IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 18, 2019, accepted July 2, 2019, date of publication July 8, 2019, date of current version July 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2927253

Deep Residual Convolutional Neural Network for Protein-Protein Interaction Extraction

HAO ZHANG[1], RENCHU GUAN[1,2], (Member, IEEE), FENGFENG ZHOU[1], (Senior Member, IEEE), YANCHUN LIANG[1,2], ZHI-HUI ZHAN[3,4], (Senior Member, IEEE), LAN HUANG[1,2], AND XIAOYUE FENG[1]

[1]Key Laboratory of Symbolic Computation and Knowledge Engineering of the Ministry of Education, College of Computer Science and Technology, Jilin University, Changchun 130012, China
[2]Zhuhai Sub Laboratory, Key Laboratory of Symbolic Computation and Knowledge Engineering of the Ministry of Education, Zhuhai College, Jilin University, Zhuhai 519041, China
[3]School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China
[4]Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, South China University of Technology, Guangzhou 510006, China

Corresponding author: Xiaoyue Feng (fengxy@jlu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61602207 and Grant 61572228, in part by the Key Technological Research Projects in Jilin Province under Grant 20190302107GX, in part by the Special Research and Development of Industrial Technology of Jilin Province under Grant 2019C053-7, in part by the Guangdong Premier Key-Discipline Enhancement Scheme under Grant 2016GDYSZDXK036, and in part by the Guangdong Key-Project for Applied Fundamental Research under Grant 2018KZDXM076.

ABSTRACT Knowledge extracted from the protein–protein interaction (PPI) network can help researchers reveal the molecular mechanisms of biological processes. With the rapid growth in the volume of the biomedical literature, manually detecting and annotating PPIs from raw literature has become increasingly difficult. Hence, automatically extracting PPIs by machine learning methods from raw literature has gained significance in the biomedical research. In this paper, we propose a novel PPI extraction method based on the residual convolutional neural network (CNN). This is the first time that the residual CNN is applied to the PPI extraction task. In addition, the previous state-of-the-art PPI extraction models heavily rely on parsing results from natural language processing tools, such as dependence parsers. Our model does not rely on any parsing tools. We evaluated our model based on five benchmark PPI extraction corpora, AIMed, BioInfer, HPRD50, IEPA, and LLL. The experimental results showed that our model achieved the best results compared with the previous kernel-based and CNN-based PPI extraction models. Compared with the previous recurrent neural network-based PPI extraction models, our model achieved better or comparable performance.

INDEX TERMS Deep learning, natural language processing, protein–protein interaction extraction, residual convolutional neural network.

I. INTRODUCTION

Protein-protein interaction (PPI) is a physical contact established between two or more protein molecules resulting from biochemical events, which provides a useful proxy for cellular communication lattices and can be discovered in almost all cellular processes, such as metabolism, signaling, regulation, and proliferation [1], [2]. Compiling protein-protein interaction networks are meaningful to system biology research [3]. One of the most important ways to construct a PPI network is to curate PPIs from literature. For example, Kwon *et al.* [4] manually curated a Death Domain superfamily PPI database from 295 biomedical papers. The rapid growth of the biomedical literature makes the manual search of protein homologs and PPI annotation almost impossible [5]. Until May 2019, PubMed[1] comprised more than 29 million citations for biomedical literature. Designing and implementing effective methods that automatically extract PPIs from huge amounts of biomedical literature have become a popular topic in biomedical text mining research [6].

The task of PPI extraction is to identify whether two proteins in a given fragment of text are considered to be interacted. Although the relationship between proteins may

The associate editor coordinating the review of this manuscript and approving it for publication was Nuno Garcia.

[1]https://www.ncbi.nlm.nih.gov/pubmed/

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

occur across several sentences, most studies have concentrated on discovering PPIs within a sentence [6], [7]. In our paper, we only consider extracting PPIs within a sentence. The simplest PPI extraction methods are the co-occurrence methods, which assume that two proteins interact whenever they simultaneously occur in a sentence [8]. Despite its simplicity and high recall, the precision of the co-occurrence methods is extremely low. For example, in the AIMed corpus, no more than 17% of all sentence-level protein pairs describe protein-protein interactions [6]. Pattern matching methods are another type of PPI extraction methods that can achieve a high degree of precision. Using hand-crafted patterns, Baumgartner *et al.* [9] achieved a precision of 38%, but only a recall of 6% on the PPI extraction task of BioCreative II. Yu *et al.* [10] built dependency graph patterns for PPI extraction that achieved higher precision but lower recall than other machine learning based methods.

Compared with the co-occurrence methods and pattern matching methods, machine learning based PPI extraction methods can simultaneously achieve higher precision and higher recall [6]. Traditional machine learning methods for PPI extraction include feature engineering based methods and kernel-based methods. Feature engineering based methods include two steps, building the features and training the classifier. For example, based on manually designed lexical, syntactic and dependency features, Saetre *et al.* [11] trained a support vector machine model for PPI extraction. However, manually building the features is a laborious process and these features may fail when the entity and relation annotation rules change [8].

Kernel-based methods utilize kernel function to map features into a latent high-dimensional separable space. The kernel functions are similarity functions in essence that help the classifier fully utilize structural similarity between instances. Most kernel functions for PPI extraction are based on syntactic parse trees [12]–[15], [20] or dependency parse trees [16]–[19] that consider sentence structure and semantic information. For example, Airola *et al.* [18] proposed the all-paths graph (APG) kernel that considers all dependency paths between two entity mentions because dependency paths are considered important indicators for the PPI extraction task. Murugesan *et al.* [20] proposed the Distributed Smoothed Tree kernel (DSTK), which exploits both syntactic and semantic space information. In addition to rely on parse tree information, shallow linguistic information is also useful for the PPI extraction. Giuliano *et al.* [21] proposed a kernel for PPI extraction leveraging shallow linguistic information such as chunking and parts-of-speech (POS). One specific kernel cannot fully model the semantic of sentences. To retrieve the most extensive information of a given sentence, Miwa *et al.* [22] combined several PPI extraction kernels by multiple kernel learning to build a new kernel function that outperformed previous kernel functions. However, kernel-based PPI extraction methods heavily rely on natural language processing (NLP) tools, and the error induced by these tools can cause the model's performance to decrease. Additionally, when the training dataset is very huge, maintaining a kernel matrix is a difficult task.

Deep learning has achieved remarkable success in the field of natural language process [23] and computer vision [24]. Compared with traditional machine learning methods, deep learning methods can automatically learn features from data. Many researchers recently attempted to apply deep learning methods to improve the performance of PPI extraction. Some state-of-the-art PPI extraction models are based on the recurrent neural network (RNN) [25]–[27]. Hsieh *et al.* [25] proposed using the LSTM, a variant of RNN, to extract PPIs from sentences. Yadav *et al.* [26] thought the shortest dependency path (SDP) between two entity mentions was more useful for PPI extraction than the whole sentence. They proposed the Att-sdpLSTM model that combined SDP and attention based multi-layer LSTM to extract PPIs from literature. Instead of using SDP, Ahmed *et al.* [27] used LSTM to model the input sentence's dependency tree structure. Although these RNN-based models achieved better performance, training RNN is notoriously difficult because of gradient vanishing and gradient exploding [28]. Due to sequence dependence, paralleling RNN is more difficult than convolutional neural network (CNN), and the prediction speed of RNN is generally slower than CNN with a similar size. Other researchers developed CNN-based PPI extraction models. For example, Quan *et al.* [29] proposed a multichannel convolutional neural network by fusing word embeddings of multiple versions for the PPI extraction task. Hua and Quan [30] proposed the sdpCNN model for PPI extraction by combining the SDP and CNN. In addition to sentence dependency information, Peng and Lu [31] proposed the McDepCNN model considering abundant lexical information such as parts-of-speech and chunking information. These CNN-based PPI extraction models are all single layer models that only model phrase-level information instead of sentence-level information in essence. To improve the CNN model's performance, previous work introduced additional linguistic features such as the dependency structure. However, parsing sentences into the dependency structure introduces extra computing and time complexity.

