Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

Sequence Analysis

Multifaceted Protein-Protein Interaction Prediction Based on Siamese Residual RCNN

Muhao Chen [1]*†, Chelsea J.-T. Ju [1]*, Guangyu Zhou[1], Xuelu Chen[1], Tianran Zhang[2], Kai-Wei Chang[1], Carlo Zaniolo[1], and Wei Wang[1]

[1]Department of Computer Science, University of California, Los Angeles, 90095, USA
[2]Department of Bioengineering, University of California, Los Angeles, 90095, USA

† To whom correspondence should be addressed.
∗ Both authors contributed equally to this work.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: Sequence-based protein-protein interaction (PPI) prediction represents a fundamental computational biology problem. To address this problem, extensive research efforts have been made to extract predefined features from the sequences. Based on these features, statistical algorithms are learned to classify the PPIs. However, such explicit features are usually costly to extract, and typically have limited coverage on the PPI information.
Results: We present an end-to-end framework, PIPR, for PPI predictions using only the protein sequences. PIPR incorporates a deep residual recurrent convolutional neural network in the Siamese architecture, which leverages both robust local features and contextualized information, which are significant for capturing the mutual influence of proteins sequences. PIPR relieves the data pre-processing efforts that are required by other systems, and generalizes well to different application scenarios. Experimental evaluations show that PIPR outperforms various state-of-the-art systems on the binary PPI prediction problem. Moreover, it shows a promising performance on more challenging problems of interaction type prediction and binding affinity estimation, where existing approaches fall short.
Availability: The implementation is available at `https://github.com/muhaochen/seq_ppi.git`
Contact: muhaochen@ucla.edu

1 Introduction

Detecting protein-protein interactions (PPIs) and characterizing the interaction types are essential toward understanding cellular biological processes in normal and disease states. Knowledge from these studies potentially facilitates therapeutic target identification (Petta *et al.*, 2016) and novel drug design (Skrabanek *et al.*, 2008). High-throughput experimental technologies have been rapidly developed to discover and validate PPIs on a large scale. These technologies include yeast two-hybrid screens (Fields and Song, 1989), tandem affinity purification (Gavin *et al.*, 2002), and mass spectrometric protein complex identification (Ho *et al.*, 2002). However, experiment-based methods remain expensive, labor-intensive, and time-consuming. Most importantly, they often suffer from high levels of false-positive predictions (You *et al.*, 2015; Sun *et al.*, 2017). Evidently, there is an immense need for reliable computational approaches to identify and characterize PPIs.

The amino acid sequence represents the primary structure of a protein, which is the simplest type of information either obtained through direct sequencing or translated from DNA sequences. Many research efforts address the PPI problem based on predefined features extracted from protein sequences, such as ontological features of amino acids (Jansen *et al.*, 2003), autocovariance (AC) (Guo *et al.*, 2008), conjoint triads (CT) (Shen *et al.*, 2007) and composition-transition-distribution (CTD) descriptors (Yang *et al.*, 2010). These features generally summarize specific aspects of protein sequences such as physicochemical properties, frequencies of local patterns, and the positional distribution of amino acids. On top of these features, several statistical learning algorithms (Guo *et al.*, 2008; Huang *et al.*, 2015; You *et al.*, 2014, 2015) are applied to predict PPIs in the form of binary classification. These approaches provide feasible solutions to the problem. However, the extracted features used in these approaches only have limited coverage on interaction information, as they are dedicated to specific facets of the protein profiles.

To mitigate the inadequacy of statistical learning methods, deep learning algorithms provide the powerful functionality to process large-scale data and automatically extract useful features for objective tasks (LeCun *et al.*, 2015). Recently, deep learning architectures have produced powerful systems to address several bioinformatics problems related to single nucleotide sequences, such as genetic variants detection (Anderson, 2018), DNA function classification (Quang and Xie, 2016), RNA-binding site prediction (Zhang *et al.*, 2015) and chromatin accessibility prediction (Min *et al.*, 2017). These works typically use convolutional neural networks (CNN) (Anderson, 2018; Zhang *et al.*, 2015) for automatically selecting local features, or recurrent neural networks (RNN) (Quang and Xie, 2016) that aim at preserving the contextualized and long-term ordering information. By contrast, fewer efforts (discussed in Related Work) have been made to capture the pairwise interactions of proteins with deep learning, which remains a non-trivial problem with the following challenges: (i) Characterization of the proteins requires a model to effectively filter and aggregate their local features, while preserving significant contextualized and sequential information of the amino acids; (ii) Extending a deep neural architecture often leads to inefficient learning processes, and suffers from the notorious vanishing gradient problem (Pascanu *et al.*, 2013); (iii) An effective mechanism is also needed to apprehend the mutual influence of protein pairs in PPI prediction. Moreover, it is essential for the framework to be scalable to large data, and to be generalized to different prediction tasks.

In this paper, we introduce `PIPR` (Protein-Protein Interaction Prediction Based on Siamese Residual RCNN), a deep learning framework for PPI prediction using only the sequences of a protein pair. `PIPR` employs a Siamese architecture to capture the mutual influence of a protein sequence pair. The learning architecture is based on a residual recurrent convolutional neural network (RCNN), which integrates multiple occurrences of convolution layers and residual gated recurrent units. To represent each amino acid in this architecture, `PIPR` applies an efficient property-aware lexicon embedding approach to better capture the contextual and physicochemical relatedness of amino acids. This comprehensive encoding architecture provides a multi-granular feature aggregation process to effectively leverage both sequential and robust local information of the protein sequences. It is important to note that the scope of this work focuses only on the primary sequence as it is the fundamental information to describe a protein.

Our contributions are three-fold. First, we construct an end-to-end framework for PPI prediction that relieves the data pre-processing efforts for users. `PIPR` requires only the primary protein sequences as the input, and is trained to automatically preserve the critical features from the sequences. Second, we emphasize and demonstrate the needs of considering the contextualized and sequential information when modeling the PPIs. Third, the architecture of `PIPR` can be flexibly used to address different PPI tasks. Besides the binary prediction that is widely attempted in previous works, our framework extends its use to two additional challenging problems: multi-class interaction type prediction and binding affinity estimation. We use five datasets to evaluate the performance of our framework on these tasks. `PIPR` outperforms various state-of-the-art approaches on the binary prediction task, which confirms the effectiveness in terms of integrating both local features and sequential information. The promising performance of the other two tasks demonstrates the wide usability of our approach. Especially on the binding affinity estimation of mutated proteins, `PIPR` is able to respond to the subtle changes of point mutations and provides the best estimation with the smallest errors.

2 Related Work

Sequence-based approaches provide a critical solution to the binary PPI prediction task. Homology-based methods (Philipp *et al.*, 2016) rely on BLAST to map a pair of sequences to known interacting proteins. Alternatively, other works address the task with statistical learning models, including SVM (Guo *et al.*, 2008; You *et al.*, 2014), kNN (Yang *et al.*, 2010), Random Forest (Wong *et al.*, 2015), multi-layer perceptron (MLP) (Du *et al.*, 2017), and ensemble ELM (EELM) (You *et al.*, 2013). These approaches rely on several feature extraction processes for the protein sequences, such as CT (You *et al.*, 2013; Sun *et al.*, 2017), AC (Guo *et al.*, 2008; You *et al.*, 2013; Sun *et al.*, 2017), CTD (Yang *et al.*, 2010; Du *et al.*, 2017), multi-scale continuous and discontinuous (MCD) descriptors (You *et al.*, 2013), and local phase quantization (LPQ) (Wong *et al.*, 2015). These features measure physicochemical properties of the 20 canonical amino acids, and aim at summarizing full sequence information relevant to PPIs. More recent works (Sun *et al.*, 2017; Wang *et al.*, 2017) propose the use of stacked autoencoders (SAE) to refine these heterogeneous features in low-dimensional spaces, which improve the aforementioned models on the binary prediction task. On the contrary, fewer efforts have been made towards multi-class prediction to infer the interaction types (Zhu *et al.*, 2006; Silberberg *et al.*, 2014) and the regression task to estimate binding affinity (Srinivasulu *et al.*, 2015; Yugandhar and Gromiha, 2014). These methods have largely relied on their capability of extracting and selecting better features, while the extracted features are far from fully exploiting the interaction information.

