# A Hybrid Deep Learning Model for Protein-Protein Interactions Extraction from Biomedical Literature

*applied sciences*

Article

# A Hybrid Deep Learning Model for Protein–Protein Interactions Extraction from Biomedical Literature

**Changqin Quan [1,*], Zhiwei Luo [1] and Song Wang [2]**

[1] Graduate School of System Informatics, Kobe University, 1-1, Rokkodai-cho, Nada-ku, Kobe 657-8501, Japan; luo@gold.kobe-u.ac.jp
[2] School of Elec Eng, Comp and Math Sci; Curtin University, Kent St, Bentley WA 6102, Australia; Song.Wang@curtin.edu.au
* Correspondence: quanchqin@gold.kobe-u.ac.jp

Received: 11 March 2020; Accepted: 9 April 2020; Published: 13 April 2020

**Abstract:** The exponentially increasing size of biomedical literature and the limited ability of manual curators to discover protein–protein interactions (PPIs) in text has led to delays in keeping PPI databases updated with the current findings. The state-of-the-art text mining methods for PPI extraction are primarily based on deep learning (DL) models, and the performance of a DL-based method is mainly affected by the architecture of DL models and the feature embedding methods. In this study, we compared different architectures of DL models, including convolutional neural networks (CNN), long short-term memory (LSTM), and hybrid models, and proposed a hybrid architecture of a bidirectional LSTM+CNN model for PPI extraction. Pretrained word embedding and shortest dependency path (SDP) embedding are fed into a two-embedding channel model, such that the model is able to model long-distance contextual information and can capture the local features and structure information effectively. The experimental results showed that the proposed model is superior to the non-hybrid DL models, and the hybrid CNN+Bidirectional LSTM model works well for PPI extraction. The visualization and comparison of the hidden features learned by different DL models further confirmed the effectiveness of the proposed model.

**Keywords:** protein–protein interactions; deep learning (DL); convolutional neural networks (CNN); bidirectional long short-term memory (bidirectional LSTM)

## 1. Introduction

Protein–protein interactions (PPIs) play important roles in various biological processes and are of pivotal importance in the regulation of biological systems, and are consequently implicated in the development of disease states [1]. The exponentially increasing size of biomedical literature and the limited ability of manual curators to discover PPIs in text has led to delays in keeping PPI databases, such as BIND (The Biomolecular Interaction Network Database) [2], MINT (The Molecular INTeraction Database) [3], and The IntAct molecular interaction database [4], updated with the current findings. Consequently, this causes a bottleneck when leveraging the valuable information that is currently available in order to develop personalized health care solutions.

Previous studies have explored different methods for PPI extraction. The dominant techniques generally fall under the following three broad categories: co-occurrence-based methods [5], rule-and-pattern-based methods [6,7], and statistical machine learning (ML)-based methods [8–12]. Co-occurrence- based methods measure the correlation between each pair of entities by co-occurrence. A major weakness of these methods is their tendency for having a high recall, but a low precision. This is mainly because when entities do not appear in pairs in the training set, no co-occurrence with correlation can be recorded. Rule-and-pattern-based methods employ predefined patterns and rules to match the labelled sequence.

Although traditional rule-and-pattern-based methods have achieved high accuracy, their sophistication in pattern design and attenuated recall performance make them unsuitable for practical usage. Compared with co-occurrence- and rule–pattern-based methods, ML-based methods show a much better performance and generalization. In particular, the recent surge of interest in deep learning (DL) methods is due to the fact that they have been shown to outperform previous state-of-the-art techniques for PPI extraction tasks.

Generally, ML based approaches cast the problem of PPIs extraction into a classification problem. Suppose to extract the binary PPIs between entity $e_1$ and entity $e_2$ in a given sentence $= w_1 w_2 \ldots e_1 \ldots e_2 \ldots w_n$, where $w_i$ is a word in $S$. The classification model is constructed to output 1 when $e_1$ and $e_2$ are related, otherwise it is 0. The inputs of the classification model are the features extracted from $S$. A key difference between traditional ML and DL is in how features are extracted and represented. Traditional ML-based methods usually collect words around target entities as features, such as unigram, bigram, and some semantic and syntactic features, and then these features are put into a bag-of-words model and encoded into one-hot type representations. However, such representations are unable to capture semantic relations among words or phrases and fail in generalizing the long context dependency [13]. The former issue is rendered as "vocabulary gap" (e.g., the words "depend" and "rely" are different in one-hot representations, albeit their similar linguistic functions). The latter one is introduced because of the n-order Markov restriction that attempts to alleviate the issue of "curse of dimensionality." Moreover, the inability to extract features automatically leads to laborious manual efforts in designing features.

Different from one-hot encoding representation, word distribution representation is proposed to solve the "vocabulary gap" problem by mapping words to dense vectors of real numbers in a low-dimensional space [14,15]. In addition to using words as features, some semantic and syntactic features, such as part-of-speech (POS), word position, and dependency path between two entities, can also be embedded in their distribution representations for feature learning.