To improve CNN model's feature expression ability and avoid too much speed sacrifice, stacking several convolutional modules into one deep architecture is a feasible solution. However, when multiple convolution modules are directly stacked, the performance of the model will not be improved too much because the gradient vanishing will occur [32]. Johnson and Zhang *et al.* [33] discovered that the residual connection between convolutional modules was very helpful to train a deep convolutional neural network for text categorization. Inspired by their model, we propose a deep residual convolutional neural network model for PPI extraction. We do not directly stack single convolutional modules but stack residual convolutional blocks that contain several convolutional modules. There is one shortcut connection between the input and output of the residual

convolutional block. Our model downsamples sentence representation by half before putting the input into the residual convolutional block. The downsampling operation makes our model refines the sentence's representation gradually. Additionally, it makes the model's computation bounded to a constant as the model deepens. The combination of residual connection and downsampling operation makes our model achieve great performance improvement compared with previous shallow CNN-based models. Most of previous deep learning based models rely on other NLP tools. For example, Yadav *et al.*'s Att-sdpLSTM model relies on the Enju parser[2] that parses the input sentence into a predicate-argument dependency graph. The Enju parser only parses 3 sentences per second when we apply it to large corpora. When deploying the models, the time overhead is not tolerated. Our model does not rely on other NLP tools; thus, the practicality of our model is guaranteed. We evaluate our model on five benchmark PPI corpora, AIMed [34], BioInfer [35], HPRD50 [36], IEPA [37] and LLL [38]. Our model achieve better or comparable performance than previous models [25]–[27], [29]–[31].

The remainder of this paper is organized as follows. Section II lists some related work. Section III describes our proposed PPI extraction model. Section IV describes the experimental details and presents experimental results. Section V discusses our model and gives error analysis. Section VI concludes our work and gives future research directions.

II. RELATED WORK

A. DEEP LEARNING BASED BIOMEDICAL RELATION EXTRACTION MODEL

Most of the current deep learning based PPI extraction models are based on the RNN [25]–[27] and CNN [29]–[31], [56]. Zhao *et al.* [39] combined the deep feed-forward neural network and manually selected features to extract PPIs from sentences but the model did not present deep learning model's advantage compared with other RNN and CNN based models. The shortest dependency path (SDP) can be regarded as a simplified sentence, and some deep learning based biomedical relation extraction models rely on the SDP [26], [30], [40], [41]. However, the shortest dependency path can cause information loss for some sentences, especially when candidate entity pairs are located in the subordinate clause. The recursive neural network can fully use the parsed tree information and can also be used to extract biomedical relation [27], [42]. Compared with RNN and CNN, recursive neural network dose not obtain great improvement on performance, and its computation complexity is very high. Combining the RNN and CNN is another way to improve the performance of biomedical relation extraction [41], [43]. Dewi *et al.* [44] proposed to use deep CNN to extract drug-drug interactions but their model directly stacked several convolutional modules instead of using residual connection.

B. DEEP CNN FOR TEXT

Compared with single-layer CNN, RNN processes a sentence by the token sequence order and is more suitable to build text representation. RNN can achieve satisfactory performance in complex NLP tasks such as machine translation [23] and image caption [24]. However, parallel implementation of RNN is troublesome due to the sequence dependence of RNN. Compared with RNN, CNN owns a high computing efficiency and simple parallel implementation. Increasingly, more researchers use CNN to model these tasks. For example, Gehring *et al.* [46] designed a CNN-based sequence to sequence architecture for machine translation. The deep CNN was also applied to text classification [33], [47]. The common characteristics of these models are deeper architecture and residual connection compared with earlier CNN-based NLP models [48], [49]. Dilated convolution is another strategy to deepen the CNN. Strubell *et al.* [50] applied dilated convolution to the sequence tagging task. The deep CNN is widely applied to real-world systems. For example, the biomedical publication recommender system *Pubmender* uses multi-layer CNN to help researchers choose the publication venue [51].

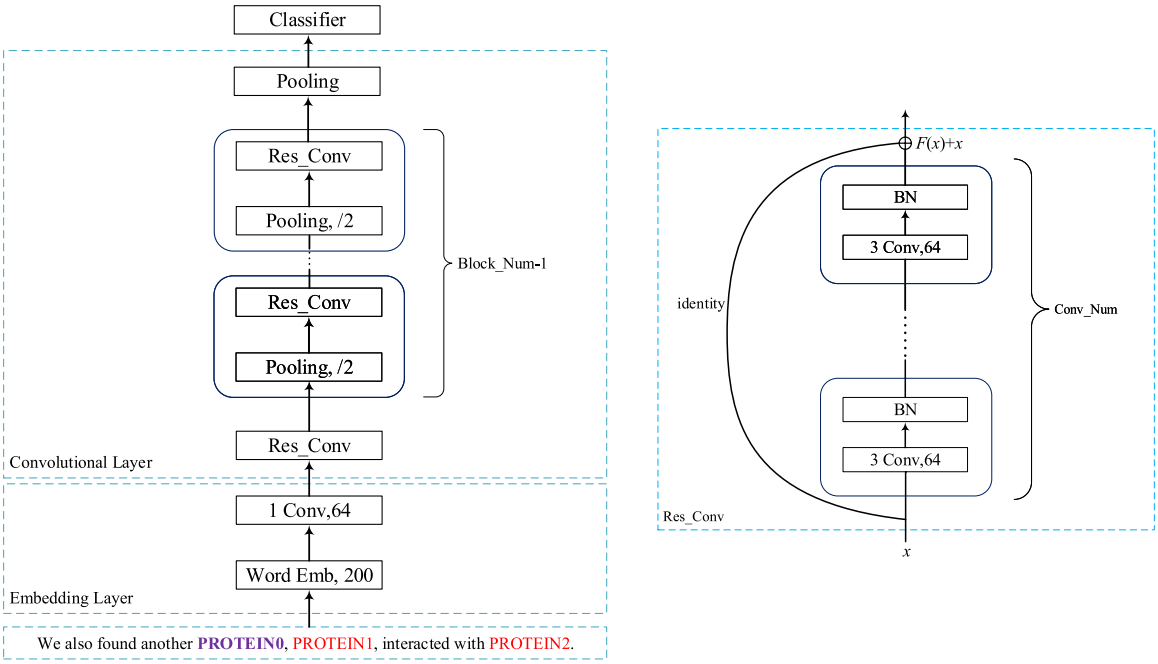
III. METHOD

A deeper architecture makes CNN have a larger receptive field. Residual connection guarantees gradient propagation stability. Combined deep CNN and residual strategy, we proposed a deep residual convolutional network for PPI extraction. As shown in Figure 1a, our model includes three parts, the embedding layer, the convolutional layer and the classifier layer. Our model leverages multi-layer CNN to extract meaningful features from the sentence with two marked protein entities. In the word embedding layer, our model transforms each word in the input sentence into word embedding. These word embeddings are fed into the convolutional layer. In the convolutional layer, we stack several residual convolutional blocks instead of convolutional modules. The residual convolutional block encapsulates several convolutional modules with a residual connection as shown in Figure 1b. Before each residual convolutional block, we apply a max-pooling layer with 2 strides that concentrates context information gradually and reduces the amount of calculation. After the convolutional layer, our model feeds the extracted sentence representation into the classifier layer and gives the prediction results. We describe our model in detail in the following subsections.