By nature, the PPI prediction task is comparable to the neural sentence pair modeling tasks in natural language processing (NLP) research, as they both seek to characterize the mutual influence of two sequences based on their latent features. In NLP, neural sentence pair models typically focus on capturing the discourse relations of lexicon sequences, such as textual entailment (Yin *et al.*, 2016; Hu *et al.*, 2014), paraphrases (Yin and Schütze, 2015; He *et al.*, 2015) and sub-topic relations (Chen *et al.*, 2018). Many recent efforts adopt a Siamese encoding architecture, where encoders based on convolutional neural networks (CNN) (Hu *et al.*, 2014; Yin and Schütze, 2015) and recurrent neural networks (RNN) (Mueller and Thyagarajan, 2016) are widely used. A binary classifier is then stacked to the sequence pair encoder for the detection of a discourse relation.

In contrast to sentences, proteins are profiled in sequences with more intractable patterns, as well as in a drastically larger range of lengths. Precisely capturing the PPI requires much more comprehensive learning architectures to distill the latent information from the entire sequences, and to preserve the long-term ordering information. One recent work (Hashemifar *et al.*, 2018), DPPI, uses a deep CNN-based architecture which focuses on capturing local features from protein profiles. DPPI represents the first work to deploy deep-learning to PPI prediction, and has achieved the state-of-the-art performance on the binary prediction task. However, it requires excessive efforts for data pre-processing such as constructing protein profiles by PSI-BLAST (Altschul *et al.*, 1997), and does not incorporate a neural learning architecture that captures the important contextualized and sequential features. DNN-PPI (Li *et al.*, 2018) represents another relevant work of this line, which deploys a different learning structure with two separated CNN encoders. However, DNN-PPI does not incorporate physicochemical properties into amino acid representations, and does not employ a Siamese learning architecture to fully characterize pairwise relations of sequences.

3 Methods

We introduce an end-to-end deep learning framework, `PIPR`, for sequence-based PPI prediction tasks. The overall learning architecture is illustrated in Fig 1. `PIPR` employs a Siamese architecture of residual RCNN encoder to better apprehend and utilize the mutual influence of two sequences.

To capture the features of the protein sequences from scratch, PIPR pre-trains the embeddings of canonical amino acids to capture their contextual similarity and physicochemical properties. The latent representation of each protein in a protein pair is obtained by feeding the corresponding amino acid embeddings into the sequence encoder. The embeddings of these two sequences are then combined to form a sequence pair vector. Finally, this sequence pair vector is fed into a multi-layer perceptron with appropriate loss functions, suiting for specific prediction tasks. In this section, we describe the details of each model component. We begin with the denotations and problem specifications.

3.1 Preliminary

We use A to denote the vocabulary of 20 canonical amino acids. A protein is profiled as a sequence of amino acids $S = [a_1, a_2, ..., a_l]$ such that each $a_i \in A$. For each amino acid a_i, we use bold-faced $\mathbf{a}_i$ to denote its embedding representation, which we are going to specify in Section 3.2.3. We use I to denote the set of protein pairs, and $p = (S_1, S_2) \in I$ denotes a pair of proteins of which our framework captures the interaction.

We address three challenging PPI prediction tasks based only on the primary sequence information: (i) *Binary prediction* seeks to provide a binary classifier to indicate whether the corresponding protein pair interacts, which is the simplest and widely considered problem setting in previous works (Sun *et al.*, 2017; Hashemifar *et al.*, 2018; Skrabanek *et al.*, 2008). (ii) *Interaction type prediction* is a multi-class classification problem, which seeks to identify the interaction type of two proteins. (iii) *Binding affinity estimation* aims at producing a regression model to estimate the strength of the binding interaction.

3.2 RCNN-based Protein Sequence Encoder

We employ a deep Siamese architecture of Residual RCNN to capture latent semantic features of the protein sequence pairs.

3.2.1 Residual RCNN

The RCNN seeks to leverage both the global sequential information and local features that are significant to the characterization of PPI from the protein sequences. This deep neural encoder stacks multiple instances of two computational modules, i.e. *convolution layers with pooling* and *bidirectional residual gated recurrent units*. The architecture of an RCNN unit is shown on the left of Fig. 2.

The convolution layer with pooling. We use $X = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_l]$ to denote an input vector sequence that corresponds to the embedded amino acids or the outputs of a previous neural layer. A convolution layer applies a weight-sharing kernel $\mathbf{M}_c \in \mathbb{R}^{h \times k}$ to generate a k-dimension latent vector $\mathbf{h}_t^{(1)}$ from a window $\mathbf{v}_{t:t+h-1}$ of the input vector sequence X:

$$\mathbf{h}_t^{(1)} = \mathrm{Conv}(\mathbf{v}_{t:t+h-1}) = \mathbf{M}_c \mathbf{v}_{t:t+h-1} + \mathbf{b}_c$$

for which h is the kernel size, and $\mathbf{b}_c$ is a bias vector. The convolution layer applies the kernel as a sliding window to produce a sequence of latent vectors $\mathbf{H}^{(1)} = [\mathbf{h}_1^{(1)}, \mathbf{h}_2^{(1)}, ..., \mathbf{h}_{l-h+1}^{(1)}]$, where each latent vector combines the local features from each h-gram of the input sequence. The n-max-pooling mechanism is applied to every consecutive n-length subsequence (i.e., non-overlapped n-strides) of the convolution outputs, which takes the maximum value along each dimension j by $\mathbf{h}_{i,j}^{(2)} = \max(\mathbf{h}_{i:n+i-1,j}^{(1)})$. The purpose of this mechanism is to discretize the convolution results, and preserve the most significant features within each n-stride (Chen *et al.*, 2018; Kim, 2014; Hashemifar *et al.*, 2018). By definition, this mechanism divides the size of processed features by n. The outputs from the max-pooling are fed into the bidirectional gated recurrent units in our RCNN encoder.

The Residual Gated Recurrent Units. The Gated Recurrent Unit model (GRU) represents an alternative to the Long-short-term Memory network (LSTM) (Cho *et al.*, 2014), which consecutively characterizes the sequential information without using separated memory cells (Dhingra *et al.*, 2016). Each unit consists of two types of gates to track the state of the sequence, i.e. the reset gate $\mathbf{r}_t$ and the update gate $\mathbf{z}_t$. Given the embedding $\mathbf{v}_t$ of an incoming item (either a pre-trained amino acid embedding, or an output of the previous layer), GRU updates the hidden state $\mathbf{h}_t^{(3)}$ of the sequence as a linear combination of the previous state $\mathbf{h}_{t-1}^{(3)}$ and the candidate state $\tilde{\mathbf{h}}_t^{(3)}$ of a new item $\mathbf{v}_t$, which is calculated as below.

$$\begin{aligned}
\mathbf{h}_t^{(3)} &= \mathrm{GRU}(\mathbf{v}_t) = \mathbf{z}_t \odot \tilde{\mathbf{h}}_t^{(3)} + (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1}^{(3)} \\
\mathbf{z}_t &= \sigma\left(\mathbf{M}_z \mathbf{v}_t + \mathbf{N}_z \mathbf{h}_{t-1}^{(3)} + \mathbf{b}_z\right) \\
\tilde{\mathbf{h}}_t^{(3)} &= \tanh\left(\mathbf{M}_s \mathbf{v}_t + \mathbf{r}_t \odot (\mathbf{N}_s \mathbf{h}_{t-1}^{(3)}) + \mathbf{b}_s\right) \\
\mathbf{r}_t &= \sigma\left(\mathbf{M}_r \mathbf{v}_t + \mathbf{N}_r \mathbf{h}_{t-1}^{(3)} + \mathbf{b}_r\right).
\end{aligned}$$

$\odot$ thereof denotes the element-wise multiplication. The update gate $\mathbf{z}_t$ balances the information of the previous sequence and the new item, where capitalized $\mathbf{M}_*$ and $\mathbf{N}_*$ denote different weight matrices, $\mathbf{b}_*$ denote bias vectors, and σ is the sigmoid function. The candidate state $\tilde{\mathbf{h}}_t^{(3)}$ is calculated similarly to those in a traditional recurrent unit, and the reset gate $\mathbf{r}_t$ controls how much information of the past sequence contributes to $\tilde{\mathbf{h}}_t^{(3)}$. Note that GRU generally performs comparably to LSTM in sequence encoding tasks, but is less complex and requires much fewer computational resources for training.