Recent studies have proposed several feature embedding methods combining DL models for PPI extraction. Most of these studies focused on finding effective linguistic features for embedding or on tuning model hyperparameters for a certain framework of DL (e.g., convolutional neural networks (CNN) and long short-term memory (LSTM)). Different from previous work, this study focuses on exploring hybrid deep learning architecture for PPI extraction tasks. Our contributions can be summarized as follows:

(1) we propose a hybrid architecture of a bidirectional LSTM+CNN model for PPI extraction. The proposed model is able to solve the main issue of the inability to model long-distance contextual information with CNN for PPI extraction in a long sentence. Furthermore, CNN is applied to encode the important information contained in the bidirectional LSTM networks, and to extract the local features and structure information effectively.

(2) two embeddings (word-embeddings and shortest dependency path (SDP) embedding) are novelty applied as the inputs of for the bidirectional LSTM networks, which are able to capture semantical and syntactical features effectively.

(3) we investigate the hidden features learned by different DL models by reducing the feature dimensions and visualizing these features, which can help to compare the classification performance of these hidden features trained by different DL models in an approximate way.

The outline of the paper is as follows: Section 2 discusses related works. The framework of the model is introduced in Section 3. The experimental study is shown in Section 4. Section 5 is the discussion and Section 6 is the conclusion.

## 2. Related Work

Recent natural language processing (NLP) research is now increasingly focusing on the use of new DL architectures and algorithms to solve difficult NLP tasks, including PPI extraction from biomedical literature. This section reviews significant DL-related models and methods that have been employed for PPI extraction tasks.

The convolutional neural network (CNN) was originally developed for image recognition tasks [16–18] and has been successfully applied to the NLP domain by exploiting distributed representations for words (word embeddings). Collobert and Weston [19] made the first work to show the utility of pre-trained word embeddings. They proposed a neural network architecture that forms the foundation of many current approaches. The work also established word embeddings as a useful tool for PPI extraction tasks.

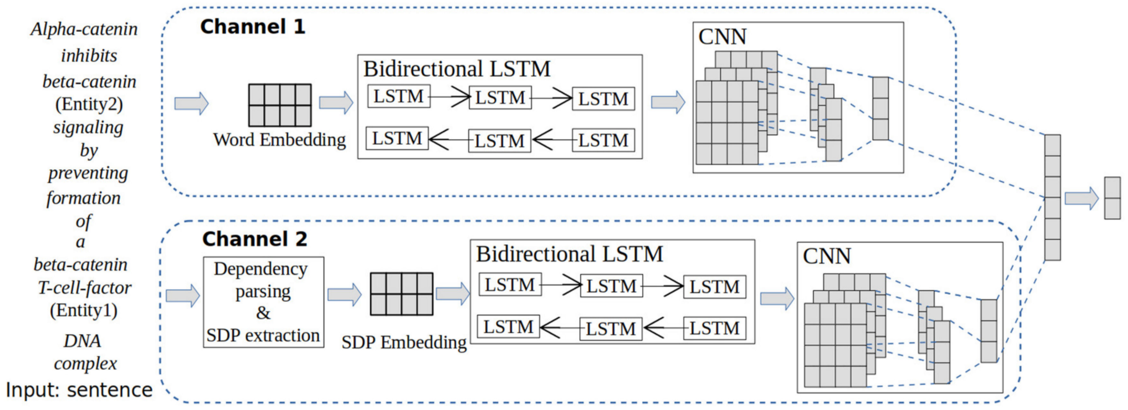
Like many other natural language tasks, using CNN along with word embeddings has been shown to be effective in PPI extraction tasks [20], as they have the ability to extract salient n-gram features from the input sentence in order to create an informative latent semantic representation of the sentence for downstream tasks [21–23]. Hua and Quan [24] proposed a CNN with pre-trained word vectors of the shortest dependency path between entity pairs for PPI extraction. In the literature [25], Quan extended their work to propose a multichannel convolutional neural network (MCCNN) that enables the fusion of multiple pre-trained word embeddings from various data sources (PubMed, PMC (PubMed Central), MedLine, and Wikipedia), and consequently expands the coverage of input vocabulary. A similar work includes the multichannel dependency-based convolutional neural network model (McDepCNN) [26], which combines CNN along with more syntactic features (e.g., part-of-speech, chunk, named entity, dependency, and position feature) for input embedding. CNN has been shown to outperform previous state-of-the-art techniques for PPI extraction. However, the main issue with CNNs is their inability to model long-distance contextual information and to preserve sequential order in their representations [22,27].

Comparatively, the recurrent neural network (RNN) model [28] is able to model long-distance contextual information by memorizing over previous computations and utilizing this information in current processing. However, simple RNN-based methods suffer from the vanishing gradient problem, which makes it hard to learn and tune the parameters of the earlier layers in the network. This limitation was overcome by various networks, such as long short-term memory (LSTM) [29]. In addition, Lai et al. [30] proposed the bidirectional recurrent neural network (bidirectional RNN) to capture the semantic information in different directions for sentence modeling. LSTM has also recently been applied for PPI extraction [31,32]. These studies yield a comparable performance, but much more attention has been paid to selecting the classification features or tuning model hyperparameters.

Hybrid models that combine CNN with RNN have aroused some interest in the DL domain, and several different combinations of architecture have been proposed for different subjects [33]. In the NLP domain, Zhou et al. [34] proposed C-LSTM, which combines CNN with LSTM for text classification. It takes advantage of CNN to extract a sequence of higher-level phrase representations and feeds them into LSTM to represent sentences. In the literature [35], a CNN–bidirectional gated recurrent unit (BiGRU) model, integrating CNN and bidirectional gated recurrent unit (BiGRU), is proposed for sentence semantic classification. This model utilizes CNN to obtain intermediate representations, and utilizes BiGRU to obtain contextual representations and semantic distribution. Inspired by these works, this study explores hybrid deep learning architecture for PPI extraction tasks.