A. EMBEDDING LAYER

The word embedding layer transforms each word in the input sentence into a real value vector called word embedding. Word embeddings are low-dimensional dense vectors that capture words' syntactic and semantic information. For each sentence $S(w_1, w_2, \ldots, w_n)$, where n is the length of the

[2]http://www.nactem.ac.uk/enju/



(a) Our Model (b) Residual Convolutional Block

FIGURE 1. Architecture of our model. The left subfigure is the overall architecture of our model. The right subfigure is the residual convolutional block. 'Word Emb, 200' represents the word embedding layer with 200 dimensions. '1 conv, 64' represents the convolutional module with a filter size of 64 and a window size of 1. The 'Pooling, /2' represents one-dimensional max-pooling with a window size of 3 and a stride size of 2. 'Res_Conv' is the residual convolutional block. 'Block_Num' is the number of residual convolutional blocks. 'Pooling' represents one-dimensional global max-pooling. 'Classifier' represents the classifier layer. 'Conv_Num' is the number of convolutional modules in the residual convolutional block. 'BN' is the batch normalization operation.

sentence and w_i is the ith word of the sentence S, the word embedding matrix E of the sentence S is constructed by looking up the word embedding table W_{emb}. E is an $n * d$ matrix, and each column of matrix E represents the corresponding word's embedding in the input sentence. The word embedding matrix E is fed into next layer. All the words occurring in corpora are encoded into the word embedding table W_{emb}. Each column of the matrix W_{emb} represents a word's embedding in the vocabulary table. W_{emb} will be tuned when we train our models. In addition to words occurring in corpora, the vocabulary table also includes three special words, **PROTEIN0**, **PROTEIN1**, **PROTEIN2**. Before feeding sentence marked with entities into the word embedding layer, two marked proteins are substituted with **PROTEIN1** and **PROTEIN2** respectively. Other protein entities are substituted with **PROTEIN0**. **PROTEIN1** and **PROTEIN2** indicate candidate interaction's protein pairs in the input sentence. If we do not substitute two marked protein entities, our model cannot determine which pair of proteins in the sentence is interacting. Substituting the protein mentions with special words also makes our model learn the general pattern of corpora instead of paying excessive attention to proteins' literal meaning. Due to the diversity of biomedical entity mentions, using special protein words to replace them will reduce the size of the vocabulary table.

Pretrained word embeddings include rich semantic and syntactic information of words [57]. Collobert *et al.* [48] reported that initializing the word embedding layer by pretrained word embeddings is more beneficial to train model than randomly initializing the embedding layer. We initialize word embedding layer using pretrained word embeddings and tune parameters of the word embedding layer when we train our model.

The convolutional module with a window size of 1 is behind the word embedding layer. The aim of introducing the convolutional layer is to make the dimension of word embedding match following residual convolutional block.

B. RESIDUAL CONVOLUTIONAL BLOCK

In this section, we first introduce the convolutional module and then introduce how we stack several convolutional modules into a residual convolutional block.

The input of the convolutional module is $X = x_1 \oplus x_2 \oplus \cdots \oplus x_n$, where $\oplus$ is the concatenation operator. Compared with computer vision, the one-dimensional convolutional module is more widely used in the field of natural language processing. The one-dimensional convolution operator is applied to each window to produce a new feature. Concretely, assuming $x_{i:i+h-1}$ is the concatenation from the ith feature to the $(i+h-1)$th feature, where h is the window

size, we obtain new features as follows

$$c_i = f(w \cdot x_{i:i+h-1} + b) \tag{1}$$

where c_i is the new feature of the ith window, w is the weight of the convolution module, b is the bias, and f is the activation function. In this study, we adopt ReLU as our model's activation function. The feature map $c = c_1 \oplus c_2 \oplus \cdots \oplus c_{n-h+1}$ is produced after the one-dimensional convolutional module. The length of the output feature map, $n - h + 1$, is not equal to that of the input feature, n. To match the length of them, $\lfloor h/2 \rfloor$ zero vectors should be added into both sides of the input feature X.

When directly deepening the deep learning model, it may meet difficulty such as gradient vanishing or exploding problem. Residual learning, which connects low-level features and high-level features, can tackle the gradient propagation problem of deep learning models [52]. The residual convolutional block is inspired from the residual learning. As shown in Figure 1b, a sequence of convolutional modules are stacked and these convolutional modules are used to learn residual function. Assuming x is the input of the residual convolutional block and F is the stacked convolutional modules,

$$F(x) + x \tag{2}$$

is the output of the module. When stacking several residual convolutional blocks, gradient propagation from the high-level block to the low-level block will be easier because of the residual connection. Batch normalization can make the training process more stable and less sensitive to the learning rate [53]. We add batch normalization after each convolutional module. The number of convolutional modules 'Conv_Num' in each residual convolutional block is a hyper-parameter.

C. DOWNSAMPLING BY MAX-POOLING

Our model performs max-pooling operation with a window size of 3 and a stride of 2 before each residual convolutional block except the first one. The pooling layer produces new hidden representations by obtaining the componentwise maximum of 3 contiguous input vectors. The new hidden representations can cover more context than the original input features. Thus, subsequent residual convolutional block models the text in more abstract features. Performing a pooling operation every other triplet reduces the amount of calculation by half. Even if more layers are stacked, the amount of calculation of the model is bounded to a constant. We perform a global max-pooling operation after the last residual convolutional block. Thus, we can obtain a fixed dimensional feature vector.

D. CLASSIFIER AND LOSS FUNCTION

The output of global max-pooling is fed into the feed-forward classifier. Assuming the input of the feed-forward layer is h, the hidden representation of the feed-forward classifier is calculated as

$$m = f(W_f \cdot h + b_f), \tag{3}$$

where $W_f \in \mathbb{R}^{d_h \times d_i}$ is the weight matrix, $b_f \in \mathbb{R}^{d_h}$ is the bias, d_i is the dimension of the input, and d_h is the dimension of the hidden. Then, the hidden representation is fed into linear transformation to get each category's score as shown in (4).

$$s = W_s \cdot m + b_s \tag{4}$$

In (4), $W_s \in \mathbb{R}^{d_c \times d_h}$ is the weight matrix, $b_s \in \mathbb{R}^{d_c}$ is the bias, and d_c is the number of categories. We use the softmax function

$$p(label = class|S) = \frac{\exp(s_{class})}{\sum_{i=1}^{d_c} \exp(s_i)} \tag{5}$$

to normalize the score s into probability.

The loss function of our model is the cross entropy function. It is defined as

$$L = -\sum_{i=1}^{N} \log(p(label = class_i|S_i)), \tag{6}$$

where S_i is the ith sentence of the corpora, $class_i$ is the label of the ith sentence, and N is the number of sentences in corpora. We minimize the loss function by the Adam optimizer [54]. Dropout [55] is added after the word embedding layer and the penultimate layer of our model *i. e.* the feed-forward classifier's hidden layer. Additionally, the weight decay term is added to the loss function to prevent our model from overfitting.

IV. EXPERIMENTS

A. DATASETS

We evaluate our model using five benchmark PPI corpora, AIMed [34], BioInfer [35], HPRD50 [36], IEPA [37] and LLL [38]. Some differences exist in entity annotation and interaction annotation among these corpora. For example, BioInfer corpora include n-ary interactions that are not considered in other corpora. HPRD50 and LLL corpora define the types of interactions, but other corpora ignore them. Pyysalo *et al.* [8] transformed the five PPI corpora's annotations to a shared level of information and stored them in a unified format. The converted corpora are available on the Internet[3] and have been adopted in previous PPI extraction research [6], [25], [27].

All the converted corpora only annotate protein-protein interactions within a sentence. Interactions across several sentences are ignored. Moreover, these converted corpora ignore the interaction types and only annotate whether two entities within a sentence interact. Hence, the PPI extraction task is cast into a binary classification task. We assume annotated interacted protein pairs as positive instances and assume other unannotated protein pairs in the same sentence as negative instances. In a sentence with n protein entities ($n \geq 2$), $\binom{n}{2}$ instances will be generated. For example, in the sentence *"We also found another armadillo-protein, p0071, interacted with PS1."*, there are three protein entities which are *armadillo-protein*, *p0071* and *PS1*. As shown in Table 1,

[3]http://mars.cs.utu.fi/PPICorpora

TABLE 1. Example about candidate PPI instances.

