Bioinformatics, 34, 2018, i802–i810
doi: 10.1093/bioinformatics/bty573
ECCB 2018

OXFORD

Predicting protein–protein interactions through sequence-based deep learning

Somaye Hashemifar, Behnam Neyshabur, Aly A. Khan and Jinbo Xu*

Toyota Technological Institute at Chicago, Chicago, IL 60637, USA

*To whom correspondence should be addressed.

Abstract

Motivation: High-throughput experimental techniques have produced a large amount of protein–protein interaction (PPI) data, but their coverage is still low and the PPI data is also very noisy. Computational prediction of PPIs can be used to discover new PPIs and identify errors in the experimental PPI data.

Results: We present a novel deep learning framework, DPPI, to model and predict PPIs from sequence information alone. Our model efficiently applies a deep, Siamese-like convolutional neural network combined with random projection and data augmentation to predict PPIs, leveraging existing high-quality experimental PPI data and evolutionary information of a protein pair under prediction. Our experimental results show that DPPI outperforms the state-of-the-art methods on several benchmarks in terms of area under precision-recall curve (auPR), and computationally is more efficient. We also show that DPPI is able to predict homodimeric interactions where other methods fail to work accurately, and the effectiveness of DPPI in specific applications such as predicting cytokine-receptor binding affinities.

Availability and implementation: Predicting protein-protein interactions through sequence-based deep learning): https://github.com/hashemifar/DPPI/.

Contact: j3xu@ttic.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

The complete map of all protein–protein interactions (PPIs) remains an urgent but incomplete task due to numerous roadblocks, including experimental cost and noise, and the massive combinatorial space of potential protein interactions. Despite these challenges, tens of thousands of direct interactions have been elaborated using a combination of high- and low-throughput experimental techniques (Das and Yu, 2012; Li *et al.*, 2016). These experimentally determined protein interactions collectively form a substantial dataset that is suitable for training predictive computational models of protein interactions. Thus, new computational methods that can accurately predict interactions while efficiently leveraging the continuously increasing numbers of new experimentally resolved protein–protein interactions are desperately needed.

Proteins directly interact via their three-dimensional structures, but they typically interact in a highly restricted manner given the vast number of possible orientations (Jones and Thornton, 1996). One reason is that the amino acid composition of proteins encodes the information required to not only reproducibly fold them into their specific shapes, but to also reliably negotiate their biophysical interactions. The systematic analysis of PPI interfaces has shown complementarity in shape and electrostatic composition of amino acids (Jones and Thornton, 1996; Sheinerman *et al.*, 2000). More recently, evolutionary coupling has demonstrated that coordinated changes in amino acids across evolution in pairs of proteins often occurs at interaction interfaces (Hopf *et al.*, 2014; Ovchinnikov *et al.*, 2014). Taken together, the composition and residue order of a protein's sequence informs both its shape and subsequent capacity to interact with other proteins.

The primary amino acid sequence remains the most complete type of data available for all proteins. Thus, there is a longstanding interest in using sequence-based methods to model and predict protein interactions (Ben-Hur and Noble, 2005; Gomez *et al.*, 2003; Hamp and Rost, 2015; Kuang *et al.*, 2005; Shen *et al.*, 2007). Several sequence-based methods have been developed to predict PPIs, such as PIPE2 (Pitre *et al.*, 2012), SigProd (Martin *et al.*, 2005), AutoCorrelation (Guo *et al.*, 2008), Zhou's work (Zhou *et al.*, 2011), You's work (You *et al.*, 2014), Profppikernel (Hamp and Rost, 2015), Wong's work (Wong *et al.*, 2015), Yang's work (Yang *et al.*, 2010) and Sun's work (Sun *et al.*, 2017).

Given a pair of proteins A and B, PIPE2 predicts a PPI if subsequences of A and B occur frequently among positive training database (Pitre

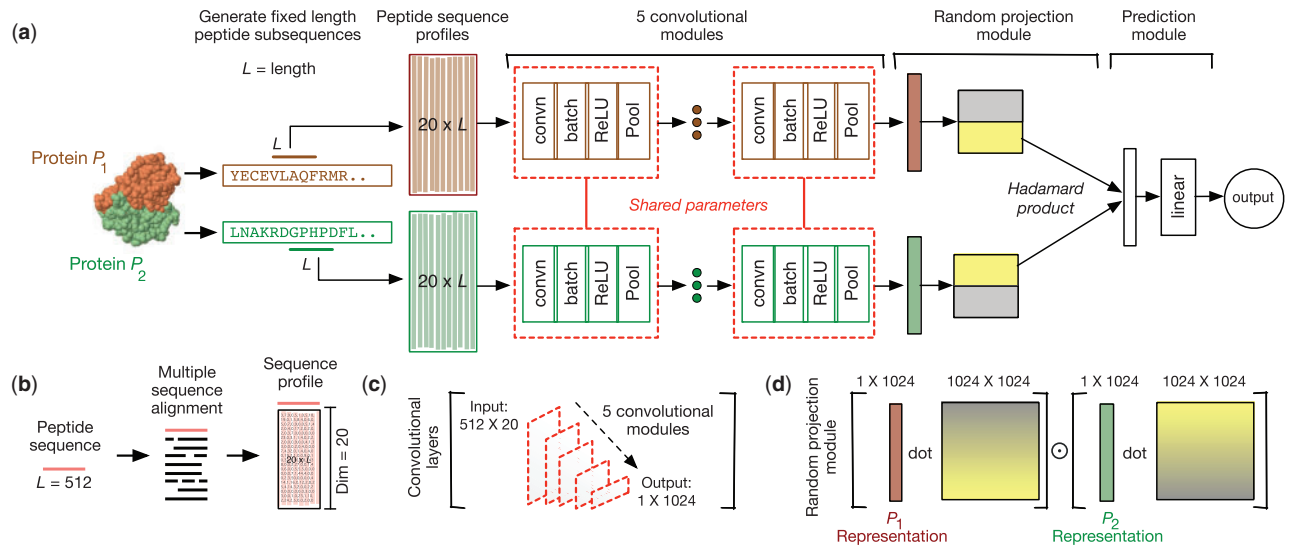

Fig. 1. Illustration of the DPPI model for predicting binary protein–protein interactions. (**a**) DPPI takes as input a pair of protein sequences and learns to predict an interaction score. (**b**) Each input protein is represented as a probabilistic sequence profile generated by PASI-BLAST. (**c**) The convolutional module consists of several convolutional layers that learn a set of filters, each of which is responsible for detecting various patterns in a sequence. (**d**) The random projection module maps each sequence to a representation useful for modelling paired sequences

et al., 2012). Using an inexact matching, PIPE2 enumerates how frequently each 20-mer segment in proteins A and B co-occur in the positive training database. A pair of 20-mer is considered a hit if their PAM120 score (Dayhoff, 1972) is above a certain threshold. If the average count for several adjacent subsequences is above a threshold, the algorithm then predicts that protein A and B would interact.