## 3. The Model Description

The architecture of the proposed hybrid deep learning model is shown in Figure 1. By employing bidirectional LSTM to extract the semantic information in both the preceding and succeeding contexts, the proposed model is able to solve the main issue of the inability to model long-distance contextual information with CNN for PPI extraction in a long sentence. Furthermore, CNN is applied to encode the important information contained in the bidirectional LSTM networks, and to capture the local features and structure information effectively. As shown in Figure 1, two embeddings (word-embeddings and SDP embedding) are novelty applied as the input of bidirectional LSTM networks, which is able to capture semantical and syntactical features effectively.



Alpha-catenin inhibits beta-catenin (Entity2) signaling by preventing formation of a beta-catenin T-cell-factor (Entity1) DNA complex
Input: sentence
Channel 1
Word Embedding
Bidirectional LSTM
LSTM
CNN
Channel 2
Dependency parsing & SDP extraction
SDP Embedding
Bidirectional LSTM
LSTM
CNN

**Figure 1.** Architecture of the hybrid deep learning model. The input of the model is a sentence which is through two embeddings (word-embeddings and shortest dependency path (SDP) embedding). The two embeddings are separately fed to a bidirectional long short-term memory (LSTM) network followed by a convolutional neural networks (CNN) network, as two separate channels. The outputs of the CNN networks of the two channels are concatenated together for classification.

## *3.1. Model Input*

In the expression of PPIs, most of the interaction words are verbs and nouns, and thus dependency parsing is particularly well-suited for relation extraction because the dependency grammar (DG) views the verb as the structural center of all of the clause structure. Dependency parsing has been well used for PPI extraction in previous studies [25,26,33]. Given an input sentence, we extract the SDP (shortest dependency path) by dependency parsing. Figure 2 shows the dependency parsing graph (Stanford parser [36] and a visualization tool [37] are utilized).

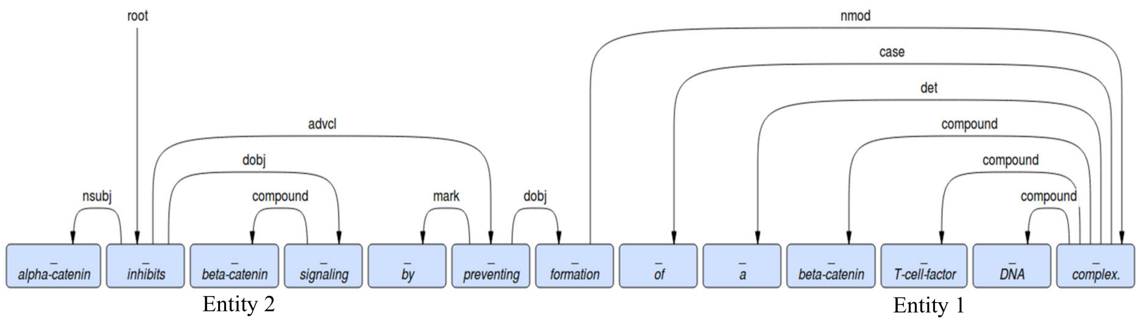


**Figure 2.** The dependency parsing graph for the sentence "alpha-catenin inhibits beta-catenin (Entity 2) signaling by preventing the formation of a beta-catenin T-cell-factor (Entity 1) DNA complex" (dependency relations are shown by the arrow lines). The shortest dependency path extracted between the target entity pairs is "T-cell-factor (Entity 1) compound complex nmod formation dobj preventing advcl inhibits dobj signaling compound beta-catenin (Entity 2)".

The extracted SDPs are treated as texts for the embedding. Embedding is a feature learning process where words from the vocabulary are mapped to vectors of real numbers in a low-dimensional space relative to the vocabulary size. For notation, we use $D \in R^{|V| \times d}$ to represent the pretrained embedding, where $V$ is the vocabulary and d is the embedding dimension. Specifically, for word-embedding, $V$ is composed of the words from the input sentences and for SDP embedding, $V$ is composed by the symbols from the input SDPs.

When we assign each word (or symbol) in an input sentence (or SDP) with a corresponding row vector from $D$, we would get a matrix representation $P \in R^{N \times d}$ for an input sentence (or SDP). The word-embedding and SDP embedding are two c × N × d tensors, where $c$ is the input size of the channel, $N$ is the max length of input sentences, and $d$ is the embedding dimension.

The input of the model is a sentence that goes through two embedding channels, namely: (1) pretrained word embedding trained on the PubMed abstract corpus [38] and (2) SDP embedding trained on the shortest dependency paths extracted between target entity pairs.

## *3.2. Intermediate Structure*

As shown in Figure 1, the two channel embeddings each separately pass through a bidirectional LSTM network consequent with a CNN network. bidirectional LSTM enables learning long-term dependencies for both the preceding and succeeding contexts. After that, CNN encodes the important information contained in the bidirectional LSTM networks.

