, a binary query code of the protein P to search on the binary ontic codes is conducted by calculating Hamming distance. Using the proposed DHL-PPI prediction method, the comparison time is decreased greatly. In the proposed DHL-PPI prediction method, it can be done in $O(M \log_2 M)$. The detail implementation is explained in next subsection.

5.5 The PPI Prediction Algorithms

Here, we introduce the PPI prediction algorithms. Algorithm 1 shows the overall process of PPI prediction. There are three input parameters in the algorithm. One is the Hamming distance threshold, d, which determines whether there is a relationship between two proteins. The second one is the ontic codes of a protein database, D, which are generated in the previous DHL model. The third is the query code of a protein database, Q, which is also generated in the previous DHL model. The output parameter $PPIset$ contains the predicted PPI relationship.

Let's examine the algorithm line by line. In Line 1, set $PPIset$ is initialized to empty. And M is assigned to the number of proteins in the query codes of database Q. In Line 2, indexes of ontic codes of database D are built. There are total of C_{2d}^{d} difference indexes given a known Hamming distance threshold, d. Please see more detailed information in the following indexing step. The for loop from Line 3 to Line 7 checks each protein's query code in Q. In Line 4, it takes a query code $q = Q[i]$ from database Q. In Line 5, it calls the binary code searching algorithm (BCSA) to obtain the predicted results T. In Line 6, the relationship of query q and the predicted results T is added into the output set $PPIset$. After all members inside database Q are checked throughout, the final result is returned in Line 8.

Algorithm 1 The Prediction Algorithm With All proteins vs. All proteins

Input:
The distance threshold, d;
The ontic codes of a protein database, D;
The query codes of a protein database, Q;
Output:
$PPIset$ contains PPI relationship $\{q, T\}$;
1: $PPIset \leftarrow \Phi$, $M = |Q|$
2: Construct indexes for D;
3: **for** $i = 1$ **to** M **do**
4: $q = Q[i]$;
5: $T = BCSA(d, D, q, index)$
6: $PPIset \leftarrow PPIset \cup \{q, T\}$;
7: **end for**
8: return $PPIset$;

Going through Algorithm 1, one can find that there are two places needed to be further explained. One is in Line 2, the process to build indexes for an ontic codes of database D. The other is the binary code searching algorithm (BCSA) called in Line 5. Both of them is presented in the following subsection. Before that, we introduce the premise knowledge, a Pigeonhole principle first.

5.5.1 the Pigeonhole Principle

Pigeonhole principle: given two sequences with length of N, if the Hamming distance between them is less than or equal to d, after dividing these two sequences into $2d$ fragments evenly, inside these $2d$ fragments, there are at most having d number of fragments which are difference, that is, the remaining d fragments would have the exactly same sequences.

The Pigeohole principle is utilized in the prescreen step. According to the Pigeonhole principle, in $2d$ fragments, if one find that d fragments are the same, then we can claim that the distance between these compared sequences is less than or equal to d. To take d parts from $2d$, there are C_{2d}^{d} combinations. In this study, $d = 4$ is used. Then, the combinations of choosing d from $2d$ fragments, $C_{2d}^{d} = 70$.

In the prescreen step, the algorithm searches all these C_{2d}^{d} situations to identify possible protein candidates whose distance are less than or equal to d. For example, in this study, if the distance between two sequences is less than or equal to 4, then at least four out of the eight parts are having the exactly same bits, that is, at least half of 64 bits (32 bits) Hash codes are the same. These 32 bits can be regarded as numbers. In this way, we can turn sequence comparision problem to a much simpler problem of searching a number inside an array. This operation greatly improves the prescreen process and narrows down the candidate set.

5.5.2 The indexing process

The goal of indexing process is to build indexes which are binary numbers, and then turn sequence comparision problem into a much simpler problem of finding a number inside an array.

The indexing process are conducted as follows. First, the protein Binarized Hash codes are divided into $2d$ fragments. And then, d fragments are randomly selected from these $2d$ fragments. The exact same d fragments are chosen from all ontic codes in a database. Finally, these d fragments are connected together serving as an index code for a protein.

Remembering that, the ontic binary Hash codes and query binary Hash codes are binary numbers. An index by concatenating random d fragments from $2d$ fragments together is still a binary number. There are total of C_{2d}^{d} indexes.