SigProd (Martin *et al.*, 2005) predicts PPIs using pairwise kernel and representing a sequence by 3-mers. Each 3-mer is converted to a signature where the middle amino acid is the root and the other two are its neighbors alphabetically ordered. A protein is then represented by a signature vector, in which each element indicates the frequency of one signature. The feature vector for a protein pair is then defined by the tensor product of their corresponding protein signature vectors. Finally, the signature product of two protein pairs is computed by their standard Euclidean inner product within a SVM classifier as a kernel function.

AutoCorrelation (Guo *et al.*, 2008) employs seven physicochemical properties of amino acids (e.g. polarity) to represent a protein as a real-valued vector of dimension $7 \times l$, where l is the protein sequence length. This vector is then used to derive a vector of correlation coefficients, indicating whether two residues along the sequence have correlated properties. A protein pair is then characterized by concatenating the correlation coefficient vectors of its constitute proteins, which is then used as input to a machine learning algorithm such as SVM.

Profppikernel (Hamp and Rost, 2015) represents the state-of-the-art sequence-based method for PPI prediction. Profppikernel counts the co-occurrence frequency of two different triads in a protein pair, and uses this frequency to represent each protein pair as a feature vector of $20^3 \times 20^3$ elements (In total there 20^3 possible 3-mers taken from 20 amino acids). Each element is the number of times its corresponding k-mer is conserved in the evolutionary profile of the protein, i.e. how often the sum of amino acid substitution scores is below a user-defined threshold. Profppikernel employs an SVM to predict PPIs from the feature vectors.

Although all these machine learning models have been effective at capturing information in amino acid sequences to model and predict PPIs (Ben-Hur and Noble, 2005; Hamp and Rost, 2015; Kuang *et al.*, 2005), they have significant memory and runtime limitations when training with large datasets. On the other hand, the use of evolutionary coupling for predicting protein interactions often requires a long evolutionary history(Hopf *et al.*, 2014; Ovchinnikov *et al.*, 2014), limiting predictions for engineered proteins and more recently evolved proteins such as those from the adaptive immune system.

In this paper, we present a novel deep learning framework, DPPI, to model and predict direct physical protein–protein interactions from primary amino acid sequences (Fig. 1a). Our deep learning framework can (i) efficiently handle massive amounts of training data, (ii) flexibly train models for different applications without significant parameter tuning and (iii) robustly capture complex and non-linear relationships involved in protein interactions. For training, DPPI takes as input a pair of protein sequences and an experimentally determined binding score. Binding scores can be a binary label (0 = non-binding, and 1 = binding) or experimentally-determined real values (e.g. enrichment value or dissociation constant, K_d). We have tested DPPI on different datasets, showing that DPPI greatly outperforms several popular methods in terms of both accuracy and running time.

2 Materials and methods

We introduce three major ideas of our novel deep learning framework. First, DPPI leverages a large corpus of unsupervised data to build a robust profile representation of each protein sequence (Fig. 1b) (Hamp and Rost, 2015; Kuang *et al.*, 2005). DPPI uses a sliding window to look at patches of linear protein sequences and then builds probabilistic profiles using the known proteome to characterize the protein sequence. Thus, instead of the raw sequence, a high dimensional position-specific profile representation of the sequences is given to the model. This enables non-identical but homologous proteins, such as from human and mouse, to have similar sequence profile representations. Furthermore, our use of a sliding window also enables DPPI to effectively operate on sequences of variable lengths by training on all combinations of patches. From a deep-learning perspective, while most patches between two proteins

may not be involved in direct interactions, these 'random' combinations are a form of data augmentation that improves model robustness by injecting and training with random noise (Sietsma and Dow, 1991; Vincent *et al.*, 2008).

Second, DPPI employs a Siamese-like convolutional neural network architecture. Siamese neural networks are useful for tasks that involve learning a complex relationship between two entities (Bromley *et al.*, 1993). Our architecture contains two identical sub-networks, which share the same configuration, parameters and weights. Within each sub-network, the convolutional module (Fig. 1c) convolves a protein profile with specific parameters (filters), the rectification layer introduces non-linearity, and the pooling stage computes the average of each filter's rectified response across the profile. Each sub-network produces a high-dimensional representation of a single protein sequence. Importantly, parameter sharing guarantees that two similar sequences are transformed by their respective sub-networks to similar feature representations.

Third, we introduce a novel random projection module. The values of the last convolutional module are randomly projected into a subspace using a pair of (pseudo-orthogonal) random weight vectors (Fig. 1d). The pair of random untrained weight vectors are generated only once before training the whole deep learning model and gives the model its unique ability to learn both homodimeric and heterodimeric protein interactions. The random weights are reversed between the two sub-networks, which induces symmetry and enables the model to ignore the order of the input profiles to either sub-network. Lastly, the final vectors are combined as a high-dimensional feature representation of the protein pair and used to calculate an interaction score and predict interaction probability. For predicting and testing the interaction between two proteins the maximum interaction score (or probability) over all patches is reported.

2.1 DPPI model design

DPPI has three main modules: a convolutional module (Sect. 2.1.2), a random projection module (Sect. 2.1.3) and a prediction module (Fig. 1a; Sect. 2.1.4). The convolutional module is the core module. It learns a set of filters and is responsible for detecting patterns in a protein sequence. Specifically, the convolutional module maps a pair of input protein sequences (Sect. 2.1.1) to a representation that is useful for predicting protein–protein interactions.

The random projection module projects the representations learned by the convolutional module for a pair of input proteins to two different spaces (Sect. 2.1.3).

The prediction module takes the representations generated by the random projection module and outputs a score that is then used to predict existence of an interaction between two input proteins (Sect. 2.1.4).