The bidirectional LSTM network is a combination of Bidirectional RNNs with LSTM, using an input word embedding sequence $(e_1, e_2, \ldots, e_N)$, where $N$ denotes the max length of input sequences. As illustrated in Figure 3, bidirectional RNNs compute the forward hidden sequence $\overrightarrow{h}$, the backward hidden sequence $\overleftarrow{h}$, and the output sequence $y$ by iterating the forward layer from $t = (1, \ldots, N)$, the backward layer from $t = (N, \ldots, 1)$ and then updating the output layer as follows:

$$\overrightarrow{h}_t = S\left(W_{e\overrightarrow{h}} e_t + W_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t-1} e_t + b_{\overrightarrow{h}}\right) \tag{1}$$

$$\overleftarrow{h}_t = S\left(W_{e\overleftarrow{h}} e_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t+1} e_t + b_{\overleftarrow{h}}\right) \tag{2}$$

$$y_t = W_{\overrightarrow{h}y} \overrightarrow{h}_t + W_{\overleftarrow{h}y} \overleftarrow{h}_t + b_y \tag{3}$$

where $W$ denotes weight matrices, $b$ denotes bias vectors, and $S$ is the hidden layer function on each element of a vector.

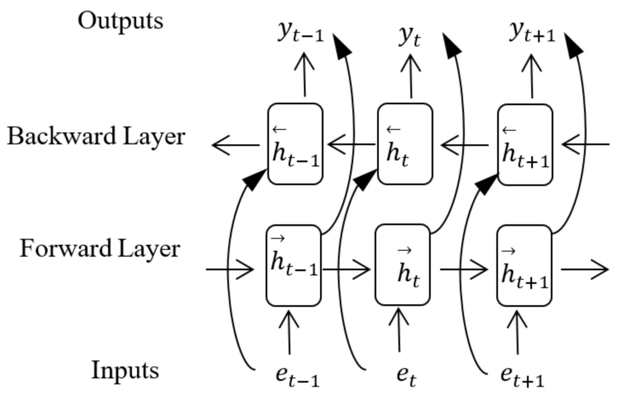


**Figure 3.** Illustration of the bidirectional recurrent neural network.

In the bidirectional LSTM network, each unit of RNN is an LSTM (Figure 4):

$$f_t = \sigma\left(W_{ef} e_t + W_{hf} h_{t-1} + W_{cf} c_{t-1} + b_f\right) \tag{4}$$

$$i_t = \sigma(W_{ei} e_t + W_{hi} h_{t-1} + W_{ci} c_{t-1} + b_i) \tag{5}$$

$$o_t = \sigma(W_{eo} e_t + W_{ho} h_{t-1} + W_{co} c_t + b_o) \tag{6}$$

$$c_t = f_t c_{t-1} + i_t \tan\mathrm{h}(W_{ec} e_t + W_{hc} h_{t-1} + b_c) \tag{7}$$

$$h_t = o_t \tan\mathrm{h}(c_t) \tag{8}$$

where $\sigma$ is the logistic sigmoid function, and $f_t$, $i_t$, $o_t$, $c_t$ are the forget gate, input gate, output gate, and cell state, respectively at the time step $t$.

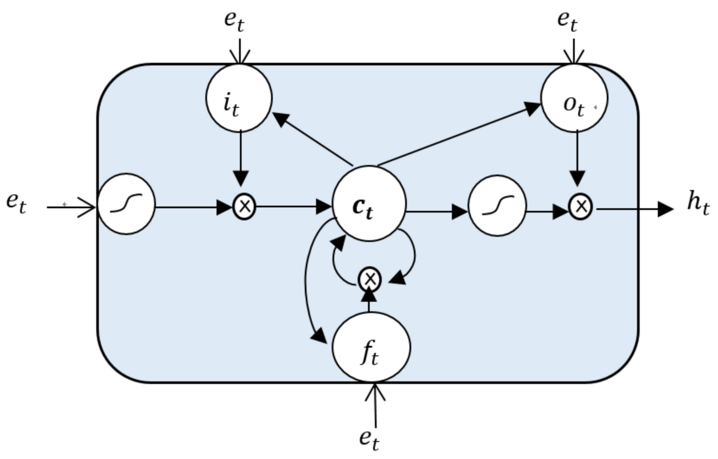


**Figure 4.** Illustration of the long short-term memory (LSTM) cell.

The bidirectional LSTM network outputs for both channels are each separately fed to CNN networks. Given an output sequence $y_i$ of the bidirectional LSTM network, it is input into the convolutional layer. The convolutional layer contains a set of filters for different window sizes $g$, and computes the output feature map $C$ as follows:

$$C = \left[m_1,\ m_2,\ \ldots, m_{N-g+1}\right] \tag{9}$$

$$m_i = f\left(W_i y_{i:i+g-1} + b_i\right) \tag{10}$$

where $f$ is an activation function, $b$ is a bias term, and  is element-wise multiplication.

Then the concatenation operation is applied to join multiple outputs from the Max-Pooling layer into a single tensor for each channel. Finally, the CNN network outputs for both channels are concatenated to a unique vector and fed to a fully connected layer.

## 4. Datasets and Experimental Setup

### *4.1. Datasets and Preprocessing*

Two benchmarking corpora, Aimed [39] and BioInfer [40], were used for the experiments and evaluation. The Aimed dataset was manually tagged, which includes 1955 sentences, and was considered as the standard dataset for the PPI extraction tasks. BioInfer was developed by the Turku BioNLP group [41], and contains 1100 sentences.