Protein Pair	Candidate PPI instance	type
(armadillo-protein, p0071)	We also found another **PROTEIN1**, **PROTEIN2**, interacted with **PROTEIN0**.	Negative
(armadillo-protein, PS1)	We also found another **PROTEIN1**, **PROTEIN0**, interacted with **PROTEIN2**.	Negative
(p0071, PS1)	We also found another **PROTEIN0**, **PROTEIN1**, interacted with **PROTEIN2**.	Positive

the sentence generates $\binom{3}{2} = 3$ candidate PPI instances. The third instance is annotated as a positive instance in AIMed corpus, and the other two instances are negative instances. To ensure coverage of our model's vocabulary table, the two proteins of a candidate interaction are substituted by **PROTEIN1** and **PROTEIN2**, and other proteins in the sentence are substituted by **PROTEIN0**.

Although Pyysalo *et al.* [8] provided unified corpora for researchers, annotation differences of entities among corpora still exist. In AIMed corpus, there are some nested protein entities. For example, *"p75"* and *"p75 neurotrophin receptor"* are both annotated as protein entity mentions in the sentence *"...the p75 neurotrophin receptor and the p140trk (trkA) tyrosine kinase receptor..."*. When generating candidate PPI instances, we ignore instances between nested entities because no interactions exists in most of cases. If one of the nested entities occurs in a candidate instance, others will not be substituted by **PROTEIN0**. If nested entities do not participate in the interaction, we replace the longest one of nested entities with **PROTEIN0** and other mentions will be ignored. In BioInfer corpus, there are some composite named entity mentions and discontinuous entity mentions. For example, in the sentence *"Arp2/3 complex from Acanthamoeba binds profilin and cross-links actin filaments."*, *"Arp2/3"* is annotated as two entity mentions, *"Arp2"* and *"Arp3"*. We propose several rules to preprocess these composite and discontinuous entity mentions. These rules are listed as follows.

1) If composite entities have a common prefix, we concatenate the prefix and each component's suffix. For example, *"Arp2/3"* will be replaced by *"Arp2 / Arp3"*.
2) If composite entities have a common suffix, we concatenate the suffix and each component's prefix. For example, *"muscle and brain actin"* will be replaced by *"muscle actin and brain actin"*.
3) If an entity mention is discontinuous and does not intersect with other entity mentions, we make it continuous by including other text within the annotated range. For example, in the phrase *"Integrin (beta) chains"*, the parentheses are removed, and the *"Integrin beta chains"* is annotated as an entity mention. In our experiments, we consider *"Integrin (beta) chains"* as the entity mention.
4) If composite entities have a common prefix or suffix and intersect with other entity mentions, we directly drop these intersected entity mentions and process composite entities as before.

We preprocess the corpora by previous steps. The statistics of the preprocessed corpora are presented in Table 2.

TABLE 2. Corpora statistics.

Corpora	Sentences	Positive Instances	Negative Instances
AIMed	1955	999	4414
BioInfer	1100	2526	7050
HPRD50	145	163	270
IEPA	486	335	482
LLL	77	164	166

B. EVALUATION METRICS

The F1-score is the most common performance evaluation metric for the PPI extraction task [6]. The F1-score is calculated by (9).

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{9}$$

In (7) and (8), TP is the number of true positive instances, FP is the number of false positive instances and FN is the number of false negative instances. As shown in (9), the F1-score considers both precision calculated by (7) and recall calculated by (8). When precision is low and recall is high or precision is high and recall is low, the F1-score is low. A high F1-score means that most of the positive instances have been discovered and most of the extracted interactions are correct.

Because the original corpora are not divided into the training dataset and test dataset, we perform 10-fold cross-validation on the corpora following previous studies [6], [18] and then use average F1-score of 10-fold cross-validation experiments to evaluate our model's performance.

The macro F1-score, the average F1-score of positive instances' and negative instances' F1-scores, is adopted to evaluate performance by other previous studies [26], [27], [56]. It is meaningless to consider the negative instances' F1-score when evaluating a model's performance. Moreover, unbalanced corpora such as AIMed corpus can cause the model's macro F1-score is higher than normal F1-score due to the dominance of negative instances. To directly compare our model with these models adopting macro F1-score as the evaluation metric, we also evaluate our model using the macro F1-score.

C. PARAMETER SETTINGS

Chiu *et al.* [59] gave some suggestions about how to train word embeddings for biomedical literature using word2vec

TABLE 3. Hyper-parameter settings.

Hyper-parameter	Values
word embedding dimension	200
input dropout[a]	0.2
convolutional window	3
convolutional filter size	64
hidden layer dimension[b]	256
hidden dropout[c]	0.5
weight decay	0.00001
optimizer	Adam
learning rate	0.001
batch size	32
the number of epochs	100

[a]Word embedding layer's dropout rate
[b]Penultimate layer's output dimension
[c]Penultimate layer's dropout rate

and provided word embeddings[4] trained on the PMC corpus. We initialize our model's word embedding layer using the pretrained word embeddings. For out-of-vocabulary words such as special words **PROTEIN0/1/2**, we randomly initialize their word embeddings by sampling from the uniform distribution in $[-0.001, 0.001]$. The weights of convolutional layers and linear layers are initialized by the Xavier Uniform initializer [60], and their biases are initialized to zero. When the sentence length is less than 100, we pad it with a special pad token.

The hyper-parameter settings are shown in Table 3. For hyper-parameters not mentioned in the table, the number of residual blocks and the number of convolutional modules, we tune them by 10-fold cross-validation. We select the number of residual blocks from the range $[1, 6]$ and the number of convolutional modules from the range $[1, 4]$. For AIMed and BioInfer corpora, the number of residual blocks is 6 and the number of convolutional modules is 3. For HPRD50 corpus, the number of residual blocks is 4 and the number of convolutional modules is 2. For IEPA corpus, the number of residual blocks is 5 and the number of convolutional modules is 2. For LLL corpus, the number of residual blocks is 6 and the number of convolutional blocks is 2.

D. RESULTS

We compared our model with several kernel-based PPI extraction methods and deep learning based methods. The compared kernel-based models are briefly introduced as follows.

1) **Edit kernel**. The edit kernel obtains the similarity between two annotated sentences by computing the edit distance between them [17].
2) **APG kernel**. The APG kernel counts weighted shared dependency paths of all possible lengths. The dependency path weight is higher when the dependency path between entities is shorter [18].
3) **kBSPS kernel**. The kBSPS kernel is based on the shortest path between two annotated entities, dependency graph nodes and k distance dependency information [16].
4) **Hybrid kernel**. Miwa *et al.* [22] constructed the Hybrid kernel by combining several parsers and kernels.
5) **DSTK kernel**. The DSTK kernel exploits structural syntactic and phrase semantic information, which can be regarded as a compositional distributional semantic model [20].

The compared deep learning based methods are briefly introduced as follows.

1) **DNN**. Zhao *et al.* [39] pretrained feed-forward neural network on extracted features by autoencoders and then fine-tuned the pretrained feed-forward neural network on PPI corpora.
2) **MCCNN**. The MCCNN model is a convolutional neural network with a multichannel word embedding input layer [29].
3) **sdpCNN**. The sdpCNN model combines a convolutional neural network with the shortest dependency path between two marked protein entities [30].
4) **McDepCNN**. The McDepCNN is a convolutional neural network with abundant lexical and syntactic input features [31].
5) **LSTM**. Hsieh *et al.* [25] combined LSTM and the multilayer fully connected neural network to extract PPIs from the literature.
6) **DCNN**. The DCNN model combined the convolutional neural network with feature embeddings such as WordNet embeddings to extract PPIs [56].
7) **Att-sdpLSTM**. The Att-sdpLSTM model is an attention based multilayer LSTM model whose input is the shortest dependency path parsed by the Enju parser [26].
8) **tLSTM**. The tLSTm model comprising of structured attention based architecture and tree-LSTM [27]. The tree-LSTM is build on dependency parse tree.