2.1.1 Input

DPPI takes a pair of protein profiles S and S', and outputs a binary value $f(S, S')$ indicating whether the corresponding proteins interact. For a given primary protein sequence with length n we generate a $n \times 20$ array S, called sequence profile, as follows:

$$S = \begin{bmatrix} s_{1,1} & \cdots & s_{1,20} \\ \vdots & & \vdots \\ s_{i,1} & \vdots & s_{i,20} \\ \vdots & & \vdots \\ s_{n,1} & \cdots & s_{n,20} \end{bmatrix}$$

where $s_{i,j}$ is the probability of j_{th} amino acid in i_{th} position of the sequence. DPPI constructs the protein profile by running PSI-BLAST with 3 iterations and E-value = 0.001 to search through the Uniprot protein sequence database dated in 2016. These parameters are observed to work well for generating profiles (Hamp and Rost, 2015).

2.1.2 Convolutional module

Each convolutional module consists of four layers including convolution layer, rectified linear unit, batch normalization and pooling. In our DPPI model, output of a convolutional module is computed by an expression starting with a convolution layer and ending in a pooling layer:

$$R = \text{Pool}\Big(\text{ReLU}\Big(\text{Batch}_{\Gamma,\beta}\Big(\text{conv}_M(S)\Big)\Big)\Big),$$

where R is the output vector, S is the input profile and Γ, β, M are the parameters of batch normalization and convolution layers. Convolution layer detects the patterns in an input sequence and outputs the corresponding features. Batch normalization layer takes the output of a convolutional layer and normalizes it with respect to the mean and variance of feature values. The rectified linear unit (ReLU) takes the output of a batch normalization layer and clamps all the negative values to zero to introduce non-linearity. Finally, a pooling layer is employed to reduce the dimension of output of ReLU layer to a one-dimensional array (See Supplementary Section S1 for more details).

2.1.3 Random projection module

DPPI exploits a Siamese-like convolutional neural network architecture, and contains two identical convolutional modules with the same configuration and parameters for a pair of input profiles S and S'. Let R_S and $R_{S'}$ be the representations learned by convolutional modules for two input proteins S and S', respectively.

The random projection (RP) module projects the representations learned by the convolutional module for the two input proteins to two different spaces. The RP layer consists of two separate fully-connected sub-networks. Each takes as input a $1 \times d$ output array R and computes a d-dimensional vector $\text{o} = \text{Net}_{\text{W}}(R)$where:

$$\text{o}_S = \text{ReLU}\Big(\text{Batch}([W^1 \;\; W^2]R_S)\Big)$$

$$\text{o}_{S'} = \text{ReLU}\Big(\text{Batch}([W^2 \;\; W^1]R_{S'})\Big)$$

where W^1 and $W^2$are two $d \times (d/2)$ arrays denoting the weights of first and second sub-networks, respectively. $[W^1 \;\; W^2]$ represents the concatenation of matrices W^1 and W^2 and $[W^2 \;\; W^1]$ is defined in a similar way. The vector o is a representation of profile S. Having two sets of weights W^1 and W^2 and flipping them helps the model investigate the combination of the patterns learned from two input proteins. Without random projection, the Hadamard product module mainly calculates the similarity of (the convoluted representations of) two input proteins but not their interaction potential and thus, may reduce the predictive power of DPPI. Random projection also enables DPPI to ignore the order of the input profiles (see Sect. 2.1.4.)

The weights W^1 and W^2 are initialized using a standard Gaussian distribution and are fixed during the training (i.e. we do not train them). Having untrained weights will reduce the number of parameters and thus, speed up model training. It also prevents the model from over-fitting and thus, results in a better classification. The output of the random projection layer is the final representation of each input protein.

2.1.4 Prediction module

Prediction module takes a pair of protein representations o_S and $o_{S'}$, respectively, for two input proteins S and S', and computes a probability value indicating whether two proteins interact. This module first performs an element-wise multiplication (i.e. Hadamard product) on two vectors o_S and $o_{S'}$ by:

$$q = o_S \odot o_{S'}$$

where $\odot$ denotes Hadamard product. Next, a linear layer transforms d-dimensioanl vector q into an output score $f(S,S') = \text{Linear}_w(q)$. A linear layer is a $(1+d)$-dimensional vector of tunable weights w, where the output score is calculated as the linear combination

$$f(S,S') = w_0 + \sum_{k=1}^{d} w_k q_k$$

where weight w_k is the weight of q_k's contribution to the output, and w_0 is an additive bias term. Finally, we predict the interaction probability between S and S' by $1/\left(1+e^{-f(S,S')}\right)$ and calculate the loss function as the negative log-likelihood of the correct interaction:

$$\ell\left(f(S,S'), y_{S,S'}\right) = \ln\left(1+e^{-y_{S,S'}f(S,S')}\right)$$

where $y_{S,S'} = 1$ if there is an interaction between S and S' and $y_{S,S'} = -1$ otherwise.

Remark. DPPI switches the random weights in the RP (random projection) module so that it is insensitive to the order of input profiles. That is, the linear layer in the prediction module has the same input vector regardless of the order of input proteins.

Furthermore, switching the weights enables the model to learn interacting motifs (patterns) of proteins with different patterns (e.g. heterodimeric proteins). Let $W^1_{d,k}$ and $W^2_{d,k}$ denote the $(d \times 1)$ weight vector for filter k, $1 \le k \le d/2$, respectively, in first and second sub-networks of the RP module. By doing an element-wise multiplication in prediction module the value for filter k is computed as:

$$q_k = (R_S \times W^2_{d,k}) \odot (R_{S'} \times W^1_{d,k})$$

Because $W^2_{d,k} \neq W^1_{d,k}$, even if one of R_S and $R_{S'}$ has a low value, it still possible that q_k may have a large value, which is beneficial when the two proteins have different composition of amino acids. That is, by random projection DPPI can detect different combinations of protein motifs (patterns) rather than just similar motifs.

2.2 Training pipeline

We trained DPPI described in Figure 1 using error backpropagation with stochastic gradient descent algorithm with momentum.

2.2.1 Variable sequence lengths

Our training proteins have different sequence lengths; we crop each protein sequence into overlapping subsequences of length 512 with an overlapping margin of 256. Specifically, the first subsequence starts at position 1, the second one starts at position 256 and so on until the whole sequence is covered. Therefore, two adjacent subsequences overlap on 256 residues. That is, for a sequence profile with length l_s, cropping generates C different subsequences where

$$C = \begin{cases} 1 & l_s \le 512 \\ l_s/256 - 1 & otherwise \end{cases}$$

In databases used in our experiments, 34% of sequences are longer than 512 residues. These 'random' combinations of subsequences used in training are a form of data augmentation, which has been used to improve the generalization of the convolutional layers and prevent overfitting in image processing (Zeiler and Fergus, 2014). During the training, each subsequence is treated as a separate protein and for a pair of interacting proteins S and S', we train the model to be able to predict their interaction by given any pair of their crops. For prediction, however, we are interested in a single value $f(S,S')$. Therefore, we compute $f(S,S')$ to be the maximum predicted score over all crop combinations of S and S'. Accordingly, we also convert the maximum predicted score to an interaction probability value by formula $1/\left(1+e^{-f(S,S')}\right)$.