To ensure the generalization of features, we use a similar data preprocessing method to the authors of [9], by replacing two target entities with special symbols "Entity 1" and "Entity 2", separately, and entities that are not target entities are all represented as "Entity". Text preprocessing includes dependency parsing and shortest dependency paths (SDPs) extraction. If there is an interaction between the two entities, we consider this instance as a positive one; otherwise, we consider it as a negative one. Table 1 shows the statistics for the PPIs corpora.

**Table 1.** Statistics for protein–protein interactions (PPIs) corpora.

| | **Sentence Num.** | **Positive Num.** | **Negative Num.** |
|---|---|---|---|
| BioInfer | 1100 | 2534 | 7132 |
| AIMed | 1955 | 1000 | 4834 |

### *4.2. The Experiments*

#### 4.2.1. Pretrained Embeddings

There are two pretrained embeddings used in our model, namely: word embedding and SDP embedding. The pretrained word embedding is trained on PMC and PubMed (Pyysalo et al. [38]), with a vocabulary size of 4,087,446 words. The SDP embedding is trained on the shortest dependency paths extracted between the target entity pairs from PPIs corpora by Gensim Word2Vec tool (CBOW training algorithm, Radim Řehůřek and Petr Sojka, Brno, Czech Republic) [42]. The vocabulary size of the SDP embedding on the BioInfer corpus and Aimed corpus is 1840 words and 1349 words, respectively. Corresponding to the word and SDP embeddings, two types of input, sentences and the shortest dependency paths (SDPs) between target entity pairs, are fed into the model as inputs, separately.

#### 4.2.2. Parameter Setting

The parameter setting is summarized in Table 2.

**Table 2.** Parameter setting.

| Parameter | Aimed | BioInfer |
|---|---|---|
| Max length of sentences | 133 | 95 |
| Max length of SDPs | 31 | 35 |
| Optimization algorithm | Adam | |
| Learning rate | $1 \times 10^{-4}$ | |
| Number of epochs | 100 | |
| Training batch size | 128 | |
| **Bidirectional LSTM network** | | |
| Number of LSTM units | 100 | |
| Dropout rate | 0.2 | |
| Recurrent dropout rate | 0.2 | |
| **CNN network** | | |
| Window sizes | (3, 4, and 5) | |
| Num. of filters on each window size | 100 | |
| Dropout rate after each max pooling layer | 0.5 | |
| Activation function in convolution layer | Relu | |

#### 4.2.3. Implementation and Evaluation Metrics

Keras 2.2.4 (https://keras.io/) is applied to implement the models. The configurations of the machine are as follows: GPU—Quadro M1200/PCIe/SSE2, Nvidia, Santa Clara, CA, USA; CPU—Intel®Core™ i7-7820HQ CPU @ 2.90GHz × 8 Intel, Colorado Springs, CO, USA; System—Ubuntu 18.04.2 LTS 64-bit Memory, 16 GiB, Canonical Ltd., London, UK.

We use the average F1 macro score to evaluate the performances of the DL models using the 10-cross-validation (10-fold CV) method. In the calculation of each fold, we calculate the F1 macro score on the entire testing data by creating a Callback function at the end of each epoch, instead of a batch-wise average value.

### *4.3. Performance Comparison*

The performance of the proposed DL model is compared with the state-of-the-art non-hybrid models and hybrid models. Table 3 shows the comparison results.

CNN-based models [24–26] and LSTM-based models [31,32] have been applied for PPI extraction recently. However, there are many differences in the text preprocessing strategies (e.g., protein entities are generalized with special symbols or protein IDs, utilizing different tokenization and parsing tools, and utilizing filtering rules or not) and other experimental setups (e.g., hyperparameter settings and evaluation metrics). These make it difficult to compare the performance of the DL models directly. In our experiments, we compare the non-hybrid models (1–5 in Table 3) and hybrid models (6–7 in Table 3) with the same experimental settings, except for the models and the inputs of the models.

**Table 3.** Comparison of the F1 macro score for deep learning (DL) models on benchmark datasets.

| | Approach | BioInfer | Aimed |
|---|---|---|---|
| | Non-hybrid | | |
| (1) | CNN+word embedding | 70.4 | 68.8 |
| (2) | CNN+word embedding+SDPs embedding | 71.7 | 69.3 |
| (3) | LSTM+word embedding | 72.2 | 71.6 |
| (4) | LSTM+word embedding+SDPs embedding | 73.0 | 71.9 |
| (5) | Bidirectional LSTM+word embedding+ SDPs embedding | 73.4 | 72.4 |
| | Hybrid | | |
| (6) | CNN+Bidirectional LSTM+word embedding+ SDPs embedding | 73.3 | 70.0 |
| (7) | Bidirectional LSTM+CNN+word embedding+ SDPs embedding (the proposed) | 74.4 | 73.7 |

By comparing CNN-based models 1 with 2, and LSTM-based models 3 with 4 in Table 3, it can be found that the two embeddings (word-embedding and SDP embedding) input is able to improve the performance for both CNN- and LSTM-based models. This result demonstrates the effectiveness of integrating word embedding and SDP embedding for PPI extraction. The interaction verbs (e.g., "affect" and "bind") and dependency relation symbols (e.g., "nsubj" and "dobj") in the SDPs could provide useful information for classifying target protein pairs and extracting the relations.

It is also observed that the LSTM model outperforms the CNN model in the two datasets. The bidirectional LSTM achieved the best performance among the non-hybrid models.