A *bidirectional GRU layer* characterizes the sequential information in two directions. It contains the forward encoding process $\overrightarrow{\mathrm{GRU}}$ that reads the input vector sequence $X = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_l]$ from $\mathbf{v}_1$ to $\mathbf{v}_l$, and a backward encoding process $\overleftarrow{\mathrm{GRU}}$ that reads in the opposite direction. The encoding results of both processes are concatenated for each input item $\mathbf{v}_t$, i.e. $\mathbf{h}_t^{(4)} = \mathrm{BiGRU}(\mathbf{v}_t) = [\overrightarrow{\mathrm{GRU}}(\mathbf{v}_t), \overleftarrow{\mathrm{GRU}}(\mathbf{v}_t)]$.

The *residual mechanism* passes on an identity mapping of the GRU inputs to its output side through a residual shortcut (He *et al.*, 2016). By adding the forwarded input values to the outputs, the corresponding neural layer is only required to capture the difference between the input and output values. This mechanism aims at improving the learning process of non-linear neural layers by increasing the sensitivity of the optimization gradients (He *et al.*, 2016; Kim *et al.*, 2016), as well as preventing the model from the vanishing gradient problem. It has been widely deployed in deep learning architectures for various tasks of image recognition (He *et al.*, 2016), document classification (Conneau *et al.*, 2017) and speech recognition (Zhang *et al.*, 2017). In our deep RCNN, the bidirectional GRU is incorporated with the residual mechanism, and will pass on the following outputs to its subsequent neural network layer:

$$\mathbf{h}_t^{(5)} = \mathrm{ResGRU}(\mathbf{v}_t) = [\overrightarrow{\mathrm{GRU}}(\mathbf{v}_t) + \mathbf{v}_t, \overleftarrow{\mathrm{GRU}}(\mathbf{v}_t) + \mathbf{v}_t]$$

In our development, we have found that the residual mechanism is able to drastically simplify the training process, and largely decreases the epochs of parameter updates for the model to converge.

3.2.2 Protein Sequence Encoding

Fig 2. shows the entire structure of our RCNN encoder. The RCNN encoder $E_{RCNN}(S)$ alternately stacks multiple occurrences of the above two intermediary neural network components. A convolution layer serves as the first encoding layer to extract local features from the input sequence. On top of that, a residual GRU layer takes in the preserved local features, whose

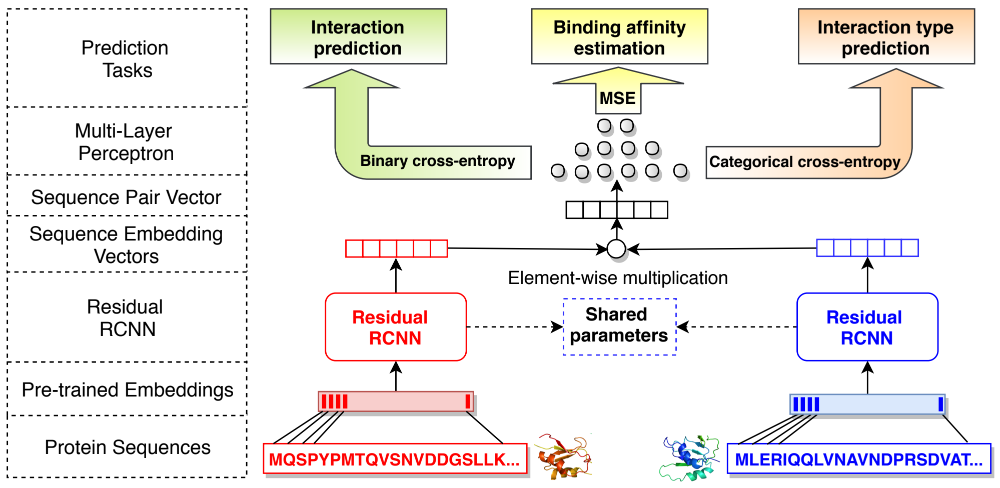


Fig. 1: The overall learning architecture of our framework.

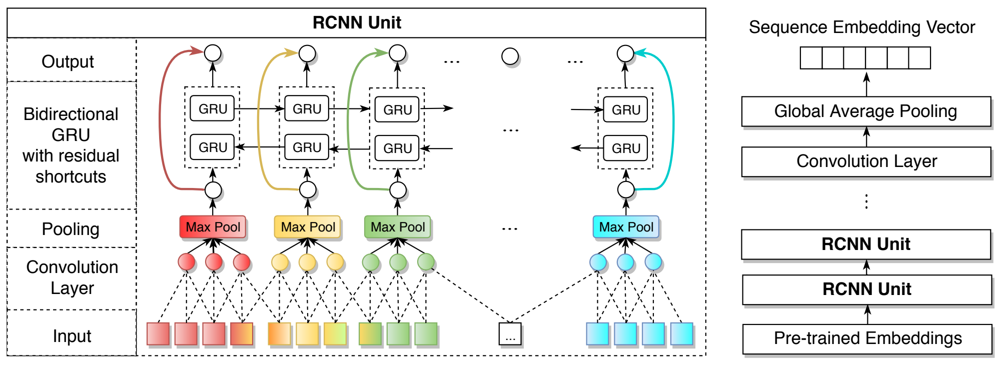


Fig. 2: The structure of our residual RCNN encoder is shown on the right, and the RCNN unit is shown on the left. Each RCNN unit contains a convolution-pooling layer followed a bidirectional residual GRU.

outputs are passed to another convolution layer. Repeating of these two components in the network structure conducts an automatic multi-granular feature aggregation process on the protein sequence, while preserving the sequential and contextualized information on each granularity of the selected features. The last residual GRU layer is followed by another convolution layer for a final round of local feature selection to produce the last hidden states $H' = [\mathbf{h}'_1, \mathbf{h}'_2, ..., \mathbf{h}'_{|H'|}]$. Note that the dimensionality of the last hidden states does not need to equal that of the previous hidden states. A high-level sequence embedding of the entire protein sequence is obtained from the global average-pooling (Lin *et al.*, 2013) of H', i.e. $E_{RCNN}(S) = \frac{1}{|H'|}\sum_{i=1}^{|H'|} \mathbf{h}'_i$.

3.2.3 Pre-trained Amino Acid Embeddings

To support inputting the non-numerical sequence information, we provide a useful embedding method to represent each animo acid $a \in A$ as a semi-latent vector $\mathbf{a}$. Each embedding vector is a concatenation of two sub-embeddings, i.e. $\mathbf{a} = [\mathbf{a}_c, \mathbf{a}_{ph}]$.

The first part $\mathbf{a}_c$ measures the co-occurrence similarity of the amino acids, which is obtained by pre-training the Skip-Gram model (Mikolov *et al.*, 2013) on protein sequences. The learning objective of Skip-Gram is to minimize the following negative log likelihood loss.

$$J_{SG} = -\frac{1}{|S|}\sum_{a_t \in S}\sum_{-C<j<C} \log p(\mathbf{a}_{c,t+j}|\mathbf{a}_{c,t})$$

$\mathbf{a}_{c,t}$ thereof is the first-part embedding of the t-th amino acid $a_t \in S$, $\mathbf{a}_{c,t+j}$ is that of a neighboring amino acid, and C is the size of half context[1]. The probability p is defined as the following softmax:

$$p(\mathbf{a}_{c,t+j}|\mathbf{a}_{c,t}) = \frac{\exp(\mathbf{a}_{c,t+j}\cdot \mathbf{a}_{c,t})}{\sum_{k=1}^{n}\exp(\mathbf{a}'_{c,k}\cdot \mathbf{a}_{c,t})}$$

where n is the negative sampling size, and $\mathbf{a}'_{c,k}$ is a negative sample that does not co-occur with $\mathbf{a}_{c,t}$ in the same context.