The experimental results are presented in Table 4. Our model achieved better performance than all kernel-based PPI extraction methods in the AIMed and BioInfer corpora according to the precision, recall and F1-score. Compared with the DSTK kernel, the state-of-the-art kernel method, our model achieved 6.9% and 10.3% F1-score improvement in the AIMed and BioInfer corpora, respectively. Our model also shows performance advantages over other CNN-based PPI extraction models, MCCNN, sdpCNN and McDepCNN, in the AIMed and BioInfer corpora. The three models all rely on external resources. The sdpCNN and McPepCNN rely on linguistic tools to enhance input features of CNN. The MCCNN relies on several pretrained word embedding resources. Compared with these models, our model improves CNN's feature extraction ability by deepening the architecture instead of introducing more linguistic knowledge. Although the DNN model is a deep learning based model, it is a feature engineering based model in essence. Hence, the DNN model

[4]https://github.com/cambridgeltl/BioNLP-2016

TABLE 4. Evaluation results by precision, recall and F1-score.

Model	AIMed			BioInfer			HPRD50			IEPA			LLL		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Edit kernel[a] [17]	68.8	27.7	39.0	50.4	39.2	43.8	71.3	45.2	53.3	77.2	60.2	67.1	68.0	98.0	78.4
APG kernel[a] [18]	59.9	53.6	56.2	60.2	61.3	60.7	68.2	69.8	67.8	66.6	82.6	73.1	71.3	91.0	78.1
kBSPS kernel[a] [16]	50.1	41.4	44.6	49.9	61.8	55.1	62.2	87.1	71.0	58.8	89.7	70.5	69.3	93.2	78.1
Hybrid kernel [22]	55.0	68.8	60.8	65.7	71.1	68.1	68.5	76.1	70.9	67.5	78.6	71.7	77.6	86.0	80.1
DSTK kernel [20]	68.9	73.2	71.0	75.7	76.9	76.3	76.3	84.2	80.0	75.9	85.2	80.2	87.3	91.2	89.2
DNN [39]	51.5	63.4	56.1	53.9	72.9	61.6	58.7	92.4	71.3	71.8	79.4	74.22	76.0	91.0	81.4
MCCNN [29]	76.4	69.0	72.4	81.3	78.1	79.6	-	-	-	-	-	-	-	-	-
sdpCNN [30]	64.8	67.8	66.0	73.4	77.0	75.2	-	-	-	-	-	-	-	-	-
McDepCNN [31]	67.3	60.1	63.5	62.7	68.2	65.3	-	-	-	-	-	-	-	-	-
LSTM [25]	78.8	75.2	76.9	87.0	87.4	87.2	-	-	-	-	-	-	-	-	-
Our model	79.0	76.8	77.6	87.4	86.5	86.9	74.9	82.8	77.7	71.6	80.6	75.5	80.5	87.2	83.2

[a] The Edit, APG, kBSPS kernels' experimental results are extracted from [6].

TABLE 5. Evaluation results by macro precision, macro recall and macro F1-score.

Model	AIMed			BioInfer			HPRD50			IEPA			LLL		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
DCNN[a] [56]	88.6	81.7	85.0	72.1	77.5	74.7	-	-	-	-	-	-	-	-	-
Att-sdpLSTM [26]	92.6	93.9	93.3	80.8	82.6	81.7	79.9	77.6	78.7	76.9	75.6	76.3	84.2	83.6	83.9
tLSTM [27]	81.4	81.9	81.6	88.9	89.3	89.1	81.7	82.3	81.3	78.6	78.7	78.5	84.8	84.3	84.2
Our model	87.0	85.9	86.3	91.5	90.8	91.1	82.6	82.2	81.7	78.4	78.5	78.3	83.2	82.7	82.6

[a] The DCNN's experimental results are extracted from [26].

achieved similar or worse performance compared than kernel-based models. Compared with the LSTM model, our model achieved better performance in AIMed corpus.

Almost all deep learning based PPI extraction models do not present experimental results for the HPRD50, IEPA and LLL corpora because these three corpora are too small. For example, there are only 164 positive instances in the LLL corpus, as shown in Table 2. To comprehensively analyze our model's performance, we present experimental results in the three corpora. Our model still outperforms most of kernel-based PPI extraction despite the small size of corpora. The DSTK kernel achieved better performance in the three corpora. A plethora of external lexical and syntactic knowledge is introduced into the DSTK kernel. When the data are very scarce, the knowledge might make DSTK achieve better performance.

The DCNN model, Att-sdpLSTM model and tLSTM model are evaluated by the macro F1-score. We also evaluated our model using the macro F1-score to directly compare our model with the three models. The macro F1-score experimental results are presented in Table 5. Our model achieved better performance in the AIMed and BioInfer corpora than the DCNN model and the tLSTM model. Although the Att-sdpLSTM model achieves better performance in the AIMed corpus, our model outperforms it in almost all other corpora. Moreover, the parse speed of the Enju parser, which relied by the Att-sdpLSTM model, is very slow. The time-consuming parser makes the Att-sdpLSTM model lack of practicality compared with our model. The tLSTM model achieved slightly better performance in the IEPA and LLL corpora, likely due to the introduction of the dependency tree structure.

TABLE 6. Ablation experimental results.

Settings	AIMed		BioInfer	
	F1	Δ[a]	F1	Δ
Our Model	77.6	-	86.9	-
w/o residual connection	74.4	-3.2	84.6	-2.3
w/o 2-stride max-pooling	75.2	-2.4	84.8	-2.1
random word embedding	74.4	-3.2	85.1	-1.8

[a] Δ is the difference between each setting and our model.

[b] w/o represents removing some component from our model.

V. DISCUSSIONS

A. ABLATION STUDY

To investigate the effectiveness of each component of our model, we perform an ablation study on the AIMed and BioInfer corpora, the results of which are shown in Table 6. In the study, we analyze the effectiveness of residual connection, 2-stride max-pooling and pretrained word embeddings respectively. We delete a part of our model and reserve other settings. Next, we perform 10-fold cross-validation on the AIMed and BioInfer corpora. After we remove the residual connection in the residual convolutional blocks, our model is

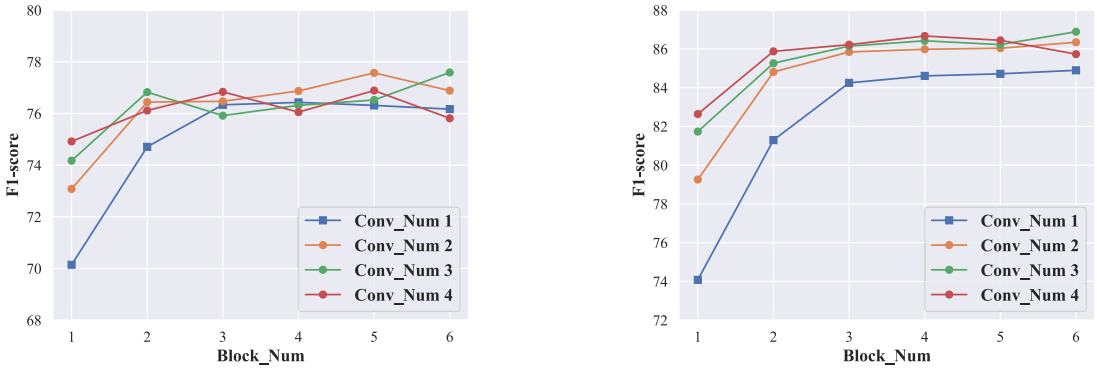


FIGURE 2. Effect of varying Block_Num and Conv_Num on the F1-score.

similar to the ordinary multilayer convolutional neural network. Compared with the other two modifications, removing residual connection causes the largest performance decline, e.g. 4.1% in AIMed and 2.6% in BioInfer. This shows that the residual connection is an important component of our model. The 2-stride max-pooling operation before each residual convolutional block enlarges the coverage of context, and brings performance gain. Initializing word embedding layer using pretrained word embeddings is an important step when training deep learning based NLP models. Our ablation experiments also verify the strategy. Additionally, the effectiveness of initializing word embeddings is more obvious for the AIMed corpus than the BioInfer corpus. As shown in Table 2, there are more instances in the BioInfer corpus than the AIMed corpus. We suppose that the effect of initializing word embeddings diminishes with data increasing.