2.2.2 Training objective

Suppose we are given N training pairs $((S,S')^1, y^1), \ldots, ((S,S')^N, y^N)$ where $(S,S')^i$ is a pair of input protein profiles and $y^i = y((S,S')^i)$ is the binary training label where:

$$\begin{cases} y^i = 1 & S \text{ and } S' \text{ interact} \\ y^i = -1 & otherwise \end{cases}$$

Let f^i indicate the corresponding DPPI prediction $f^i = f\left((S,S')^i\right)$ with respect to θ. We train our model by minimizing the following regularized objective function:

$$\sum_{i=1}^{N} \ell(f^i, y^i) + \lambda \|\theta\|_2$$

where θ is the vector of all parameters, λ is the regularization parameter controlling the model complexity, and $\|.\|_2$ is the $L2$-norm. The $L2$-norm regularization prevents over-fitting by restricting the parameter space.

2.2.3 Initial weight scale

We randomly initialize the parameters of convolution and linear layers by drawing weights from a zero-mean Gaussian distribution with standard deviation 1, which is a popular way in deep convolutional neural network (Bengio, 2012). The parameter values of the convolution layers are then divided by the square root of $m \times d$. The parameters Γ and β of batch normalization are set respectively to $\vec{1}$ and $\vec{0}$, according to Cooijmans *et al.* (2016) and Ioffe and Szegedy (2015). The parameters of the random projection layer are un-trained and initialized using a standard Gaussian distribution.

2.2.4 Implementation details

Our implementation uses the software package Torch7/Lua (Bromley *et al.*, 1993). A Titan X Pascal was used for training the model. We trained our model using binary physical protein–protein interactions (Das and Yu, 2012; Li *et al.*, 2016). Protein sequences were retrieved from the Uniprot database (http://www.uniprot.org/).

DPPI has five convolutional modules, respectively with input size 20, 64, 128, 256 and 512. Every convolution layer has two hyper-parameters dimension d and filter width m. The size of pooling window is equal to 4, i.e. $l_p = 4$ and the learning rate is set to 0.01, i.e. $\eta = 0.01$.

The stochastic gradient descent method uses a subset of training examples to compute gradient. The batch size determines how many

training pairs are considered in calculating the gradient. Here we use batches of size 100.

DPPI was trained by a stochastic gradient descent (SGD) method with a momentum set to 0.9 to minimize the objective function (Sect. 2.2.2). It is observed that momentum-based SGD can effectively train deep learning models (Sutskever *et al.*, 2013). Let θ denote the vector of all parameters. DPPI updates θ with respect to the objective:

$$\Delta\theta^{t+1} = \alpha\Delta\theta^{t} - \eta\nabla_{\theta} f(s, s')$$
$$\theta^{t+1} = \theta^{t} + \Delta\theta^{t+1}$$

where θ^{t+1} is the updated parameter vector, η is learning rate and $\nabla_{\theta} f(s, s')$ is gradient.

3 Results

We compare DPPI with many state-of-the-art sequence-based PPI prediction methods including PIPE2 (Pitre *et al.*, 2012), SigProd (Martin *et al.*, 2005), AutoCorrelation (Guo *et al.*, 2008), Profppikernel (Hamp and Rost, 2015), You's work (You *et al.*, 2013), You's work (You *et al.*, 2014), Wong's work (Wong *et al.*, 2015), Zhou's work (Zhou *et al.*, 2011), Yang's work (Yang *et al.*, 2010) and (Huang *et al.*, 2015) on a variety of datasets. We also test the learning capacity and computational time of DPPI. Further, we evaluate how well DPPI predicts binding between specific cytokine variants and their receptor. We could not compare our method with Sun's work (Sun *et al.*, 2017) because neither their method nor their data was available. We also for the sake of fairness, did not compare DPPI with Du's work (Du *et al.*, 2017) because their method uses variety of complex features beside protein sequences.

3.1 PPI prediction quality using cross-validation

Human and yeast datasets. To evaluate DPPI's ability to predict direct physical protein interactions in an unbiased manner, we adopted the high-quality human and yeast PPI benchmark data used by Profppikernel (Hamp and Rost, 2015). The human PPI data is obtained by collecting the 10% top-scoring interactions from Hippie database v1.2 (Schaefer *et al.*, 2012). The high-quality yeast PPI data is extracted from the DIP core set, which is a subset of the full DIP database (Salwinski *et al.*, 2004) and contains only the most reliable physical interactions. Negative interactions were generated by randomly sampling from all proteins. Following the strategy used in (Hamp and Rost, 2015), the number of negative samples is 10 times that of positive samples.

Following the same strategy as Profppikernel, we removed redundancy of the human and yeast data such that no two PPIs are similar at sequence level. Two PPIs are considered similar if at least two sequences, each of one PPI, have a sequence identity greater than 40%. Next, we split each non-redundant set into 10 parts, using nine as training data and one for test. Overall, there are on average 842 and 746 positive PPIs in each test data for human and yeast respectively. We performed 10-fold cross validation and report their average results.

Figure 2a shows that DPPI yields results with a much better mean auPR (area under precision-recall curve), demonstrating that DPPI outperforms other sequence-based methods, including the state-of-the-art Proffppikernel.

Prediction of homodimeric interactions. Our method DPPI is able to predict both homodimeric and heterodimeric PPIs. We took all the cross validation partitions of the yeast data (explained in Section 3.1), where the test set in each partition contains on average 12 positive homodimeric interactions and none of their proteins are similar with the proteins in the yeast training data. We do not include the human data here since it has very few (or zero) positive homodimeric PPIs.

Profppikernel could not predict any of homodimeric interactions in the yeast set, indicating that it is not able to detect the proper interacting motifs (patterns) among homodimers. However, DPPI on average could yield interaction probability value more than 0.5 for half of the homodimeric interactions. That is, DPPI can predict half of the homodimeric interactions, including Anaphase-promoting complex subunit 1 (APC1) and Protein SRL2. The outperformance of DPPI on homodimeric PPIs comes from its deep architecture, especially the Random Projection module that can learn interacting motifs effectively.