As a hybrid model, we compare the proposed bidirectional LSTM+CNN model (7 in Table 3) with the hybrid CNN+bidirectional LSTM model (6 in Table 3), which is a variant of C-LSTM proposed by the authors of [34]. C-LSTM learns the sentence representation by combining CNN and LSTM, and has been shown to be superior to CNN and LSTM for some text classification tasks. The results show that the proposed bidirectional LSTM+CNN model outperforms the CNN+bidirectional LSTM model greatly for PPI extraction in both of the datasets.

## *4.4. Hidden Features Visualization and Comparison*

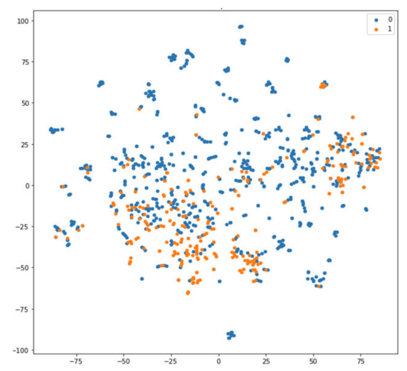
We further investigate the hidden features learned by different DL models by reducing the feature dimensions and visualizing these features. We choose four DL models (2, 4, 6, and 7 in Table 3, represent CNN, LSTM, and the two hybrid-based DL models, respectively) for the visualization.

We first split the dataset into two parts randomly, as follows: training set 90% and testing set 10%. Using the same training set, we train the DL models separately. Then, we create a truncated model for each DL model. The truncated model keep the same network layers until the last hidden feature layer. Then, we set the weights for it from the trained model. The truncated model is used to predict the features for the testing data. After that, we apply principal component analysis (PCA) [43] to reduce the features predicted by the truncated model to a lower dimension. The PCA variance is 1.00, which implies that the reduced dimensions do represent the hidden features well (scale 0 to 1). Then, t-Distributed Stochastic Neighbor Embedding (t-SNE) [44] is applied for the visualization.

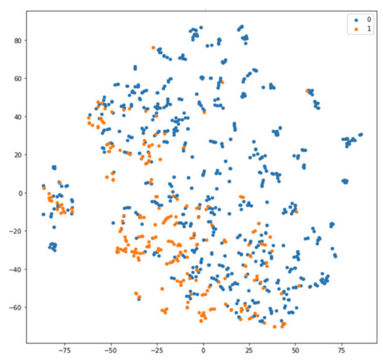
Figure 5 illustrates the scatter plots of the hidden features on the last layer of the DL models (CNN, LSTM, CNN+bidirectional LSTM, and Bidirectional LSTM+CNN) after dimensionality reduction.

In Figure 5, the comparison of the four DL models (CNN, LSTM, CNN+bidirectional LSTM, and bidirectional LSTM+CNN) on both datasets shows that bidirectional LSTM+CNN has a much better classification performance than the other models.

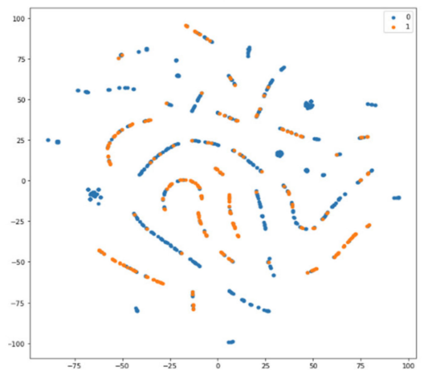
Figure 6 illustrates the F1-score increasing trends of the four DL models as the number of epochs increase. The datasets are split into two parts randomly, as follows: training set 90% and testing set 10%.



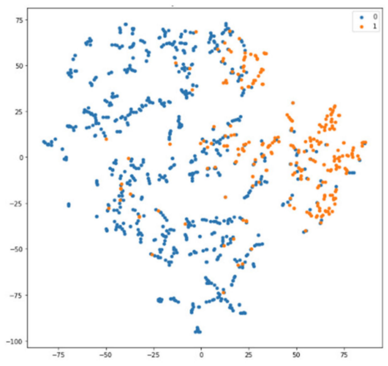
(**a**)



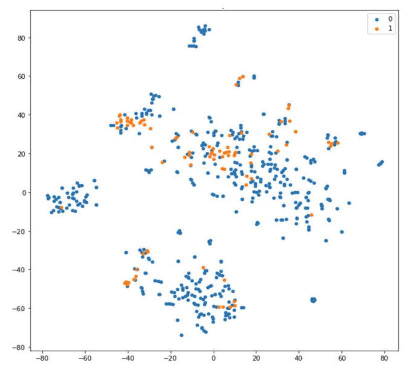
(**b**)



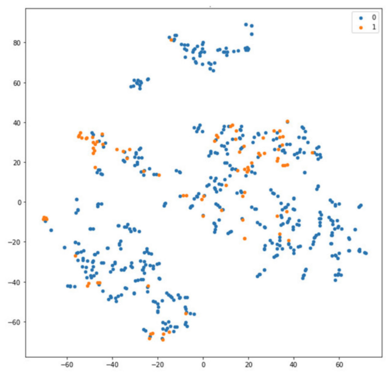
(**c**)