B. HYPER-PARAMETERS ANALYSIS

The number of residual blocks and the number of convolutional modules are two hyper-parameters of our model. We determine their values in the hyper-parameters selection phase. To examine how the two hyper-parameters affect the performance of our model, we perform experiments on the AIMed and BioInfer corpora. For the number of residual blocks, we select it from the range [1, 6]. For the number of convolutional modules, we select it from the range [1, 4].

As shown in Figure 2, the F1-score increases with the number of blocks, illustrating the effect of deepening the model's architecture. There is a large F1-score improvement when we increase the number of blocks from 1 to 2. However, the improvement is not very obvious when the number of blocks is greater than 3. We suppose that the captured context information by our model is limited when the number of blocks is 1 and the context is enlarged by stacking several residual blocks. Moreover, max-pooling with a stride size of 2 also plays an important role in expanding the context range. However, the residual, which supplements missing information from the first few layers, gradually decreases in the last few layers. Hence, when the prediction speed is an important factor of the PPI extraction system, 2 or 3 residual blocks are the best choices. When better accuracy is needed, 5 or 6 residual blocks are the best choices. When the number of convolutional modules is set to 1, our model achieved the worst performance. When the number of convolutional modules is set to 4, our model's performance may degrade. Thus, residual convolutional blocks with 2 or 3 convolutional modules are the best choices for our model.

C. ERROR ANALYSIS

In this section, we analyze our model's error predictions on the AIMed, BioInfer, HPRD50, IEPA and LLL corpora and summarize them as follows.

1) Indication words such as *bind* and *link* between two entities interfere with our model to make a correct judgement. For example, no interaction relation between the marked entities exists in the sentence *"This suggests that **PROTEIN1** may link **PROTIEIN2** activation to molecules that regulate GTP binding proteins."*, but our model predicts the relation is positive. The word *link* located between ***PROTEIN1*** and ***PROTEIN2*** is confusing because the objective of *link* is easy to be mistaken for the word ***PROTEIN2***. However, the objective of *link* is the word *activation*.
2) Some sentences are incorrectly annotated. For example, the sentence *"Expression of **PROTEIN2** was generally weak and did not correlate with the expression of either **PROTEIN0** or **PROTEIN1**."* is annotated to be a positive instance. However, the sentence indicates that the expression of **PROTEIN2** does not correlate with the expression of ***PROTEIN1***; thus, we think the instance is negative.
3) The absence of entity mentions makes our model lose the literal information due to the substitution of entity mentions. For example, in the sentence *"These results suggest that **PROTEIN1** can confer transcriptional regulation and possibly cell cycle control and tumor suppression through an interaction with **PROTEIN0**, in particular with **PROTEIN2**."*, the mention of

PROTEIN0 is *TFIID* and the mention of ***PROTEIN2*** is *TAFII250*. Our model makes the wrong prediction because it does not know protein *TAFII250* is a type of protein *TFIID*.

4) The complex sentence structure makes our model produce the wrong prediction. For example, our model does not detect the interaction between ***PROTEIN1*** and ***PROTEIN2*** in the sentence *"Overexpression of* ***PROTEIN0*** *promotes degradation of* ***PROTEIN1*** *in a pVHL-dependent manner that requires the ATPase domain of* ***PROTEIN2****"*.
5) The ambiguous expression of sentences may induce our model to make the wrong prediction. For example, our model predicts the candidate instance *"Coexpression of* ***PROTEIN1*** *with* ***PROTEIN2*** *in the baculovirus expression system resulted in the phosphorylation of* ***PROTEIN0*** *on tyrosine residues."* to be positive. Although there is a coexpression relation between ***PROTEIN1*** and ***PROTEIN2***, it is not sufficient to describe the presence of the interaction between two entities[5].

VI. CONCLUSION AND FUTURE WORK

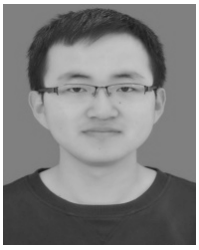
In the paper, we propose a residual convolutional neural network based PPI extraction model. Our model stacks more convolutional modules by residual connection. Compared with previous models, our model achieves better or comparable performance. Our model alleviates the use of traditional biomedical natural language processing tools such as the dependency parser, which can cause the accumulation of errors. The ablation experiments verify the necessity of residual connection and 2-stride max-pooling. Additionally, we provide advice concerning how to choose hyperparameters for our model. To further improve PPI extraction model's performance, many labeled sentences are required. Distant supervision can provide large amounts of annotated sentences by aligning the PPI knowledge base with the biomedical literature. Distant supervision based deep learning model for PPI extraction is our future research direction. Additionally, sequence-based PPI prediction models can discover new PPIs from proteins' amino acid sequences. Enriching existing PPI networks by combining our model and other sequence-based PPI prediction models [2] is another research direction.