S.cerevisiae core subset. This PPI dataset, described by You *et al.* (2014), is derived from the *S.cerevisiae* core subset in the DIP database. It has 11188 positive and negative interactions. Following You *et al.* (2014), we performed 5-fold cross validation and report their average results. Table 1 shows that DPPI is better than the other 5 methods (which are trained and tested on the same dataset) in terms of precision, recall and accuracy. Accuracy is the percentage of correct predictions out of all predictions. DPPI used 0.5 as the probability cutoff to assign labels.

3.2 DPPI performance on large data

Here we study if DPPI improves with additional training examples.

Large dataset. We curated a large dataset consisting of direct physical PPIs from 11 different species H.sapiens, *S.cerevisiae*, *S.pombe*, *M.musculus*, *D.melanogaster. C.elegans*, *A.thaliana*, *B.subtilis*, *B.taurus*, *E.coli* and *R.norvegicus* obtained from HINT (Das and Yu, 2012). Overall, there are 289180 interactions. We randomly sampled 1k, 2.5k, 5k, 10k, 25k and 50k positive interactions for six different training sizes, and then assembled test data using 25% of the training size from the remaining interactions. Again, following Profppikernel that uses 40% sequence identity as cutoff to remove redundancy, we used the same procedure to reduce sequence homology bias. Similar to Hamp and Ros (Hamp and Rost, 2015), for each positive interaction, 10 negatives were generated by randomly sampling from all proteins.

Evaluation on datasets with different size. Figure 2b presents the mean auPR (area under precision-recall curve) of DPPI and Profppikernel on the test data with respect to different training sizes. We discontinued training Profppikernel with 10k examples after 17 days since we did not know when it could finish training. The trend line is plotted by the first three training sizes and then extended to the last three sizes with the 95% interval illustrated. In summary, we found that DPPI continues to learn and improve auPR with the addition of more examples, but this is not the case for the kernel-based method Profppikernel.

To assess the scalability of DPPI, we compared the wall clock running time of DPPI to Profppikernel. Figure 2c shows that in contrast to Profppikernel, DPPI scales efficiently with respect to the increasing number of training examples. By contrast, the running time of Profppikernel increases significantly along with the training size.

3.3 Performance against more stringent criteria

Here we evaluate the performance of DPPI on new test datasets generated by a more stringent criterion. We took all the cross-validation partitions of the human data (explained in Section 3.1) as the original train/test sets. Then, we made new test sets for human by selecting

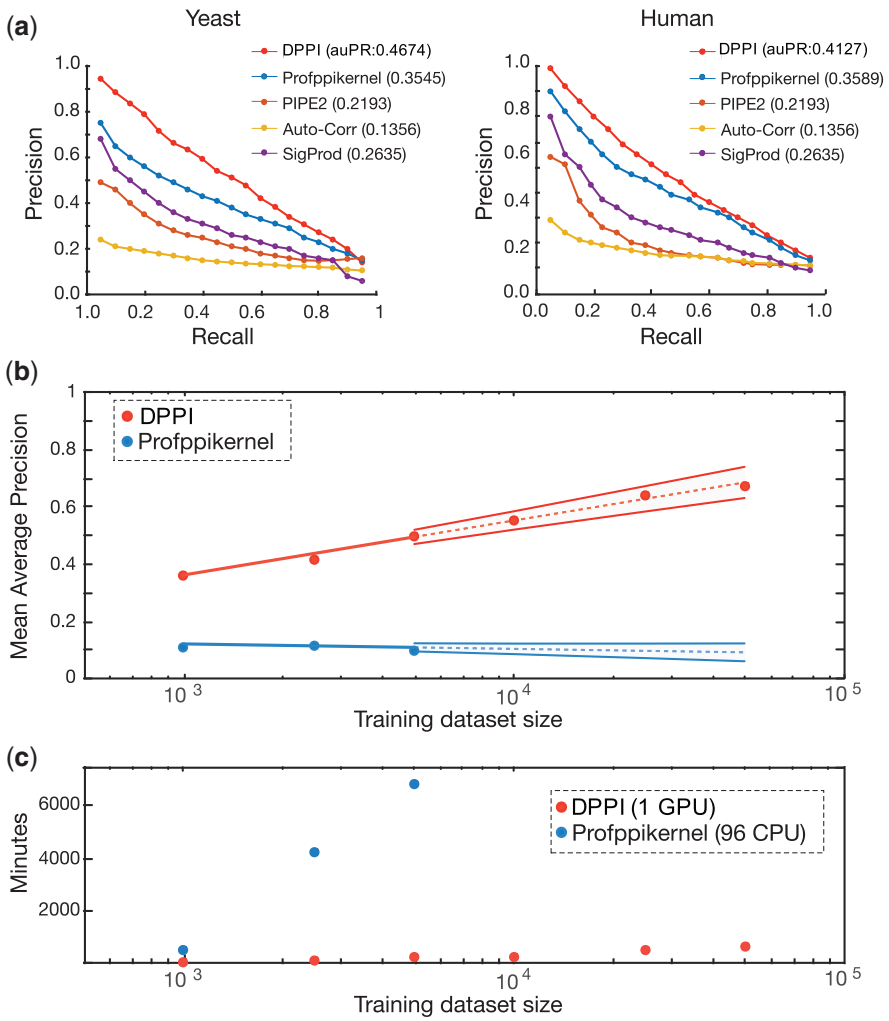

Fig. 2. DPPI accurately and efficiently predicts protein–protein interactions. (**a**) Performance comparison of DPPI with competing sequence-based methods on human and yeast benchmark data. Values were adopted from Hamp and Rost (2015) and show the average precision and recall rate of 10-fold cross-validation. The numbers in parenthesis indicate the area under precision-recall curve (auPR). (**b**) The mean auPR of DPPI and Profppikernel with respect to different training sizes. We discontinued training Profppikernel with 10k examples after 17 days before it finished. The trend line using the first 3 training sizes is plotted for each method and extended over the last 3 sizes with the 95% interval illustrated. (**c**) The wall time for training DPPI and Profppikernel. We trained Profppikernel using 96 threads by running it on 32 AMD Opteron(TM) Processor 6272 CPUs

Table 1. Performance comparison of DPPI with other state-of-the art methods on the *S.cerevisiae* core subset

Method	Precision	Recall	Acc.
You's work (You *et al.*, 2013)	87.59	86.15	87.00
You's work (You *et al.*, 2014)	91.94	90.67	91.36
Wong's work (Wong *et al.*, 2015)	96.45	91.10	93.92
Zhou's work (Zhou *et al.*, 2011)	89.50	87.37	88.56
Yang's work (Yang *et al.*, 2010)	90.24	81.03	86.15
DPPI	**96.68**	**92.24**	**94.55**