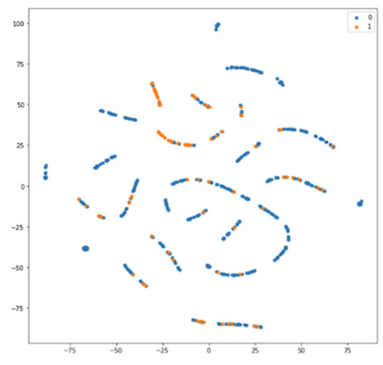
(**d**)



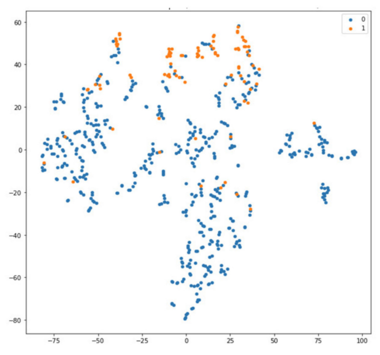
(**e**)



(**f**)

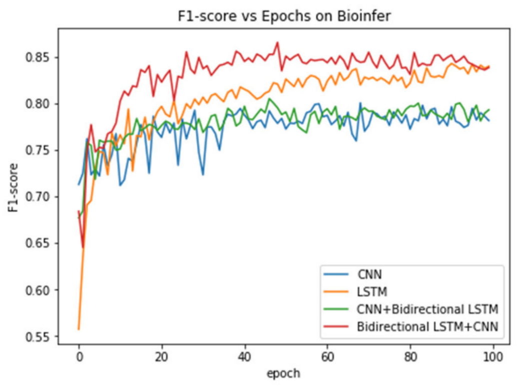


(**g**)

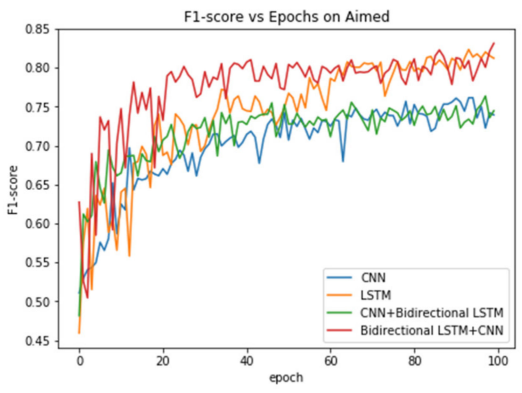


(**h**)

**Figure 5.** The scatter plots of the hidden features in the last layer of DL models (CNN, LSTM, CNN+bidirectional LSTM, and bidirectional LSTM+CNN) after dimensionality reduction in the Bioinfer and Aimed datasets. (**a**) CNN on Bioinfer; (**b**) LSTM on Bioinfer; (**c**) CNN+bidirectional LSTM on Bioinfer; (**d**) Bidirectional LSTM+CNN on Bioinfer; (**e**) CNN on Aimed; (**f**) LSTM on Aimed; (**g**) CNN+bidirectional LSTM on Aimed; (**h**) Bidirectional LSTM+CNN on Aimed



(**a**)



(**b**)

**Figure 6.** The F1-score increasing trends of the DL models as the number of epochs increase, (**a**) F1-score vs Epochs on Bioinfer; (**b**) F1-score vs Epochs on Aimed

As can be seen from Figure 6, bidirectional LSTM+CNN produces a high F1-score with less epochs than the other models. It also keeps a high classification performance as the number of epochs increases. We also found that LSTM also could obtain a high F1-score on some points, but it needs more training than the proposed bidirectional LSTM+CNN model.

## 5. Discussion

Recently, DL models have been shown to be superior to traditional ML models in many NLP tasks. In PPI extraction tasks, the performance of a DL-based method is mainly affected by the architecture of the DL models and the feature embedding methods. In this study, we compared different architectures of DL models, including CNN, LSTM, and hybrid models, and found that the proposed bidirectional LSTM+CNN model is superior to the other DL models for PPI extraction. As more complex network architectures (e.g., a deep hybrid model with more layers or a deep reinforcement learning model) have not been considered in this study, there is still space for further improvement of the DL model architecture.

In addition to the DL model architecture, the feature embedding methods also have an effect on the classification performance. In DL models, the input semantic and syntactic features are embedded in their distribution representations for feature learning. Previous studies have experimented with some different feature embeddings (such as ngram, part-of-speech (POS), word position, and dependency path). In this study, the stacked pretrained word embedding and SDP embedding are fed into a two embedding channel (word embedding and SDP embedding) architecture, such that the model can extract the feature information more accurately. With more feature channels, an improved performance can be expected.

The difference in the text preprocessing strategies (e.g., protein entities are generalized with special symbols or protein IDs, utilizing different tokenization and parsing tools, and utilizing filtering rules or not) and other experimental settings (e.g., hyperparameter settings and evaluation metrics) make it difficult to compare the performance of the DL models directly. This study did not directly compare the results with other related studies. Instead, under the same experimental environment, our experiments covered all of the DL models that have been applied in previous studies [24–26,31,32]. It would be more objective to compare the performance of different DL models directly. The experiments of this study used two standard PPI datasets, and a more robust DL model can be expected when using large-scale training data.

DL models are generally opaque, meaning that although they can produce accurate predictions, it is not clear how or why a given prediction is made. This study investigated the hidden features learned by different DL models by reducing feature dimensions and visualizing these features. This would help us to compare the classification performance of these hidden features trained by different

DL models in an approximate way. However, how to connect an input with the hidden features and how to control them during training stage are still challenging problems.