The second part $\mathbf{a}_{ph}$ represents the similarity of electrostaticity and hydrophobicity among amino acids. The 20 amino acids can be clustered into seven classes based on their dipoles and volumes of the side chains to reflect this property. Thus, $\mathbf{a}_{ph}$ is a one-hot encoding based on the classification defined by Shen *et al.* (2007).

3.3 Learning Architecture and Learning Objectives

Our framework characterizes the interactions in the following two stages.

3.3.1 Siamese Architecture

Given a pair of proteins $p = (S_1, S_2) \in I$, the same RCNN encoder is used to obtain the sequence embeddings $E_{RCNN}(S_1)$ and $E_{RCNN}(S_2)$ of both proteins. Both sequence embeddings are combined using element-wise multiplication, i.e., $E_{RCNN}(S_1) \odot E_{RCNN}(S_2)$. This is a

[1] The context of Skip-Gram means a subsequence of a given protein sequence S, such that the subsequence is of $2C + 1$ length.

commonly used operation to infer the relation of sequence embeddings (Tai *et al.*, 2015; Rocktäschel *et al.*, 2016; Hashemifar *et al.*, 2018; Jiang *et al.*, 2018). Note that some works use the concatenation of sequence embeddings (Yin and Schütze, 2015; Sun *et al.*, 2017) instead of multiplication, which we find to be less effective in modeling the symmetric relations of proteins.

3.3.2 Learning Objectives

A multi-layer perceptron (MLP) with leaky ReLU (Maas *et al.*, 2013) is applied to the previous sequence pair representation, whose output $\hat{s}^p$ is either a vector or a scalar, depending on whether the model solves a classification or a regression task for the protein pair p. The entire learning architecture is trained to optimize the following two types of losses according to different PPI prediction problems.

(i) *Cross-entropy loss* is optimized for the two classification problems, i.e. binary prediction and interaction type prediction. In this case, the MLP output $\hat{s}^p$ is a vector, whose dimensionality equals the number of classes m. $\hat{s}^p$ is normalized by a softmax function, where the i-th dimension $s_i^p = \frac{\exp(\hat{s}_i^p)}{\sum_j \exp(\hat{s}_j^p)}$ corresponds to the confidence score for the i-th class. The learning objective is to minimize the following cross-entropy loss, where c^p is a one-hot indicator for the class label of protein pair p.

$$L^{(1)} = -\frac{1}{|I|}\sum_{p\in I}\sum_{i=1}^{m} c_i^p \log s_i^p$$

(ii) *Mean squared loss* is optimized for the binding affinity estimation task. In this case, $\hat{s}^p$ is a scalar output that is normalized by a sigmoid function $s^p = \frac{1}{1+\exp(\hat{s}^p)}$, which is trained to approach the normalized ground truth score $c^p \in [0, 1]$ by minimizing the following objective function:

$$L^{(2)} = \frac{1}{|I|}\sum_{p\in I} |s^p - c^p|^2$$

4 Experiments

We present the experimental evaluation of the proposed framework on three PPI prediction tasks, i.e. binary prediction, multi-class interaction type prediction, and binding affinity estimation. The experiments are conducted on the following datasets.

4.1 Datasets

Guo's Datasets. Guo *et al.* (2008) generate several datasets from different species for the binary prediction of PPIs. Each dataset contains a balanced number of positive and negative samples. Among these resources, the *Yeast dataset* is a widely used benchmark by most state-of-the-art methods (Hashemifar *et al.*, 2018; Wong *et al.*, 2015; You *et al.*, 2013, 2014). There are 2,497 proteins forming 11,188 cases of PPIs, with half of them representing the positive cases, and the other half the negative cases. The positive cases are selected from the database of interacting proteins DIP_20070219 (Salwinski *et al.*, 2004), where proteins with fewer than 50 amino acids or $\geq$ 40% sequence identity are excluded. We use the full protein sequences in our model, which are obtained from the UniProt (Consortium *et al.*, 2018). The negative cases are generated by randomly pairing the proteins without evidence of interaction, and filtered by their sub-cellular locations. In other words, non-interactive pairs residing in the same location are excluded.

In addition, we combine the data for *C.elegans*, *E.coli*, and *Drosophila* as the *multi-species dataset*. We use the cluster analysis of the CD-HIT (Li and Godzik, 2006) program to generate non-redundant subsets. Proteins with fewer than 50 amino acids or high sequence identify (40%, 25%, 10%, or 1%) are removed.

STRING Datasets. The STRING database (Szklarczyk *et al.*, 2016) annotates PPIs with their types. There are seven types of interactions: activation, binding, catalysis, expression, inhibition, post-translational modification (ptmod), and reaction. We download all interaction pairs for *Homo sapiens* from database version 10.5 (Szklarczyk *et al.*, 2016), along with their full protein sequences. Among the corresponding proteins, we randomly select 3,000 proteins and 8,000 proteins that share less than 40% of sequence identity to generate two subsets. In this process, we randomly sample instances of different interaction types to ensure a balanced class distribution. Eventually, the two generated datasets, denoted by SHS27k and SHS148k, contain 26,945 cases and 148,051 cases of interactions respectively. We use these two datasets for the PPI type prediction task.

SKEMPI Dataset. We obtain the protein binding affinity data from SKEMPI (the Structural database of Kinetics and Energetics of Mutant Protein Interactions) (Moal and Fernández-Recio, 2012) for the affinity estimation task. It contains 3,047 binding affinity changes upon mutation of protein sub-units within a protein complex. The binding affinity is measured by equilibrium dissociation constant (K_d), reflecting the strength of biomolecular interactions. The smaller K_d value means the higher binding affinity. Each protein complex contains single or multiple amino acid substitutions.The sequence of the protein complex is retrieved from the Protein Data Bank (PDB) (Berman *et al.*, 2000). We manually replace the mutated amino acids. For duplicate entries, we take the average K_d. The final dataset results in the binding affinity of 2,792 mutant protein complexes, along with 158 wild-types.

4.2 Binary PPI Prediction

Binary PPI prediction is the primary task targeted by a handful of previous works (Shen *et al.*, 2007; Sun *et al.*, 2017; You *et al.*, 2015; Hashemifar *et al.*, 2018; Yang *et al.*, 2010). The objective of these works is to identify whether a given pair of proteins interacts or not based on their sequences. We evaluate PIPR based on Guo's datasets. The Yeast benchmark dataset thereof is used to compare PIPR with various baseline approaches, and the multi-species dataset is to demonstrate PIPR's capability of predicting interactions for proteins of different species that share very low sequence identity with those in training.

The baseline approaches include SVM-AC (Guo *et al.*, 2008), kNN-CTD (Yang *et al.*, 2010), EELM-PCA (You *et al.*, 2013), SVM-MCD (You *et al.*, 2014), MLP (Du *et al.*, 2017), Random Forest LPQ (RF-LPQ) (Wong *et al.*, 2015), SAE (Sun *et al.*, 2017), DNN-PPI (Li *et al.*, 2018) and DPPI (Hashemifar *et al.*, 2018). In addition, we report the results of a Siamese Residual GRU (SRGRU) architecture, which is a simplification of PIPR, where we discard all intermediary convolution layers and keep only the bidirectional residual GRU. The purpose of SRGRU is to show the significance of the contextualized and sequential information of protein profiles in characterizing PPIs. We also report the results of Siamese CNN (SCNN) by removing the residual GRU in PIPR. This degenerates our framework to a similar architecture to DPPI, but differs in that SCNN directly conducts an end-to-end training on raw sequences instead of requiring the protein profiles constructed by PSI-BLAST.

We use AMSGrad (Reddi *et al.*, 2018) to optimize the cross-entropy loss, for which we set the learning rate α to 0.001, the exponential decay rates β_1 and β_2 to 0.9 and 0.999, and batch size to 256 on both datasets. The number of occurrences for the RCNN units (i.e., one convolution-pooling layer followed by one bidirectional residual GRU layer) is set to 5, where we adopt 3-max-pooling and the convolution kernel of size 3. We set the hidden state size to be 50, and the RCNN output size to be 100. We set this configuration to ensure the RCNN to compress the selected features in a reasonably small vector sequence, before the features are aggregated by the last global average-pooling. We zero-pad short sequences to the longest sequence length in the dataset. This is a widely adopted technique

Table 1. Evaluation of binary PPI prediction on the Yeast dataset based on 5-fold cross-validation. We report the mean and standard deviation for the test sets.