Note: Bold font is used to indicate the best performance.

a subset of interactions in the original test sets, where each protein sequence in new test set shares less than 25% sequence identity with the proteins in the training dataset of the same partition. We also created new test sets for yeast in a similar way. The new test sets of human and yeast include on average 128 and 58 negative interactions, respectively. Unfortunately, these new test sets do not have any positive interactions. To qualify the performance of DPPI on the new test sets, we plotted the interaction probability distribution of negative interactions of both new and original test sets and compared their distribution with each other. Figure 3 represents the plots where curves are calculated from 10-fold cross validation. It shows that the distribution of the interaction probability values in the new test sets is similar as that in the original human and yeast test sets, indicating that the behavior of DPPI on the negative interaction of the new test sets is consistent with that of original test sets. This result shows that DPPI is reliable on test data generated by an even more stringent criterion. Figure 3 also shows that most of the negative interactions are assigned a low probability value (i.e. less than 0.2) by DPPI. Moreover, it presents that the distribution of the interaction probability of positive interactions in the original test sets are very different than that of negative interactions in both new and original test sets. Most of the positive interactions are assigned a probability larger than 0.5.

3.4 PPI detection for independent species

Presuming that large numbers of physically interacting proteins in one organism have evolved in a correlated fashion, their respective

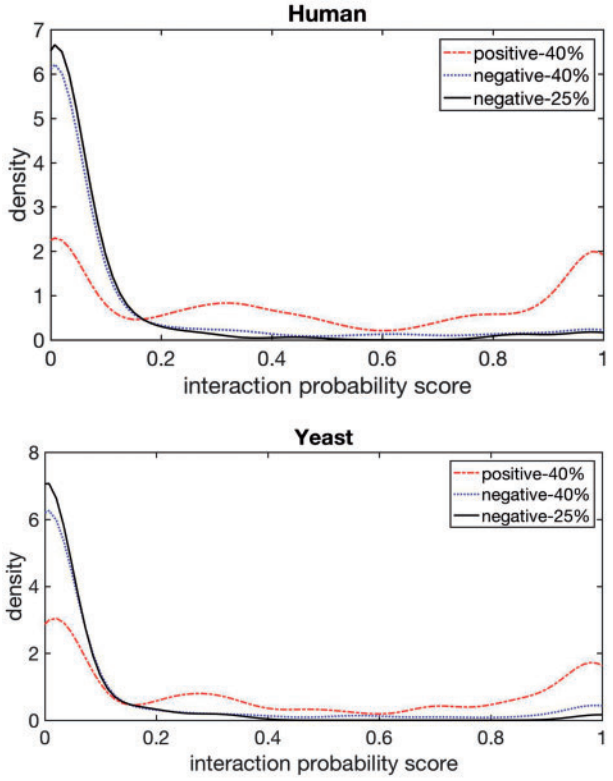

Fig. 3. DPPI performance on test sets generated by more stringent criteria. Probability values of negative interactions on both the new test sets and original test sets is consistent with each other. Black and blue lines show the density of negative interactions for new (25% sequence identity) and original (40% sequence identity) test sets respectively. Red line shows the density of positive interactions for original test set. The top and bottom pictures are for human and yeast, respectively

orthologs in other organisms also interact (Walhout, 2000). In this section, we train a model by using the *S.cerevisiae* core subset described in Section 3.1 and test that model on four other species datasets as test sets. *S.cerevisiae* dataset with a total of 11188 protein interactions, described by You *et al.* (50), is collected from DIP. The four test sets include *Caenorhabditis elegans* (*C.elegans*) with 4013 interacting pairs, *Escherichia coli* (*E.coli*) with 6954 interacting pairs, *Homo sapiens* (*H.sapience*) with 1412 interacting pairs and *Mus musculus* (*M.musculus*) with 313 interacting pairs used by Zhou *et al.* (2011). Table 2 shows that DPPI is able to predict PPIs from other species and DPPI outperforms Huang's work (Huang *et al.*, 2015) and Zhou's work (Zhou *et al.*, 2011). Both our method and the others are trained and tested on the same data by using the optimal parameters suggested by them.

3.5 Importance of random projection module

To quantify the importance of the RP module in our DPPI model, we compare the performance of our model (i.e. model with fixed-weight RP) with two alternative architectures - one without RP module (no-RP model) and the other with a variant RP module (variant-RP model). In the variant-RP model, weights of the RP module are not fixed, but are trained. We apply all three structures on the same human and yeast datasets explained in Section 3.1. Figure 4 shows that the model with fixed-weight RP achieves the highest precision on both human and yeast and its auPR values are almost 1.5 and 1.3 times higher than the other two models. This result confirms the advantage of fixed-weight RP module over the others. Such an advantage comes from the identification of interacting patterns of two proteins.

Table 2. Prediction accuracy on the other species, using *S.cerevisiae* core subset as the training data

Species	DPPI	Huang's work	Zhou's work
H.sapiens	**96.24**	82.22	76.27
M.musculus	**95.84**	79.87	76.68
C.elegans	**95.51**	81.19	75.73
E.coli	**96.66**	66.08	71.24

Note: DPPI used 0.5 as the probability cutoff to assign labels. Bold font is used to indicate the best performance.

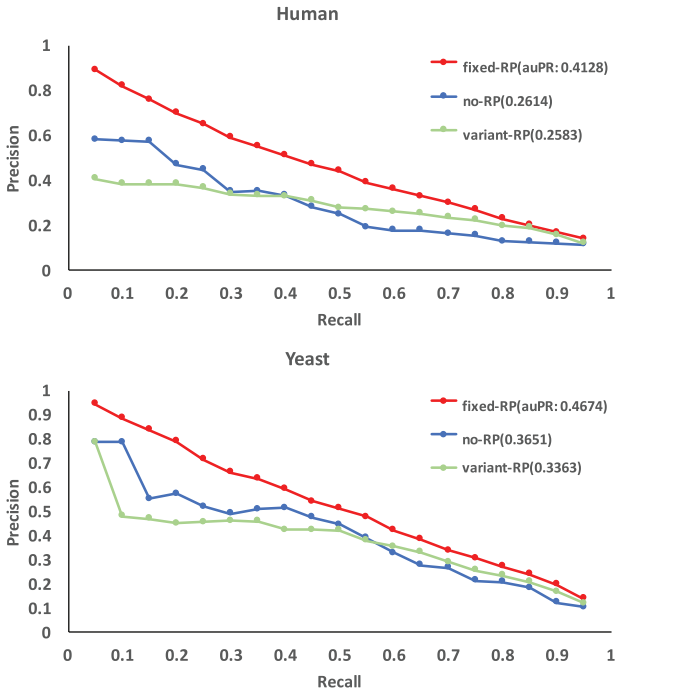

Fig. 4. Performance comparison of DPPI model (i.e. model with fixed-weight RP) with two alternative architectures—one without RP module (no-RP model) and the other with a variant RP module