REFERENCES

[1] J.-F. Rual *et al.*, "Towards a proteome-scale map of the human protein–protein interaction network," *Nature*, vol. 437, no. 7062, pp. 1173–1178, Sep. 2005.
[2] T. Sun, B. Zhou, L. Lai, and J. Pei, "Sequence-based prediction of protein protein interaction using a deep-learning algorithm," *BMC Bioinf.*, vol. 18, no. 1, p. 277, May 2017.
[3] U. Stelzl *et al.*, "A human protein-protein interaction network: A resource for annotating the proteome," *Cell*, vol. 122, no. 6, pp. 957–968, Sep. 2005.
[4] D. Kwon, J. H. Yoon, S.-Y. Shin, T.-H. Jang, H.-G. Kim, I. So, J.-H. Jeon, and H. H. Park, "A comprehensive manually curated protein–protein interaction database for the death domain superfamily," *Nucleic Acids Res.*, vol. 40, no. D1, pp. D331–D336, Jan. 2012.
[5] H. Ge, L. Sun, and J. Yu, "Fast batch searching for protein homology based on compression and clustering," *BMC Bioinf.*, vol. 18, no. 1, p. 508, Nov. 2017.
[6] D. Tikk, P. Thomas, P. Palaga, J. Hakenberg, and U. Leser, "A comprehensive benchmark of kernel methods to extract protein–protein interactions from literature," *PLoS Comput. Biol.*, vol. 6, no. 7, Jul. 2010, Art. no. e1000837.
[7] D. Tikk, I. Solt, P. Thomas, and U. Leser, "A detailed error analysis of 13 kernel methods for protein-protein interaction extraction," *BMC Bioinf.*, vol. 14, no. 1, p. 14, Jan. 2013.
[8] S. Pyysalo, A. Airola, J. Heimonen, J. Björne, F. Ginter, and T. Salakoski, "Comparative analysis of five protein-protein interaction corpora," *BMC Bioinf.*, vol. 9, p. S3, Apr. 2008.
[9] W. A. Baumgartner, Jr., Z. Lu, H. L. Johnson, J. G. Caporaso, J. Paquette, A. Lindemann, E. K. White, O. Medvedeva, K. B. Cohen, and L. Hunter, "Concept recognition for extracting protein interaction relations from biomedical text," *Genome Biol.*, vol. 9, p. S9, Sep. 2008.
[10] K. Yu, P.-Y. Lung, T. Zhao, P. Zhao, Y.-Y. Tseng, and J. Zhang, "Automatic extraction of protein-protein interactions using grammatical relationship graph," *BMC Med. Inform. Decis. Making*, vol. 18, no. S2, p. 42, Jul. 2018.
[11] R. Saetre, K. Sagae, and J. Tsujii, "Syntactic features for protein-protein interaction extraction," in *Proc. 2nd Int. Symp. Lang. Biol. Med. (LBM)*, Singapore, 2008, pp. 6.1–6.14.
[12] A. J. Smola and S. V. N. Vishwanathan, "Fast kernels for string and tree matching," in *Proc. Adv. Neural Inf. Process. Syst.*, 2003, pp. 585–592.
[13] M. Collins and N. Duffy, "Convolution kernels for natural language," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 625–632.
[14] A. Moschitti, "Efficient convolution kernels for dependency and constituent syntactic trees," in *Machine Learning—ECML*, vol. 4212. Berlin, Germany: Springer, 2006, pp. 318–329.
[15] T. Kuboyama, K. Hirata, H. Kashima, K. F. Aoki-Kinoshita, and H. Yasuda, "A spectrum tree kernel," *Trans. Jpn. Soc. Artif. Intell.*, vol. 22, no. 2, pp. 140–147, 2007.
[16] P. Palaga, "Extracting relations from biomedical texts using syntactic information," M.S. thesis, Fac. I-Humanities, Technische Univ. Berlin, Berlin, Germany, 2009.
[17] G. Erkan, A. Özgür, and D. R. Radev, "Semi-supervised classification for extracting protein interaction sentences using dependency parsing," in *Proc. Joint Conf. Empirical Methods Natural Lang. Process. Comput. Natural Lang. Learn. (EMNLP-CoNLL)*, Prague, Czech Republic, 2007, pp. 228–237.
[18] A. Airola, S. Pyysalo, J. Björne, T. Pahikkala, F. Ginter, and T. Salakoski, "All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning," *BMC Bioinf.*, vol. 9, p. S11, Nov. 2008.
[19] S. Kim, J. Yoon, and J. Yang, "Kernel approaches for genic interaction extraction," *Bioinformatics*, vol. 24, no. 1, pp. 118–126, Jan. 2008.
[20] G. Murugesan, S. Abdulkadhar, and J. Natarajan, "Distributed smoothed tree kernel for protein-protein interaction extraction from the biomedical literature," *PLoS ONE*, vol. 12, no. 11, Nov. 2017, Art. no. e0187379.
[21] C. Giuliano, A. Lavelli, and L. Romano, "Exploiting shallow linguistic information for relation extraction from biomedical literature," in *Proc. 11th Conf. Eur. Chapter Assoc. Comput. Linguistics*, Trento, Italy, 2006, pp. 3–7.
[22] M. Miwa, R. Sætre, Y. Miyao, and J. Tsujii, "Protein–protein interaction extraction by leveraging multiple kernels and parsers," *Int. J. Med. Inform.*, vol. 78, no. 12, pp. e39–e46, Dec. 2009.
[23] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," Sep. 2014, *arXiv:1409.0473*. [Online]. Available: https://arxiv.org/abs/1409.0473
[24] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. 32nd Int. Conf. Mach. Learn.*, Lille, France, vol. 37, 2015, pp. 2048–2057.
[25] Y.-L. Hsieh, Y.-C. Chang, N.-W. Chang, and W.-L. Hsu, "Identifying protein-protein interactions in biomedical literature using recurrent neural networks with long short-term memory," in *Proc. 8th Int. Joint Conf. Natural Lang. Process.*, Taipei, Taiwan, vol. 2, 2017, pp. 240–245.
[26] S. Yadav, A. Ekbal, S. Saha, A. Kumar, and P. Bhattacharyya, "Feature assisted stacked attentive shortest dependency path based Bi-LSTM model for protein–protein interaction," *Knowl.-Based Syst.*, vol. 166, pp. 18–29, Feb. 2019.

[5]https://en.wikipedia.org/wiki/Gene_co-expression_network

[27] M. Ahmed, J. Islam, M. R. Samee, and R. E. Mercer, "Identifying protein-protein interaction using tree LSTM and structured attention," in *Proc. IEEE 13th Int. Conf. Semantic Comput. (ICSC)*, Newport Beach, CA, USA, Jan./Feb. 2019, pp. 224–231.
[28] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. 30th Int. Conf. Mach. Learn.*, Atlanta, GA, USA, vol. 28, Jun. 2013, pp. 1310–1318.
[29] C. Quan, L. Hua, X. Sun, and W. Bai, "Multichannel convolutional neural network for biological relation extraction," *BioMed Res. Int.*, vol. 2016, Nov. 2016, Art. no. 1850404.
[30] L. Hua and C. Quan, "A shortest dependency path based convolutional neural network for protein-protein relation extraction," *BioMed Res. Int.*, vol. 2016, Jun. 2016, Art. no. 8479587.
[31] Y. Peng and Z. Lu, "Deep learning for extracting protein-protein interactions from biomedical literature," in *Proc. BioNLP*, Vancouver, BC, Canada, 2017, pp. 29–38.
[32] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 2818–2826.
[33] R. Johnson and T. Zhang, "Deep pyramid convolutional neural networks for text categorization," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, Vancouver, BC, Canada, vol. 1, 2017, pp. 562–570.
[34] R. Bunescu, R. Ge, R. J. Kate, E. M. Marcotte, R. J. Mooney, A. K. Ramani, and Y. W. Wong, "Comparative experiments on learning information extractors for proteins and their interactions," *Artif. Intell. Med.*, vol. 33, no. 2, pp. 139–155, Feb. 2005.
[35] S. Pyysalo, F. Ginter, J. Heimonen, J. Björne, J. Boberg, J. Järvinen, and T. Salakoski, "BioInfer: A corpus for information extraction in the biomedical domain," *BMC Bioinf.*, vol. 8, no. 1, p. 50, Feb. 2007.
[36] K. Fundel, R. Küffner, and R. Zimmer, "RelEx—Relation extraction using dependency parse trees," *Bioinformatics*, vol. 23, no. 3, pp. 365–371, Feb. 2007.
[37] J. Ding, D. Berleant, D. Nettleton, and E. Wurtele, "Mining MEDLINE: Abstracts, sentences, or phrases?" in *Proc. Pacific Symp. Biocomput.*, 2002, pp. 326–337.
[38] C. Nédellec, "Learning language in logic-genic interaction extraction challenge," in *Proc. Learn. Lang. Logic Workshop*, 2005, pp. 1–7.
[39] Z. Zhao, Z. Yang, H. Lin, J. Wang, and S. Gao, "A protein-protein interaction extraction approach based on deep neural network," *Int. J. Data Mining Bioinf.*, vol. 15, no. 2, p. 145, 2016.
[40] Z. Zhao, Z. Yang, L. Luo, H. Lin, and J. Wang, "Drug drug interaction extraction from biomedical literature using syntax convolutional neural network," *Bioinformatics*, vol. 32, no. 22, pp. 3444–3453, Nov. 2016.
[41] Z. Li, Z. Yang, C. Shen, J. Xu, Y. Zhang, and H. Xu, "Integrating shortest dependency path and sentence sequence into a deep learning framework for relation extraction in clinical text," *BMC Med. Inform. Decis. Making*, vol. 19, no. S1, p. 22, Jan. 2019.
[42] S. Lim, K. Lee, and J. Kang, "Drug drug interaction extraction from the literature using a recursive neural network," *PLoS ONE*, vol. 13, no. 1, Jan. 2018, Art. no. e0190926.
[43] X. Sun, K. Dong, L. Ma, R. Sutcliffe, F. He, S. Chen, and J. Feng, "Drug-drug interaction extraction via recurrent hybrid convolutional neural networks with an improved focal loss," *Entropy*, vol. 21, no. 1, p. 37, Jan. 2019.
[44] I. N. Dewi, S. Dong, and J. Hu, "Drug-drug interaction relation extraction with deep convolutional neural networks," in *Proc. IEEE Int. Conf. Bioinf. Biomed. (BIBM)*, Kansas City, MO, USA, Nov. 2017, pp. 1795–1802.
[45] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70. Sydney, NSW, Australia: International Convention Centre, Aug. 2017, pp. 933–941.
[46] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70. Sydney, NSW, Australia: International Convention Centre, Aug. 2017, pp. 1243–1252.
[47] A. Conneau, H. Schwenk, Y. Le Cun, and L. Barrault, "Very deep convolutional networks for text classification," in *Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguistics*, Valencia, Spain, vol. 1, 2017, pp. 1107–1116.
[48] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," Mar. 2011, *arXiv:1103.0398*. [Online]. Available: https://arxiv.org/abs/1103.0398
[49] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Doha, Qatar, 2014, pp. 1746–1751.
[50] E. Strubell, P. Verga, D. Belanger, and A. McCallum, "Fast and accurate entity recognition with iterated dilated convolutions," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Copenhagen, Denmark, 2017, pp. 2670–2680.
[51] X. Feng, H. Zhang, Y. Ren, P. Shang, Y. Zhu, Y. Liang, R. Guan, and D. Xu, "The deep learning–based recommender system 'Pubmender' for choosing a biomedical publication venue: Development and validation study," *J. Med. Internet Res.*, vol. 21, no. 5, May 2019, Art. no. e12957.
[52] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.
[53] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, Lille, France, vol. 37, Jul. 2015, pp. 448–456.
[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Dec. 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980
[55] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[56] S.-P. Choi, "Extraction of protein–protein interactions (PPIs) from the literature by deep convolutional neural networks with various feature embeddings," *J. Inf. Sci.*, vol. 44, no. 1, pp. 60–73, Feb. 2018.
[57] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," Jan. 2013, *arXiv:1301.3781*. [Online]. Available: https://arxiv.org/abs/1301.3781
[58] D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao, "Relation classification via convolutional deep neural network," in *Proc. COLING 25th Int. Conf. Comput. Linguistics*, Dublin, Ireland, 2014, pp. 2335–2344.
[59] B. Chiu, G. Crichton, A. Korhonen, and S. Pyysalo, "How to train good word embeddings for biomedical NLP," in *Proc. 15th Workshop Biomed. Natural Lang. Process.*, Berlin, Germany, 2016, pp. 166–174.
[60] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *J. Mach. Learn. Res.*, vol. 9, pp. 249–256, May 2010.