For a database containing M proteins, it takes $O(Mlog_2M)$ time to build all indexes and sort an array with length M in order. Totally, the time requirement is $O(C_{2d}^{d}Mlog_2M)$. When $d = 4$ (the distance threshold value used in this work), $C_{2d}^{d} = 70$, this value can be regarded as a constant value. Thus, the time complexity of index building is $O(Mlog_2M)$.

5.5.3 BCSA: Binary Hash Code Searching Algorithm

The binary Hash code searching algorithm (BCSA) is a Nearest-Neighbor Filtering algorithm Based on Binarized Hash Code which is shown in Algorithm 2. There are four input parameters in the algorithm: (1) the distance threshold, d, where $d = 4$ is used in this work; (2) The ontic Hash codes of protein database, called D here; (3) the corresponding index of ontic Hash codes, called idx, which is constructed in the indexing process; (4) a query Hash codes of the queried protein, called q.

In Algorithm 2, in Line 1, the candidate set S is set to empty, the output PPI relationship set T is also set to empty. In Line 2, the query Hash codes q are divided

Algorithm 2 Binary Code Searching Algorithm With One vs. All

Input:
The distance threshold, d;
The ontic code of the protein database, D;
The query code of a query protein, q;
The index of the ontic code D, idx
Output:
The ID set of protein sequences, T;
1: $S \leftarrow \Phi$, $T \leftarrow \Phi$;
2: Divide binary query codes q into $2d$ equal parts respectively;
3: **for** $i = 1$ **to** C_{2d}^{d} **do**
4: Di = select the ith index from idx;
5: qi = select the ith d parts from q;
6: $tmpSet <-$ using qi search Di with binary searching algorithm;
7: $S \leftarrow S \cup tmpSet$;
8: **end for**
9: **for all** ID such that $ID \in S$ **do**
10: **if** dist(D[ID],q) $\leq$ d **then**
11: $T = T \cup ID$;
12: **end if**
13: **end for**
14: return T;

into $2d$ fragments evenly. From Line 3 to Line 8, there is a for loop which go through each possible combinations (C_{2d}^{d}) in the query codes q. Inside the for loop, in Line 4, the corresponding ith-index inside D is choosen, called Di. Please note that, Di is an array containing $M = |D|$ number of binary integers here. In another word, Di is an sorted array with M numbers inside. And the exact same d fragments from binary Hash query code are chosen from query protein, called qi, which is one binary number (Line 5). In line 6, it compares the number qi against sorted array Di by using binary search algorithm. That is, it trys to find all numbers inside array Di which are equal to qi. In this way, the string matching problem of searching a sequence against a database is turned into a much more simpler problem of finding an exact number in an array. Thus, all proteins whose indexes which have the exact value as qi are saved into candidate Set S (Line 7).

When all indexes are processed, the candidate Set S contains all proteins who at least have one of their Di equal to qi. Please note that, the number of proteins inside Set S is much less than the number of proteins in the original database D. From Line 9 to Line 13, the algorithm further uses Hamming distance to compare the query code q to each ontic code whose protein ID is in the candidate Set S. The protein who has the distance which is less than and equal to threshold d ($d = 4$ in this study), is selected as a potential PPI into set T.

5.5.4 The Analysis of Algorithms

We examine the time complexity of these two PPI prediction algorithms here.

Let's first check Algorithm 2. In Line 1, the candidate Set S and T is set to be empty, which is in $O(1)$ constant time. In Line 2, divide a query code q into $2d$ fragments evenly, the time complexity is $O(C_{2d}^{d})$. Taking d parts out of $2d$ fragments, there are $O(C_{2d}^{d})$ possible combinations. From Line 3 to Line 8, the for loop checks each possible combination one by one. In Line 4 and Line 5, it takes d parts out of $2d$ parts from ontic codes D and the query code q respectively. The time complexity for Line 4 and Line 5 are $O(1)$. In Line 6, it uses the number qi to searched against an sorted array Di by using binary search algorithm. The time complexity for Line

6 is $O(\log_2 M)$, where M is the number of protein sequences inside database D. And all sequence's ID in database D whose Di are equal to qi are saved into Candidate set S (Line 7). Let the number of search results be saved in K_i. The time complexity for Line 7 is $O(K_i)$. Together, the time complexity of the for loop from Line 3 to Line 8 is: $O(C_{2d}^d(\log_2 M + \sum K_i))$. From Line 9 to Line 13, all sequences inside candidate Set S is calculated by using Hamming distance, the distance which is less than or equal to the threshold d is kept inside the final output set T. The time complexity from Line 9 to Line 13 is $O(\sum K_i)$.