Methods	Accuracy (%)	Precision (%)	Sensitivity (%)	Specificity (%)	F1-score (%)	MCC(%)
SVM-AC	87.35 ± 1.38	87.82 ± 4.84	87.30 ± 5.23	87.41 ± 6.33	87.34 ± 1.33	75.09 ± 2.51
kNN-CTD	86.15 ± 1.17	90.24 ± 1.34	81.03 ± 1.74	NA	85.39 ± 1.51	NA
EELM-PCA	86.99 ± 0.29	87.59 ± 0.32	86.15 ± 0.43	NA	86.86 ± 0.37	77.36 ± 0.44
SVM-MCD	91.36 ± 0.4	91.94 ± 0.69	90.67 ± 0.77	NA	91.3 ± 0.73	84.21 ± 0.66
MLP	94.43 ± 0.3	96.65 ± 0.59	92.06 ± 0.36	NA	94.3 ± 0.45	88.97 ± 0.62
RF-LPQ	93.92 ± 0.36	96.45 ± 0.45	91.10 ± 0.31	NA	93.7 ± 0.37	88.56 ± 0.63
SAE	67.17 ± 0.62	66.90 ± 1.42	68.06 ± 2.50	66.30 ± 2.27	67.44 ± 1.08	34.39 ± 1.25
DNN-PPI	76.61 ± 0.51	75.1 ± 0.66	79.63 ± 1.34	73.59 ± 1.28	77.29 ± 0.66	53.32 ± 1.05
DPPI	94.55	96.68	92.24	NA	94.41	NA
SRGRU	93.77 ± 0.84	94.60 ± 0.64	92.85 ± 1.58	94.69 ± 0.81	93.71 ± 0.85	87.56 ± 1.67
SCNN	95.03 ± 0.47	95.51 ± 0.77	94.51 ± 1.27	95.55 ± 0.77	95.00 ± 0.50	90.08 ± 0.93
`PIPR`	**97.09 ± 0.24**	**97.00 ± 0.65**	**97.17 ± 0.44**	97.00 ± 0.67	**97.09 ± 0.23**	**94.17 ± 0.48**

Table 2. Statistical assessment (t-test; two-tailed) on the accuracy of binary PPI prediction. The statistically significant differences are highlighted in red.

p-value	SRGRU	SCNN	`PIPR`
SVM-AC	9.69E-05	1.22E-04	9.69E-05
kNN-CTD	1.03E-05	2.23E-05	2.84E-05
EELM-PCA	2.33E-05	3.94E-08	2.43E-10
SVM-MCD	1.67E-03	2.60E-06	1.35E-07
MLP	1.71E-01	5.29E-02	1.12E-06
RF-LPQ	7.28E-01	4.10E-03	1.75E-06
SAE	4.27E-10	1.78E-10	4.19E-09
DNN-PPI	1.62E-08	2.27E-10	2.70E-09
SRGRU	NA	2.87E-02	6.60E-04
SCNN	2.87E-02	NA	1.80E-04

for sequence modeling in NLP (Yin *et al.*, 2016; Hu *et al.*, 2014; Zhou *et al.*, 2017; Chen *et al.*, 2018; He *et al.*, 2015) as well as in bioinformatics (Pan and Shen, 2018; Müller *et al.*, 2018; Min *et al.*, 2017) for efficient training. Note that the configuration of embedding pre-training is discussed in Section 4.5, and the model configuration study of different hyperparameter values is provided in the Supplementary Materials. All model variants are trained until converge at each fold of the cross-validation.

Evaluation protocol. Following the settings in previous works (Shen *et al.*, 2007; Sun *et al.*, 2017; You *et al.*, 2014, 2015; Hashemifar *et al.*, 2018), we conduct 5-fold cross-validation (CV) on the Yeast dataset. Under the k-fold CV setting, the data is equally divided into k non-overlapping subsets, and each subset has a chance to train and to test the model so as to ensure an unbiased evaluation. We aggregate fix metrics on the test cases of each fold, i.e. the overall *accuracy*, *precision*, *sensitivity*, *specificity*, *F1*, and *Matthews correlation coefficient (MCC)* on positive cases. All these metrics are preferred to be higher to indicate better performance. Based on the reported accuracy over 5-folds, we also conduct two-tailed Welch's t-tests (Welch, 1947) to evaluate the significance of the improvement on different pairs of approaches. The p-values are adjusted by the Benjamini-Hochberg procedure (Benjamini and Hochberg, 1995) to control the false discovery rate for multiple hypothesis testing.

Results. As shown in Table 1, the CNN-based architecture, DPPI, demonstrates state-of-the-art performance over other baselines that employ statistical learning algorithms or densely connected MLP. This shows the superiority of deep-learning-based techniques[2] in encapsulating various types of information of a protein pair, such as amino acid composition and their co-occurrences, and automatically extracting the robust ones for the learning objectives. That said, DPPI requires an extensive effort in data pre-processing, specifically in constructing the protein profile for each sequence. On average, each PSI-BLAST search of a protein against the NCBI non-redundant protein database (184,243,125 sequences) requires around 90 minutes of computation on our server. Even with eight cores, each search finishes in 15 minutes. We estimate that processing 2,497 sequences of the Yeast dataset from scratch can take about 26 days. It is worth mentioning that `PIPR` only requires 8 seconds to pre-train the amino acid embedding, and 2.5 minutes to train on the Yeast dataset (see Table 7). We implement SCNN to evaluate the performance of a simplified CNN architecture, which produces comparable results as DPPI. These two frameworks show that CNN can already leverage the significant features from primary protein sequences.

[2] We are unable to obtain the source codes of two deep-learning methods, SAE and DNN-PPI. We implement these two models following the descriptions in their papers. Our implementations are verified by achieving comparable performance on the Pan's dataset (Pan *et al.*, 2010) as reported in the papers. However, these two implementations can only achieve 67.17% and 76.61% in overall accuracy respectively on the Yeast dataset.

Table 3. Evaluation of binary PPI prediction on variants of multi-species (C. elegan, Drosophila, and E. coli) dataset.

Seq. Identity	# of Proteins	Pos. Pairs	Neg. Pairs	Accuracy (%)	F1-Score (%)
Any	11529	32959	32959	98.19	98.17
<0.40	9739	25916	22012	98.29	98.28
<0.25	7790	19458	15827	97.91	98.08
<0.10	5769	12641	9819	97.54	97.79
<0.01	5171	10747	8065	97.51	97.80

In addition, the SRGRU architecture has offered comparable performance to SCNN. This indicates that preserving the sequential and contextualized features of the protein sequences is as crucial as incorporating the local features. By integrating both significant local features and sequential information, `PIPR` outperforms DPPI by 2.54% in accuracy, 4.93% in sensitivity, and 2.68% in F1-Score. Next, we evaluate whether the improved accuracy of `PIPR` is statistically significant. Table 2 reports the p-values of SRGRU, SCNN, and `PIPR` compared to other baseline approaches, where the statistically significant comparisons (p-values < 0.01) are highlighted in red. Since the standard deviation of DPPI is unavailable, we are not able to include DPPI in this analysis. The evaluation shows that `PIPR` performs statistically significantly better than all other approaches, including SCNN and SRGRU. On the other hand, SCNN is not statistically significantly better than SRGRU. Thus, the residual RCNN is very promising for modeling binary PPIs.

We also report the 5-fold CV performance of `PIPR` on variants of the multi-species dataset, where proteins are excluded based on different thresholds of sequence identity. The results in Table 3 show that `PIPR` performs consistently well under lenient and stringent criteria of sequence identity between training and testing. More importantly, `PIPR` is able to train and test on multiple species, and is robust against extremely low sequence identity of less than 1%.