HAO ZHANG received the B.Sc. degree in computer science and technology from Jilin University, Changchun, China, in 2017, where he is currently pursuing the master's degree with the College of Computer Science and Technology under the supervision of Prof. R. Guan. His major research interests include information extraction and machine learning.

RENCHU GUAN (M'12) received the Ph.D. degree from Jilin University, Changchun, China, in 2010, where he is currently an Associate Professor with the College of Computer Science and Technology. He has published over 40 papers. His research was featured in the IEEE TKDE, the IEEE TGRS, the IEEE J-STARS, and so on. His research interests include machine learning, bioinformatics, and knowledge engineering. He was a recipient of several grants from NSFC.

FENGFENG ZHOU (M'12–SM'13) received the B.Sc. and Ph.D. degrees in computer science from the University of Science and Technology of China, in 2000 and 2005, respectively. He is currently a Full Professor of health informatics with the College of Computer Science and Technology, Jilin University, Changchun, China. His laboratory focuses on the data fusion and multivariate biomarker selection algorithms for the heterogeneous health big data, including bio-OMICs, biomedical imaging, physiological signal, biochemical screening, and electronic health record. He was awarded with the Hundred Talent Program by the Chinese Academy of Sciences and the Tang Aoqing Professorship by Jilin University.

YANCHUN LIANG received the Ph.D. degree in applied mathematics from Jilin University, Changchun, China, in 1997, where he is currently a Professor with the College of Computer Science and Technology. He was a Visiting Scholar with The University of Manchester, U.K., from 1990 to 1991, a Visiting Professor with the National University of Singapore, from 2000 to 2001, a Guest Professor with the Institute of High Performance Computing of Singapore, from 2002 to 2004, a Guest Professor with Trento University, Italy, from 2006 to 2010, and a Guest Professor with the University of Missouri, USA, from 2011 to 2017. He has published over 400 papers. His research was featured in *Bioinformatics*, the IEEE TSMC, the IEEE TKDE, *Journal of Micromechanics and Microengineering*, *Physical Review E*, *Neural Computing and Applications*, *Smart Materials and Structures*, *Artificial Intelligence in Medicine*, *Applied Artificial Intelligence*, and so on. His research interests include computational intelligence, machine learning methods, text mining, MEMS modeling, and bioinformatics. He was a recipient of several grants from NSFC, EU, and so on.

ZHI-HUI ZHAN (M'13–SM'18) received the bachelor's and Ph.D. degrees from the Department of Computer Science, Sun Yat-sen University, Guangzhou, China, in 2007 and 2013, respectively.

He is currently the Changjiang Scholar Young Professor and the Pearl River Scholar Young Professor with the School of Computer Science and Engineering, South China University of Technology, China. He has published over 100 research papers in international journals and conference proceedings. His current research interests include evolutionary computation algorithms, swarm intelligence algorithms, and their applications in real-world problems, and in environments of cloud computing and big data.

Dr. Zhan's doctoral dissertation was awarded the China Computer Federation (CCF) Outstanding Dissertation and the IEEE Computational Intelligence Society (CIS) Outstanding Dissertation. He received the Outstanding Youth Science Foundation from the National Natural Science Foundation of China (NSFC), in 2018, the Wu Wen Jun Artificial Intelligence Excellent Youth from the Chinese Association for Artificial Intelligence, in 2017, and the First Grade Award in Natural Science from Guangdong Province, in 2018. He is listed as one of the most cited Chinese researchers in computer science. He is also an Associate Editor of *Neurocomputing*.

LAN HUANG received the Ph.D. degree from the College of Computer Science and Technology, Jilin University, Changchun, China, in 2003, where she is currently a Professor and a Supervisor for Ph.D. candidates. She is mainly involved in business intelligence theory and application research. She was invited to Italy Trento University as a Senior Visitor, in 2010. As a PI and a Co-PI, she has been undertaking or accomplished more than ten teaching and scientific research projects, granted by the National 863 Hi-tech Research and Development Program, the National Science Foundation China, provincial/ministerial foundations, and other sources. The software results researched and developed by her team have brought good economic benefit for the cooperative enterprises and application enterprises. She has published 64 academic papers and obtained 14 software copyrights. In recent years, her research interests focus on business intelligence application and social network mining algorithm. The works that she participated as a Main Investigator were awarded the Jilin Province Scientific and Technological Progress Award (First Prize Winner), in 2017, and the National Commercial Science and Technology Award by the China General Chamber of Commerce (the First Prize Winner), in 2014 and 2010. She was one of the outstanding youth project funding winners of Jilin Province, in 2005, and the person in charge of Young and Middle-aged Leader and Innovation Team of Jilin Province, in 2012.

XIAOYUE FENG received the B.Sc. and Ph.D. degrees in computer science from Jilin University, Changchun, China, in 2003 and 2008, respectively, where she is currently a Lecturer with the College of Computer Science and Technology. Her main research interests include bioinformatics, machine learning, and text mining.