Together, the time complexity for Algorithm 2 is $O(C_{2d}^d(\log_2 M + \sum_{i=1}^{C_{2d}^d} K_i) + \sum_{i=1}^{C_{2d}^d} K_i) = O(C_{2d}^d(\log_2 M + \sum_{i=1}^{C_{2d}^d} K_i))$. For example, in this study, $d = 4$ and $C_{2d}^d = 70$, both of them can be regarded as constant, then the time complexity is $O(\log_2 M + \sum_{i=1}^{C_{2d}^d} K_i)$.

Let's consider the average situation for $\sum_{i=1}^{C_{2d}^d} K_i$. K_i is the number of exact matching of each search inside the for loop. Assume there are M number of proteins inside database D, then K_i is in the range of $[0, M]$. The ontic code and the query code are binary sequence of length 64. Then, in each index search, there are $\frac{d}{2d} * 64 = 32$ bits to be checked. In average situation, the possibility of exact match is $\frac{1}{2^{32}}$. In databse D with M number of proteins, the possibility of exact match is : $\frac{M}{2^{32}}$. The expected value of one loop (one search) is : $O(log_2 M + \sum_{i=1}^{C_{2d}^d} \frac{M}{2^{32}})$. In this study, M < 40000, let's assume M is equal to a larger value, say 40000 here. Then, we have $\sum_{i=1}^{C_{2d}^d} \frac{M}{2^{32}} = C_{2d}^d \frac{M}{2^{32}} = \frac{70*40000}{2^{32}} < \frac{2{,}800{,}000}{10^9} << 1$. Thus, this value can be regarded to be 1.

Then, the above formula $O(log_2 M + \sum_{i=1}^{C_{2d}^d} \frac{M}{2^{32}})$ of the time complexity for the algorithm can be rewritten to be: $O(\log_2 M + 1) = O(\log_2 M)$. This is the time complexity for Algorithm 2.

Now, we conside the time complexity of Algorithm 1. Line 1 runs in O(1). In Line 2, the time complexity of constructing indexes for database D is $O(C_{2d}^d M \log_2 M)$. From Line 3 to Line 7, the loop traverses all the query codes of the protein database Q which has M protein sequences. Both Line 4 and Line 6 run in O(1). Line 5 calls Algorithm 2 which runs in $O(\log_2 M)$. Then the time complexity of the for loop is $O(M * \log_2 M)$. Then, the total time complexity of Algorithm 1 is $O(C_{2d}^d M \log_2 M + M * \log_2 M)$. In this study, the distance threshold d in this case is 4 and $C_{2d}^d = 70$, both of them are constant. Thus, the time complexity is $O(Mlog_2 M)$. That is, we can claim that these two PPI prediction algorithms run in $O(Mlog_2 M)$ time complexity, where M is the number of proteins inside a database. This is a big improvement over the state-of-the-art $O(M^2)$.

Ethics approval and consent to participate
Not applicable.

Consent to publish
All authors consent this publication

Availability of data and materials
The Protein-Protein Interactions (PPI) dataset comes from the article, Large-scale prediction of human protein-protein interactions from amino acid sequence based on latent topic features [40]. The dataset is available in their website, URL:`http://www.csbio.sjtu.edu.cn/bioinf/LR_PPI/Data.htm`.

Competing interests
The authors declared that they have no competing interests.

Funding
This work is supported by the Chinese National Natural Science Foundation (Grant No. 61472082) , Natural Science Foundation of Fujian Province of China (Grant No. 2014J01220).

Author's contributions
YJ and JL contributed the idea and designed the study. LS implemented and performed most of the experiments. JL,SL, and YJ wrote the manuscript. All authors read and approved the final manuscript.

Acknowledgements

Author details
College of Computer and Cyber Security, Fujian Normal University, 350108 Fuzhou, PRChina.