4.3 Interaction Type Prediction

The objective of this task is to predict the interaction type of two interacting proteins. We evaluate this task based on SHS27k and SHS148k datasets. To the best of our knowledge, much fewer efforts attempt for the multi-class PPI prediction in contrast to the binary prediction. Zhu *et al.* (2006) train a two-stage SVM classifier to distinguish obligate, non-obligate, and crystal packing interactions; Silberberg *et al.* (2014) use logistic regression to predict several types of enzymatic actions. However, none of their implementations are publicly available. Different from the categories of interaction types used above, we aim at predicting the interaction types annotated by the STRING database.

Table 4. Accuracy (%) and fold changes over zero rule for PPI interaction type prediction on two STRING datasets based on 10-fold cross-validation.

Features	N/A		AC					CTD					Embedded raw seqs		
Methods	Rand	Zero rule	SVM	RF	AdaBoost	kNN	Logistic	SVM	RF	AdaBoost	kNN	Logistic	SCNN	SRGRU	PIPR
SHS27k	14.28	16.70	33.17	44.82	28.67	35.44	25.47	35.56	45.76	31.81	35.56	30.57	55.54	51.06	**59.56**
(fold×)	—	1.00×	1.99×	2.68×	1.72×	2.12×	1.52×	2.13×	2.74×	1.90×	2.13×	1.83×	3.33×	3.06×	**3.57**×
SHS148k	14.28	16.21	28.17	36.01	27.87	33.81	24.96	31.37	36.65	29.67	33.13	26.96	55.29	54.05	**61.91**
(fold×)	—	1.00×	1.74×	2.22×	1.72×	2.09×	1.54×	1.94×	2.26×	1.83×	2.04×	1.66×	3.41×	3.33×	**3.82**×

Table 5. Results for binding affinity prediction on the SKEMPI dataset. Each measurement is an average of the test sets over 10-fold cross-validation.

Features	AC				CTD				Embedded raw seqs		
Methods	BR	SVM	RF	AdaBoost	BR	SVM	RF	AdaBoost	SCNN	SRGRU	PIPR
MSE$(\times 10^{-2})$	1.70	2.20	1.77	1.98	1.86	1.84	1.49	1.84	0.87	0.95	**0.63**
MAE$(\times 10^{-2})$	9.56	11.81	9.81	11.15	10.20	11.04	9.06	10.69	6.49	7.08	**5.48**
Corr	0.564	0.353	0.546	0.451	0.501	0.501	0.640	0.508	0.831	0.812	**0.873**

We train several statistical learning algorithms on the widely employed AC and CTD features for protein characterization as our baselines. These algorithms include SVM, Random Forest, Adaboost (SAMME.R algorithm (Zhu *et al.*, 2009)), kNN classifier, and logistic regression. For deep-learning-based approaches, we deploy the SCNN architecture where an output MLP with categorical cross-entropy loss is incorporated, as well as a similar SRGRU architecture into comparison. Results of two naïve baselines of random guessing and zero rule (i.e., simply predicting the majority class) are also reported for reference.

Evaluation protocol. All approaches are evaluated on the two datasets by 10-fold CV, using the same partition scheme for a more unbiased evaluation (McLachlan *et al.*, 2005; James *et al.*, 2013). We carry forward the model configurations from the last experiment to evaluate the performance of the frameworks under controlled variables. For baseline models, we examine three different ways of combining the feature vectors of the two input proteins, i.e. element-wise multiplication, the Manhattan difference (i.e. the absolute differences of corresponding features (Mueller and Thyagarajan, 2016)) and concatenation. The Manhattan difference consistently obtains better performance, considering the small values of the input features and the asymmetry of the captured protein relations.

Results. The prediction accuracy and fold changes over the zero rule baseline are reported in Table 4. Note that since the multi-class prediction task is much more challenging than the binary prediction task, it is expected to observe lower accuracy and longer training-time (Table 7) than that reported in the previous experiment. Among all the baselines using explicit features, the CTD-based models perform better than the AC-based ones. CTD descriptors seek to cover both continuous and discontinuous interaction information (Yang *et al.*, 2010), which potentially better discriminate among PPI types.

The best baseline using Random Forest thereof achieves satisfactory results by more than doubling the accuracy of zero rule on the smaller SHS27k dataset. However, on the larger SHS148k dataset, the accuracy of these explicit-feature-based models is notably impaired. We hypothesize that such predefined explicit features are not representative enough to distinguish the PPI types. On the other hand, the deep-learning-based approaches do not need to explicitly utilize these features, and perform consistently well in both settings. The raw sequence information is sufficient for these approaches to drastically outperform the Random Forest by at least 5.30% in accuracy on SHS27k and 17.40% in accuracy on SHS148k. SCNN thereof outperforms SRGRU by 4.48% and 1.24% in accuracy on SHS27k and SHS148k, respectively. This implies that the local interacting features are relatively more deterministic than contextualized and sequential features on this task. The results by the residual RCNN-based framework are very promising, as it outperforms SCNN by 4.02% and 6.62% in accuracy on SHS27k and SHS148k respectively. It also remarkably outperforms the best explicit-feature-based baselines on the two datasets by 13.80% and 25.26% in accuracy, and more than 3.5 of fold changes over the zero rule on both datasets.

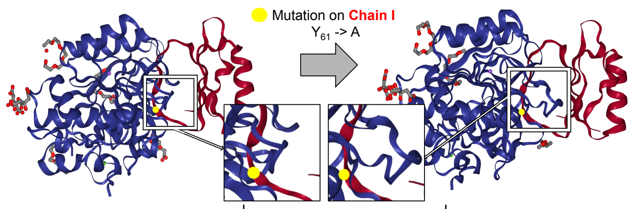
4.4 Binding Affinity Estimation

Lastly, we evaluate PIPR for binding affinity estimation using the SKEMPI dataset. We employ the mean squared loss variant of PIPR to address this regression task. Since the lengths of protein sequences in SKEMPI are much shorter than those in the other datasets, we accordingly reduce the occurrences of RCNN units to 3, while other configurations remain unchanged. For baselines, we compare against several regression models based on the AC and CTD features, which include Bayesian Redge regressor (BR), SVM, Adaboost with decision tree regressors and Random Forest regressor. The corresponding features for two sequences are again combined via the Manhattan difference. We also modify SCNN and SRGRU to their mean squared loss variants, in which we reduce the layers in the same way of RCNN.

Evaluation protocol. We aggregate three metrics through 10-fold CV, i.e. *mean squared error* (*MSE*), *mean absolute error* (*MAE*) and *Pearson's correlation coefficient* (*Corr*). These are three commonly reported metrics for regression tasks, for which lower *MSE* and *MAE* as well as higher *Corr* indicate better performance. In the cross-validation process, we normalize the affinity values of the SKEMPI dataset to $[0, 1]$ via min-max re-scaling[3].

Results. Table 5 reports the results for this experiment. It is noteworthy that, one single change of amino acid can lead to a drastic effect on binding affinity. While such subtle changes are difficult to be reflected by the explicit features, the deep-learning-based methods can competently capture such changes from the raw sequences. Our RCNN-based framework again offers the best performance among the deep-learning-based approaches, and significantly outperforms the best baseline (CTD-based Random Forest) by offering a 0.233 increase in *Corr*, as well as remarkably lower *MSE* and *MAE*. Figure 3 demonstrates an example of the effect of changing an amino acid in a protein complex. Tyrosine at position 61 of Chymotrypsin inhibitor 2 (Chain I) is substituted with Alanine, causing the neighboring region of Subtilisin BPN' precursor (Chain E) to relax. The binding affinity (k_d) changes from 2.24E-12 to 2.70E-10, which is validly captured by PIPR. While our experiment is conducted on a relatively small dataset, we seek to extend our PIPR framework to a more generalized solution for binding affinity estimation, once a larger and more heterogeneous corpus is available.

[3] This is due to that we use sigmoid function to smooth the output of the regressor. Note that this process does not affect correlation, while MSE, MAE and the original affinity scores can be easily re-scaled back.