3.6 DPPI software for predicting interactions

To deliver a software useful for the broad community, we have trained a model on a randomly selected subset of 100 000 PPIs from the large dataset described in Section 3.2. This pre-trained model can now be used for predicting PPIs for any species. Even though we have already verified the efficiency and superiority of DPPI, as a sanity check, we have further tested the trained model on a test data containing 3000 randomly-selected PPI interactions and each protein sequence in the test set shares less than 25% sequence identity with the proteins in the training data. Figure 5 shows the precision-recall curve with auPR = 0.79 achieved by DPPI.

3.7 Evaluating binding affinities

Here we sought to characterize the relationship between our predicted interaction probability values and experimentally measured binding affinities. Binding affinity is the strength of the binding interaction between a molecule and its ligand. Binding affinity is typically measured by equilibrium dissociation constant (Kd), which

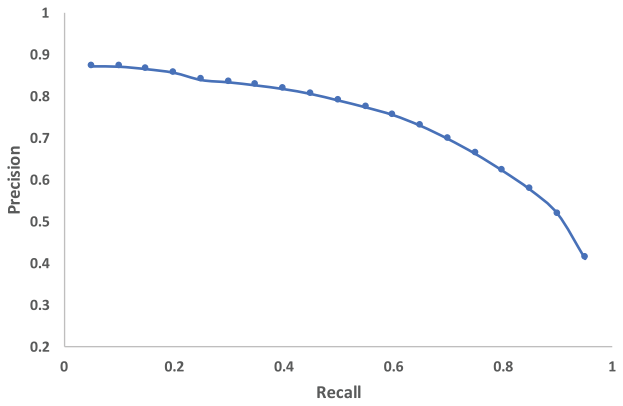

Fig. 5. Performance of DPPI software regarding 25% sequence similarity

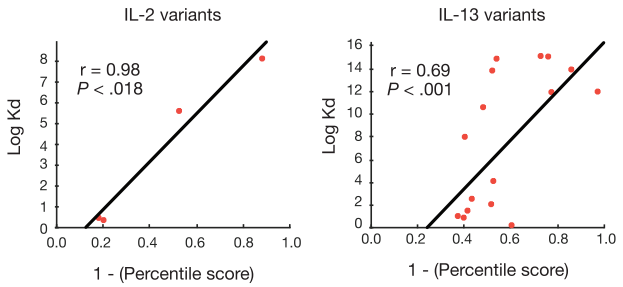

Fig. 6. Scatter plot between the ranking percentile calculated from DPPI's prediction scores versus log dissociation constant (K_d) of IL-2 and IL-13 engineered variants. Trend line emphasizing linear correlation is depicted. IL-2: Pearson correlation = 0.98, *P*-value < 0.018. IL-13: Pearson correlation = 0.69, *P*-value < 0.001

is used to rank strength of biomolecular interactions. The smaller the Kd value, the greater the binding affinity of the ligand for its target. Cytokines, such as interlukin-2 (IL-2) (Levin *et al.*, 2012) and interlukin-13 (IL-13) (Moraga *et al.*, 2015), have been engineered to bind their cognate receptors at different binding affinities. These cytokines are involved in important immunological functions (Moraga *et al.*, 2015). To train our model, we obtained a curated large dataset consisting of ∼166000 direct physical PPIs of H.sapiens from (Li *et al.*, 2016). Similar to Hamp and Rost (Hamp and Rost, 2015), for each positive interaction, 10 negatives were generated by randomly sampling from all proteins (10x random negative). Therefore, our training data consists of 166000 positive and 1660000 negative interactions. We obtained data for testing engineered cytokines IL-2 and IL-13, respectively, from Levin *et al.* (2012) and Moraga *et al.* (2015). They use structure-based engineering to generate four IL-2 and sixteen IL-13 variants that cover a spectrum of binding strength for the receptor subunit IL-2Rβ and IL-13Rα1. In vitro evolution of human IL-2 variants with high affinity for IL-2Rβ are shown in Supplementary Table S1, and sixteen IL-13 variants for IL-13Rα1 are presented in Supplementary Table S2. Protein sequences for IL-2 and IL-13 were retrieved from Uniprot database with IDs Q7Z7M3 and P35225, respectively. We applied DPPI to engineered variants of IL-2 and IL-13, and their cognate receptors. To compare results between cytokines, we simulated 1000 sequences by randomly permuting the amino acid positions engineered to confer the strongest binding affinity for each cytokine. We then ranked the interaction probability values (predicted by our method) decreasingly and calculated the percentile rank of probability values of each test variant. A sequence with a percentile 90% has a higher interaction probability value than 90% of its variants. The percentile is subtracted from 1 to enable easier comparison to Kd. That is, lower probability values and Kd indicate greater binding affinity.

Figure 6 shows these two values for all four variants of IL-2 (left) and 16 variants of IL-13 (right). There is a significant correlation between the percentile and the experimentally-determined binding affinities for both IL-2 (Pearson correlation ∼ 0.98, *P*-value ∼ 0.02) and IL-13 (Pearson correlation ∼ 0.7, *P*-value ∼ 0.001). Since there are only 4 variants of IL-2 tested, we may not be able to draw a solid conclusion from the IL-2 result. Nevertheless, the result of IL-13 indicates that DPPI's predicted scores are monotonically related to Kd values, which suggests that DPPI may help to identify specific amino acid substitutions resulting in increased or decreased binding affinities.

4 Discussion

In this paper, we have presented a new deep learning method that can greatly improve sequence-based PPI prediction. Our model is scalable with respect to the data size and applicable to different biological problems, such as predicting cytokine-receptor binding affinities, without significant parameter tuning. Our experimental results indicate that DPPI outperforms the state-of-the-art methods on several benchmarks by 15–20% in terms of auPR (area under precision-recall curve) while its running time is almost 15 times faster. Our experimental results also show that DPPI can effectively detect the homodimeric interactions while Profppikernel cannot.

Our method exploits a Siamese-like convolutional neural network architecture for PPI prediction that allows us to encapsulate information about composition of amino acids, their sequential orders and co-occurrence of interacting sequence motifs appearing in a protein pair. Our method allows us to infer interactions between homodimeric as well as heterodimeric proteins.