References
1. Wu HM, Yun N. Identification of Protein-protein Interaction Based on Feature Weighted. Computer Technology and Development. 2016;26(2):114-7.
2. Ito T, Chiba T, Ozawa R, Yoshida M, Hattori M, Sakaki Y. A comprehensive two-hybrid analysis to explore the yeast protein interactome. Proceedings of the National Academy of Sciences of the United States of America. 2001;98(8):4569-74.
3. Ho Y, Gruhler A, Heilbut A, Bader GD, Moore L, Adams SL, et al. Systematic identification of protein complexes in Saccharomyces cerevisiae by mass spectrometry. Nature. 2002;415(6868):180-3.
4. Foltman M, Sanchez-Diaz A. Studying Protein-Protein Interactions in Budding Yeast Using Co-immunoprecipitation. Methods Mol Biol. 2016;1369:239-56.
5. Huang H, Alvarez S, Nusinow DA. Data on the identification of protein interactors with the Evening Complex and PCH1 in Arabidopsis using tandem affinity purification and mass spectrometry (TAP–MS). Data in Brief. 2016;8:56-60.
6. Mrowka R. Is there a bias in proteome research? Genome Research. 2001;11(12):1971.
7. Koike A, Kobayashi Y, Takagi T. Kinase Pathway Database: An Integrated Protein-Kinase and NLP-Based Protein-Interaction Resource. Genome Research. 2003;13(6A):1231-43.
8. Bunescu RR. Integrating co-occurrence statistics with information extraction for robust retrieval of protein interactions from Medline. In: LNLBioNLP '06: Proceedings of the HLT-NAACL BioNLP Workshop on Linking Natural Language and Biology; 2006. p. 49-56.
9. Grimes GR, Wen TQ, Mewissen M, Baxter RM, Moodie S, Beattie JS, et al. PDQ Wizard: automated prioritization and characterization of gene and protein lists using biomedical literature. Bioinformatics. 2006;22(16):2055-7.
10. Fundel K, Küffner R, Zimmer R. RelEx–relation extraction using dependency parse trees. Bioinformatics. 2007;23(3):365-71.
11. Temkin JM, Gilder MR. Extraction of protein interaction information from unstructured text using a context-free grammar. Bioinformatics. 2003;19(16):2046-53.
12. Ananiadou S, Kell DB, Tsujii JI. Text mining and its potential applications in systems biology. Trends in Biotechnology. 2006;24(12):571-9.
13. Haussler D. Convolution Kernels on Discrete Structures. tech rep; 1999.
14. Lodhi H, Saunders C, Shawe-Taylor J, Cristianini N, Watkins C. Text Classification using String Kernels. Journal of Machine Learning Research. 2002;2(3):419-44.
15. Sarkar D, Saha S. Machine-learning techniques for the prediction of protein–protein interactions. Journal of Biosciences. 2019;44(4):1-12.
16. Maheshwari S, Brylinski M. Across-proteome modeling of dimer structures for the bottom-up assembly of protein-protein interaction networks. BMC Bioinformatics. 2017;18(1):1-14.
17. Sikandar A, Anwar W, Bajwa UI, Wang X, Sikandar M, Yao L, et al. Decision Tree Based Approaches for Detecting Protein Complex in Protein Protein Interaction Network (PPI) via Link and Sequence Analysis. IEEE Access. 2018;6:22108 22120.
18. Debasree S, Tanmoy J, Sudipto S, Manuela HC. LMDIPred: A web-server for prediction of linear peptide sequences binding to SH3, WW and PDZ domains. PLoS ONE. 2018;13(7):e0200430.
19. Romero-Molina S, Ruiz-Blanco YB, Harms M, Münch J, Sanchez-Garcia E. PPI-Detect: A support vector machine model for sequence-based prediction of protein–protein interactions. Journal of Computational Chemistry. 2019;40(11):1233-42.
20. Zhang SW, Hao LY, Zhang TH. Prediction of Protein–Protein Interaction with Pairwise Kernel Support Vector Machine. International Journal of Molecular Sciences. 2014;15(2):3220-33.
21. Ruan P, Hayashida M, Akutsu T, Vert JP. Improving prediction of heterodimeric protein complexes using combination with pairwise kernel. Bmc Bioinformatics. 2018;19(S1):39.
22. Gui Y, Wang R, Wei Y, Wang X. DNN-PPI: A LARGE-SCALE PREDICTION of PROTEIN-PROTEIN INTERACTIONS BASED on DEEP NEURAL NETWORKS. Journal of Biological Systems. 2019;27(1):1-18.
23. Wang YB, You ZH, Xiao L, Jiang TH, Chen X, Zhoua X, et al. Predicting protein–protein interactions from protein sequences by a stacked sparse autoencoder deep neural network. Molecular BioSystems. 2017;13(7):1336-44.
24. Long Z, Yu G, Xia D, Wang J. Protein-Protein Interactions Prediction based on Ensemble Deep Neural Networks. Neurocomputing. 2018;324(9):10-9.
25. Browne F, Wang H, Zheng H, Azuaje F. Supervised Statistical and Machine Learning Approaches to Inferring Pairwise and Module-Based Protein Interaction Networks. In: IEEE International Conference on Bioinformatics & Bioengineering; 2007. p. 1365–1369.
26. Lin X, Chen Xw. Heterogeneous data integration by tree-augmented naïve Bayes for protein-protein interactions prediction. Proteomics. 2012;13(2):261-8.