PDB ID	1TM1 (wild type)	1TO1 (mutant)
Binding Affinity (k_d)	2.24E-12	2.70E-10
PIPR Prediction (k_d)	2.54E-12	3.52E-10

Fig. 3: Mutation effects on structure and binding affinity. The blue entity is Subtilisin BPN' precursor (Chain E), and the red entity is Chymotrypsin inhibitor (Chain I). The mutation is highlighted in yellow. The wild type (1TM1) and mutant (1TO1) complexes are retrieved from PDB.

Table 6. Comparison of amino acid representations based on binary prediction.

	$[\mathbf{a}_c, \mathbf{a}_{ph}]$	$\mathbf{a}_c$ only	$\mathbf{a}_{ph}$ only	One-hot
Dimension	12	5	7	20
Accuracy	**97.09**	96.67	96.03	96.11
Precision	**97.00**	96.35	95.91	96.34
F1-Score	**97.09**	96.51	96.08	96.10

Table 7. Run-Time of training embeddings and different prediction tasks.

Task	Embeddings	Binary	Multi-class	Multi-class	Regression
Dataset	**SHS148k**	**Yeast**	**SHS27k**	**SHS148k**	**SKEMPI**
Sample Size	8,000	11,188	26,945	148,051	2,950
Training Time	8sec	2.5min	15.8min	138.3min	12.5min

4.5 Amino Acid Embeddings

We further investigate the settings of amino acid embeddings in this subsection. Each amino acid is represented by a vector of numerical values that describe its relative physicochemical properties. The first part of the embedding vector $\mathbf{a}_c$, which measures the co-occurrence similarity of the amino acids in protein sequences, is empirically set as a 5-dimensional vector. $\mathbf{a}_c$ is obtained by pre-training the Skip-Gram model on all 8,000 sequences from our largest STRING dataset, SHS148k, using a context window size of 7 and a negative sampling size of 5. The second part contains a 7-dimensional vector, $\mathbf{a}_{ph}$, which describes the categorization of electrostaticity and hydrophobicity for the amino acid. We examine the performance of using each part individually, as well as the performance of combining them as used in our framework. In addition, we include a naïve one-hot vector representation, which does not consider the relatedness of amino acids and treats each of them independently. Table 6 shows that, once we remove either of the two parts of the proposed embedding, the performance of the model slightly drops. Meanwhile, the proposed pre-trained embeddings lead to noticeably better performance of the model than adopting the naïve one-hot encodings of the canonical amino acids. This pre-training process completes in 8 seconds on a commodity workstation as shown in Table 7. This is a one-time effort that can be reused on different tasks and datasets.

4.6 Run-time Analysis

All of the experiments are conducted on one NVIDIA GeForce GTX 1080 Ti GPU. We report the training time for each experiment, as well as for the amino acid embedding in Table 7. For each experiment, we calculate the average training time over either 5-fold (Yeast dataset) or 10-fold (others) CV. In both binary and multi-class predictions, the training time increases along with the increased number of training cases. The regression estimation generally requires more iterations per training case to converge than classification tasks. Thus, with much fewer cases, the training time on SKEMPI for affinity estimation is more than that on the Yeast dataset for binary prediction.

5 Conclusion

In this paper, we propose a novel end-to-end framework for PPI prediction based on the amino acid sequences. Our proposed framework, PIPR, employs a residual RCNN, which provides an automatic multi-granular feature selection mechanism to capture both local significant features and sequential features from the primary protein sequences. By incorporating the RCNN in a Siamese-based learning architecture, the framework captures effectively the mutual influence of protein pairs, and generalizes well to address different PPI prediction tasks without the need for predefined features. Extensive experimental evaluations on five datasets show promising performance of our framework on three challenging PPI prediction tasks. This also leads to significant amelioration over various baselines. Experiments on datasets of different sizes also demonstrate satisfactory scalability of the framework. For future work, one important direction is to apply the PIPR framework to other sequence-based inference tasks in bioinformatics, such as modeling RNA and protein interactions. We also seek to incorporate attention mechanisms (Vaswani *et al.*, 2017) to help pinpoint interaction sites on protein sequences, and apply PIPR to predict confidence of interactions in the form of ordinal regression. Since PIPR has alleviated any costly domain-invariant feature engineering process, how to extend PIPR with transfer learning based domain adaptation for different species is another meaningful direction.

Funding

We thank all of the reviewers for their valuable comments and suggestions. This work was partially supported by the National Institutes of Health [R01GM115833,U54 GM114833]; and the National Science Foundation [DBI-1565137,DGE-1829071].

References

Altschul, S. F. *et al.* (1997). Gapped blast and psi-blast: a new generation of protein database search programs. *NAR*, **25**(17), 3389–3402.

Anderson, C. (2018). Google's ai tool deepvariant promises significantly fewer genome errors. *Clinical OMICs*, **5**(1), 33–33.

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the royal statistical society. Series B (Methodological)*, pages 289–300.

Berman, H. M. *et al.* (2000). The protein data bank. *NAR*, **28**(1), 235–242.

Chen, M. *et al.* (2018). Neural article pair modeling for wikipedia sub-article matching. In *ECML-PKDD*, pages 3–19.

Cho, K. *et al.* (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*.

Conneau, A. *et al.* (2017). Very deep convolutional networks for text classification. In *EACL*.

Consortium, U. *et al.* (2018). Uniprot: the universal protein knowledgebase. *Nucleic acids research*, **46**(5), 2699.

Dhingra, B. *et al.* (2016). Gated-attention readers for text comprehension. In *ACL*.

Du, X. *et al.* (2017). Deepppi: boosting prediction of protein–protein interactions with deep neural networks. *JCIM*, **57**(6), 1499–1510.

Fields, S. and Song, O.-k. (1989). A novel genetic system to detect protein–protein interactions. *Nature*, **340**(6230), 245.

Gavin, A.-C. *et al.* (2002). Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, **415**(6868), 141.

Guo, Y. *et al.* (2008). Using support vector machine combined with auto covariance to predict protein-protein interactions from protein sequences. *NAR*, **36**(9), 3025–30.

Hashemifar, S. *et al.* (2018). Predicting protein–protein interactions through sequence-based deep learning. *Bioinformatics*, **34**(17).

He, H. *et al.* (2015). Multi-perspective sentence similarity modeling with convolutional neural networks. In *EMNLP*, pages 1576–1586.

He, K. *et al.* (2016). Deep residual learning for image recognition. In *CVPR*, pages 770–778.

Ho, Y. *et al.* (2002). Systematic identification of protein complexes in saccharomyces cerevisiae by mass spectrometry. *Nature*, **415**, 180.

Hu, B. *et al.* (2014). Convolutional neural network architectures for matching natural language sentences. In *NIPS*, pages 2042–2050.

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 research intl.*, **2015**.

James, G. *et al.* (2013). *An introduction to statistical learning*, volume 112. Springer.

Jansen, R. *et al.* (2003). A Bayesian networks approach for predicting protein-protein interactions from genomic data. *Science*, **302**, 449–53.

Jiang, J.-Y. *et al.* (2018). Learning to disentangle interleaved conversational threads with a siamese hierarchical network and similarity ranking. In *NAACL*.

Kim, J. *et al.* (2016). Accurate image super-resolution using very deep convolutional networks. In *CVPR*, pages 1646–1654.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *EMNLP*.

LeCun, Y. *et al.* (2015). Deep learning. *nature*, **521**(7553), 436.

Li, H. *et al.* (2018). Deep neural network based predictions of protein interactions using primary sequences. *Molecules*, **23**(8), 1923.

Li, W. and Godzik, A. (2006). Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, **22**(13), 1658–1659.

Lin, M. *et al.* (2013). Network in network. In *ICLR*.

Maas, A. L. *et al.* (2013). Rectifier nonlinearities improve neural network acoustic models. In *ICML*, volume 30, page 3.

McLachlan, G. *et al.* (2005). *Analyzing microarray gene expression data*, volume 422. John Wiley & Sons.

Mikolov, T. *et al.* (2013). Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Min, X. *et al.* (2017). Chromatin accessibility prediction via convolutional long short-term memory networks with k-mer embedding. *Bioinformatics*, **33**(14), i92–i101.

Moal, I. H. and Fernández-Recio, J. (2012). Skempi: a structural kinetic and energetic database of mutant protein interactions and its use in empirical models. *Bioinformatics*, **28**(20), 2600–2607.