DPPI also uses a novel random projection module that investigates the combination of protein motifs. Applying DPPI to modeling protein-ligand binding, we found that our sequence-based predictions can identify specific amino acid substitutions resulting in increased or decreased binding affinities. This indicates that even without structure information DPPI can effectively model binding affinities.

In summary, our deep learning framework serves as a principled computational approach to model and predict PPIs from sequence and is generalizable to many applications.

Funding

This work was supported by the NIH R01GM089753 and NSF/CCF AF-1618648 and NVIDIA Inc. for its GPU card.

Conflict of Interest: none declared.

References

Ben-Hur,A. and Noble,W.S. (2005) Kernel methods for predicting protein–protein interactions. *Bioinformatics*, **21**, i38–i46.

Bengio,Y. (2012) *Neural Networks: Tricks of the Trade*. Springer-Verlag, Berlin, Heidelberg, pp. 437–478.

Bromley,J. *et al.* (1993) Signature Verification Using A "Siamese" Time Delay Neural network. *IJPRAI*, **07**, 669–688.

Cooijmans,T. *et al.* (2016) Recurrent batch normalization. *arXiv Preprint arXiv*: 1603.09025.

Das,J. and Yu,H. (2012) HINT: high-quality protein interactomes and their applications in understanding human disease. *BMC Syst. Biol.*, **6**, 92.

Dayhoff,M. (1972) A model of evolutionary change in proteins. *Atlas Protein Sequence Struct.*, **5**, 89–99.

Du,X. *et al.* (2017) DeepPPI: boosting prediction of protein–protein interactions with deep neural networks. *J. Chem. Inf. Model.,* **57**, 1499–1510.

Gomez,S.M. *et al.* (2003) Learning to predict protein–protein interactions from protein sequences. *Bioinformatics,* **19**, 1875–1881.

Guo,Y. *et al.* (2008) Using support vector machine combined with auto covariance to predict protein–protein interactions from protein sequences. *Nucleic Acids Res.,* **36**, 3025–3030.

Hamp,T. and Rost,B. (2015) Evolutionary profiles improve protein–protein interaction prediction from sequence. *Bioinformatics*, **31**, 1945–1950.

Hopf,T.A. *et al.* (2014) Sequence co-evolution gives 3D contacts and structures of protein complexes. *Elife*, **3**, e03430.

Huang,Y.-A. *et al.* (2015) Using weighted sparse representation model combined with discrete cosine transformation to predict protein–protein interactions from protein sequence. *BioMed Res. Int.,* **2015**, 1.

Ioffe,S. and Szegedy,C. (2015) Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In, *International Conference on Machine Learning*. pp. 448–456.

Jones,S. and Thornton,J.M. (1996) Principles of protein–protein interactions. *Proc. Natl. Acad. Sci. USA,* **93**, 13–20.

Kuang,R. *et al.* (2005) Profile-based string kernels for remote homology detection and motif extraction. *J. Bioinf. Comput. Biol.,* **03**, 527–550.

Levin,A.M. *et al.* (2012) Exploiting a natural conformational switch to engineer an interleukin-2/superkine/'. *Nature,* **484**, 529–533.

Li,T. *et al.* (2016) A scored human protein–protein interaction network to catalyze genomic interpretation. *Nat. Methods*.

Martin,S. *et al.* (2005) Predicting protein–protein interactions using signature products. *Bioinformatics,* **21**, 218–226.

Moraga,I. *et al.* (2015) Instructive roles for cytokine-receptor binding parameters in determining signaling and functional potency. *Sci. Signal.,* **8**, ra114–ra114.

Ovchinnikov,S. *et al.* (2014) Robust and accurate prediction of residue–residue interactions across protein interfaces using evolutionary information. *Elife,* **3**, e02030.

Pitre,S. *et al.* (2012) Short co-occurring polypeptide regions can predict global protein interaction maps. *Sci. Rep.*, **2**, 239.

Salwinski,L. *et al.* (2004) The database of interacting proteins: 2004 update. *Nucleic Acids Res.,* **32**, D449–D451.

Schaefer,M.H. *et al.* (2012) HIPPIE: integrating protein interaction networks with experiment based quality scores. *PloS One,* **7**, e31826.

Sheinerman,F.B. *et al.* (2000) Electrostatic aspects of protein–protein interactions. *Curr. Opin. Struct. Biol.,* **10**, 153–159.

Shen,J. *et al.* (2007) Predicting protein–protein interactions based only on sequences information. *Proc. Natl. Acad. Sci. USA,* **104**, 4337–4341.

Sietsma,J. and Dow,R.J. (1991) Creating artificial neural networks that generalize. *Neural Netw.*, **4**, 67–79.

Sun,T. *et al.* (2017) Sequence-based prediction of protein protein interaction using a deep-learning algorithm. *BMC Bioinformatics,* **18**, 277.

Sutskever,I. *et al.* (2013) On the importance of initialization and momentum in deep learning. In: *International Conference on Machine Learning*. pp. 1139–1147.

Vincent,P. *et al.* (2008) Extracting and composing robust features with denoising autoencoders. In: *Proceedings of the 25th International Conference on Machine Learning*. ACM. pp. 1096–1103.

Walhout,A.J. (2000) Protein interaction mapping in *C. elegans* using proteins involved in vulval development. *Science*, **287**, 116–122.

Wong,L. *et al.* (2015) Detection of protein–protein interactions from amino acid sequences using a rotation forest model with a novel pr-lpq descriptor. In: *International Conference on Intelligent Computing*. Springer. pp. 713–720.

Yang,L. *et al.* (2010) Prediction of protein–protein interactions from protein sequence using local descriptors. *Protein Peptide Lett.*, **17**, 1085–1090.

You,Z.-H. *et al.* (2013) Prediction of protein–protein interactions from amino acid sequences with ensemble extreme learning machines and principal component analysis. *BMC Bioinformatics*, **14**, S10.

You,Z.-H. *et al.* (2014) Prediction of protein–protein interactions from amino acid sequences using a novel multi-scale continuous and discontinuous feature set. *BMC Bioinformatics*, **15**, S9.

Zeiler,M.D. and Fergus,R. (2014) Visualizing and understanding convolutional networks. In: *European Conference on Computer Vision*. Springer. pp. 818–833.

Zhou,Y.Z. *et al.* (2011) Prediction of protein–protein interactions using local description of amino acid sequence. In: *Advances in Computer Science and Education Applications*. Springer. pp. 254–262.