27. Ngamsuriyaroj S, Thepsutum K. Identifying Dominant Amino Acid Pairs of Known Protein-Protein Interactions via K-Means Clustering. In: 2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS); 2017. p. 286-91.
28. Liu P, Lei Y, Shi D, Tang X. Prediction of Protein-Protein Interactions Related to Protein Complexes Based on Protein Interaction Networks. Biomed Res Int. 2015;2015:1-9.
29. Zhao Z, Gong X. Protein-protein interaction interface residue pair prediction based on deep learning architecture. IEEE/ACM Transactions on Computational Biology & Bioinformatics. 2017;16(5):1753-9.
30. Lecun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553):436-44.
31. Hang L, Gong XJ, Yu H, Zhou C. Deep Neural Network Based Predictions of Protein Interactions Using Primary Sequences. Molecules. 2018;23(8):1923.
32. Somaye H, Behnam N, Khan AA, Xu J. Predicting protein–protein interactions through sequence-based deep learning. Bioinformatics. 2018;17(34):802-10.
33. Sun T, Bo Z, Lai L, Pei J. Sequence-based prediction of protein protein interaction using a deep-learning algorithm. BMC Bioinformatics. 2017;18(1):277.
34. Du X, Sun S, Hu C, Yao Y, Yan Y, Zhang Y. DeepPPI : Boosting Prediction of Protein-Protein Interactions with Deep Neural Networks. Journal of Chemical Information & Modeling. 2017;57(6):1499-510.
35. Patel S, Tripathi R, Kumari V, Varadwaj P. DeepInteract: Deep Neural Network based Protein-Protein Interaction prediction tool. Current Bioinformatics. 2017;12(6):551-7.
36. Huang L, Liao L, Wu CH. Completing sparse and disconnected protein-protein network by deep learning. Bmc Bioinformatics. 2018;19(1):103.
37. Yang F, Fan K, Song D, Lin H. Graph-based prediction of Protein-protein interactions with attributed signed graph embedding. BMC Bioinformatics. 2020;21(1):1-16.
38. Huang L, Liao L, Wu CH. Inference of protein-protein interaction networks from multiple heterogeneous data. Eurasip Journal on Bioinformatics & Systems Biology. 2016;2016(1):8.
39. Suraj P, Daniel NJ, Kristiansen TZ, Ramars A, Vineeth S, Babylakshmi M, et al. Human protein reference database as a discovery resource for proteomics. Nucleic Acids Research. 2004;32(Database Issue):D497-501.
40. Pan X, Zhang Y, Shen H. Large-scale prediction of human protein-protein interactions from amino acid sequence based on latent topic features. Journal of Proteome Research. 2010;9(10):4992-5001.
41. Chollet F. Deep Learning with Python. Manning Publications; 2017.
42. Elabd H, Bromberg Y, Hoarfrost A, Lenz T, Wendorff M. Amino acid encoding for deep learning applications. BMC Bioinformatics. 2020;21(1):1-14.
43. Zhang R, Lin L, Zhang R, Zuo W, Zhang L. Bit-Scalable Deep Hashing with Regularized Similarity Learning for Image Retrieval and Person Re-identification. IEEE Transactions on Image Processing. 2015;24(12):4766-79.
44. Chen Z, Cai R, Lu J, Feng J, Jie Z. Order-Sensitive Deep Hashing for Multimorbidity Medical Image Retrieval:. In: 21st International Conference, Granada, Spain, September 16–20, 2018, Proceedings, Part I; 2018. p. 620-8.
45. Wang Hl, Yu J, Xiao Cb. Deep non-relaxation hashing based on point pair similarity. ACTA AUTOMATICA SINICA. 2021;47(5):1077-86.