Mueller, J. and Thyagarajan, A. (2016). Siamese recurrent architectures for learning sentence similarity. In *AAAI*, volume 16, pages 2786–2792.

Müller, A. T. *et al.* (2018). Recurrent neural network model for constructive peptide design. *JCIM*, **58**(2), 472–479.

Pan, X. and Shen, H.-B. (2018). Predicting rna–protein binding sites and motifs through combining local and global deep convolutional neural networks. *Bioinformatics*, **34**(20).

Pan, X.-Y. *et al.* (2010). Large-scale prediction of human protein- protein interactions from amino acid sequence based on latent topic features. *Journal of Proteome Research*, **9**(10), 4992–5001.

Pascanu, R. *et al.* (2013). On the difficulty of training recurrent neural networks. In *ICML*, pages 1310–1318.

Petta, I. *et al.* (2016). Modulation of protein–protein interactions for the development of novel therapeutics. *Molecular Therapy*, **24**(4), 707–718.

Philipp, O. *et al.* (2016). Path2ppi: an r package to predict protein–protein interaction networks for a set of proteins. *Bioinformatics*, **32**(9).

Quang, D. and Xie, X. (2016). Danq: a hybrid convolutional and recurrent deep neural network for quantifying the function of dna sequences. *NAR*, **44**(11).

Reddi, S. J. *et al.* (2018). On the convergence of adam and beyond. In *ICLR*.

Rocktäschel, T. *et al.* (2016). Reasoning about entailment with neural attention. In *ICLR*.

Salwinski, L. *et al.* (2004). The database of interacting proteins: 2004 update. *NAR*, **32**(suppl_1), D449–D451.

Shen, J. *et al.* (2007). Predicting protein-protein interactions based only on sequences information. *PNAS*, **104**(11), 4337–41.

Silberberg, Y. *et al.* (2014). A method for predicting protein-protein interaction types. *PLoS One*, **9**(3).

Skrabanek, L. *et al.* (2008). Computational prediction of protein–protein interactions. *Molecular biotechnology*, **38**(1), 1–17.

Srinivasulu, Y. S. *et al.* (2015). Characterizing informative sequence descriptors and predicting binding affinities of heterodimeric protein complexes. *BMC bioinformatics*, **16**(18), S14.

Sun, T. *et al.* (2017). Sequence-based prediction of protein protein interaction using a deep-learning algorithm. *BMC bioinformatics*, **18**(1).

Szklarczyk, D. *et al.* (2016). The string database in 2017: quality-controlled protein–protein association networks, made broadly accessible. *NAR*.

Tai, K. S. *et al.* (2015). Improved semantic representations from tree-structured long short-term memory networks. In *ACL*, pages 1556–1566.

Vaswani, A. *et al.* (2017). Attention is all you need. In *NIPS*.

Wang, Y.-B. *et al.* (2017). Predicting protein–protein interactions from protein sequences by a stacked sparse autoencoder deep neural network. *Molecular BioSystems*, **13**(7), 1336–1344.

Welch, B. L. (1947). The generalization ofstudent's' problem when several different population variances are involved. *Biometrika*, **34**(1/2), 28–35.

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 *ICIC*, pages 713–720.

Yang, L. *et al.* (2010). Prediction of protein-protein interactions from protein sequence using local descriptors. *Protein and Peptide Letters*, **17**(9), 1085–1090.

Yin, W. and Schütze, H. (2015). Convolutional neural network for paraphrase identification. In *NAACL*, pages 901–911.

Yin, W. *et al.* (2016). Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *TACL*, **4**(1).

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**.

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.

You, Z.-H. *et al.* (2015). Predicting protein-protein interactions from primary protein sequences using a novel multi-scale local feature representation scheme and the random forest. *PLoS One*, **10**(5).

Yugandhar, K. and Gromiha, M. M. (2014). Protein–protein binding affinity prediction from amino acid sequence. *Bioinformatics*, **30**(24).

Zhang, S. *et al.* (2015). A deep learning framework for modeling structural features of rna-binding protein targets. *NAR*, **44**(4), e32–e32.

Zhang, Y. *et al.* (2017). Very deep convolutional networks for end-to-end speech recognition. In *ICASSP*, pages 4845–4849.

Zhou, T. *et al.* (2017). Attention-based natural language person retrieval. In *CVPR*, pages 27–34.

Zhu, H. *et al.* (2006). Noxclass: prediction of protein-protein interaction types. *BMC bioinformatics*, **7**(1), 27.

Zhu, J. *et al.* (2009). Multi-class adaboost. *Statistics and its Interface*, **2**(3).

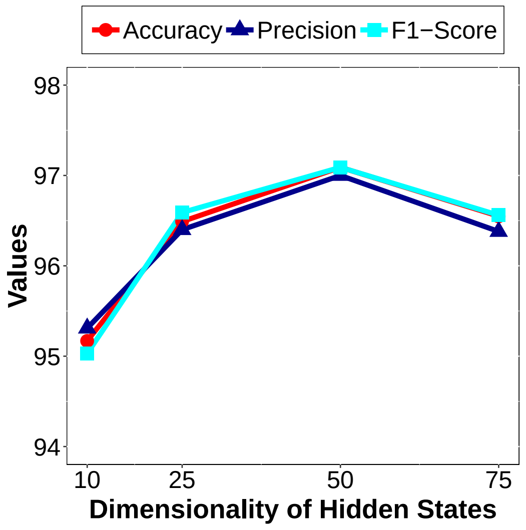
Supplementary Materials: Multifaceted Protein-Protein Interaction Prediction Based on Siamese Residual RCNN

Muhao Chen[1*], Chelsea J.-T. Ju [1*], Guangyu Zhou[1], Xuelu Chen[1], Tianran Zhang[2], Kai-Wei Chang[1], Carlo Zaniolo[1], and Wei Wang[1]

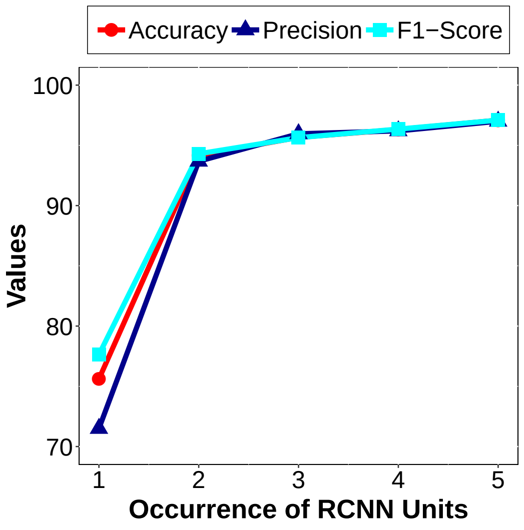
[1]Department of Computer Science, University of California, Los Angeles, 90095, USA
[2]Department of Bioengineering, University of California, Los Angeles, 90095, USA

S1 Hyperparameter Study

We examine the configuration of two critical factors that can affect the performance of our framework: the dimensionality of hidden states and the number of occurrences for the RCNN units. We show the effects of different settings of these two factors based on the binary PPI prediction task. The hidden state sizes are chosen from {10, 25, 50, 75}. As illustrated in Fig S1a, the performance of `PIPR` initially increases as we raise the dimensionality of the hidden states until it passes 50, and then starts to decline. The occurrences of RCNN units contribute to the levels of granularity in feature aggregation. Fewer occurrences correspond to less aggregation. However, too many occurrences can lead to over-compressing the features. We examine the occurrences from 1 to 5 based on Yeast. Note that we do not adopt the setting with 6 occurrences, where the RCNN encoder over-compresses the extracted features to a very small number of latent vectors before the last global average pooling. Aligned with our hypothesis, Fig S1b shows that the accuracy, precision, and F1-score improve when we increase the number of occurrences of the RCNN units. The improvement from 2 to 5 occurrences is marginal, which shows that our framework is robust to this setting as long as there are more than 2 occurrences of RCNN units.



(a) Hidden States



(b) RCNN Units

Fig. S1: Performance evaluation on dimensionality of hidden states, and the number of occurrences of the RCNN units.