## 6. Conclusions

In this paper, a hybrid DL model is proposed for PPI extraction tasks. The model innovatively integrates bidirectional LSTM with CNN in a two-embedding channel (word embedding and SDP embedding) architecture.

CNN with word embeddings has been shown to be effective in PPI extraction tasks, as it has the ability to extract salient n-gram features from the input sentence to create an informative latent semantic representation of the sentence. However, a main issue with CNNs is their inability to model long-distance contextual information for PPI extraction in a long sentence. To solve this problem, we employ bidirectional LSTM to extract the semantic information in both the preceding and succeeding contexts, because the architecture of LSTM is able to model long-distance contextual information by memorizing over previous computations and utilizing this information in current processing. Furthermore, CNN is applied to encode the important information contained in the bidirectional LSTM networks, and to capture the local features and structure information effectively.

Under the same experimental environment, the results show that the proposed model is superior to the non-hybrid DL models, including CNN-based and LSTM-based models. In addition, the proposed bidirectional LSTM+CNN model outperformed another hybrid model (CNN+bidirectional LSTM model) greatly for PPI extraction. The visualization and comparison of the hidden features learned by different DL models further confirmed the effectiveness of the proposed model.

For future work, we will apply the proposed hybrid DL model to other NLP tasks in order to explore its applicability, and will consider more complex network architectures (e.g., a deep hybrid model with more layers or a deep reinforcement learning model) to improve the performance.

**Author Contributions:** C.Q. and Z.L. conceived and designed the model; C.Q. performed the experiment and analyzed the results; C.Q. wrote the preliminary version of this manuscript; Z.W.L. and S.W. revised the manuscript. All authors have read and agreed to the published version of the manuscript.

**Funding:** This study was partially supported by the National Natural Foundation of China under grant no. 61472117.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Scott, D.E.; Bayly, A.R.; Abell, C.; Skidmore, J. Small molecules, big targets: Drug discovery faces the protein–protein interaction challenge. *Nat. Rev. Drug Discov.* **2016**, *15*, 533–550. [CrossRef] [PubMed]
2. Bader, G.D.; Betel, D.; Hogue, C.W.V. BIND: The biomolecular interaction network database. *Nucleic Acids Res.* **2003**, *31*, 248–250. [CrossRef] [PubMed]
3. Zanzoni, A.; Montecchi-Palazzi, L.; Quondam, M.; Ausiello, G.; Helmer-Citterich, M.; Cesareni, G. MINT: A molecular interaction database. *FEBS Lett.* **2002**, *513*, 135–140. [CrossRef]
4. Kerrien, S.; Aranda, B.; Breuza, L.; Bridge, A. The intact molecular interaction database in 2012. *Nucleic Acids Res.* **2012**, *38*, D525. [CrossRef] [PubMed]
5. Bunescu, R.; Mooney, R.; Ramani, A.; Marcotte, E. Integrating co-occurrence statistics with information extraction for robust retrieval of protein interactions from medline. In Proceedings of the HLT-NAACL Workshop on Linking Natural Language Processing and Biology (BioNLP '06), New York, NY, USA, 8 June 2006; pp. 49–56.
6. Fundel, K.; Küffner, R.; Zimmer, R. RelEx—Relation extraction using dependency parse trees. *Bioinformatics* **2007**, *23*, 365–371. [CrossRef]
7. Segura-Bedmar, I.; Martínez, P.; De Pablo-Sánchez, C. A linguistic rule-based approach to extract drug-drug interactions from pharmacological documents. *BMC Bioinform.* **2011**, *12* (Suppl. 2). [CrossRef]

8. Cui, B.; Lin, H.; Yang, Z. SVM-based protein-protein interaction extraction from medline abstracts. In Proceedings of the 2nd International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA '07), IEEE, Zhengzhou, China, 14–17 September 2007; pp. 182–185.
9. Erkan, G.; Özgür, A.; Radev, D.R. Semi-supervised classification for extracting protein interaction sentences using dependency parsing. In Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL '07), Prague, Czech Republic, 28–30 June 2007; Volume 7, pp. 228–237.
10. Sun, C.; Lin, L.; Wang, X. Using maximum entropy model to extract protein-protein interaction information from biomedical literature. In Proceedings of the Third International Conference on Intelligent Computing, ICIC 2007, Qingdao, China, 21–24 August 2007; pp. 730–737.
11. Segura-Bedmar, I.; Martínez, P.; de Pablo-Sánchez, C. Using a shallow linguistic kernel for drug-drug interaction extraction. *J. Biomed. Inform.* **2011**, *44*, 789–804. [CrossRef]
12. Quan, C.; Wang, M.; Ren, F. An unsupervised text mining method for relation extraction from biomedical literature. *PLoS ONE* **2014**, *9*, e102039. [CrossRef]
13. Arora, K.; Rangarajan, A. A Compositional Approach to Language Modeling. *arXiv* **2016**, arXiv:1604.00100.
14. Bengio, Y.; Schwenk, H.; Senécal, J.-S.; Morin, F.; Gauvain, J.L. Neural probabilistic language models. In *Innovations in Machine Learning, Studies in Fuzziness and Soft Computing*; Springer: Berlin, Germany, 2006; pp. 137–186.
15. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. In Proceedings of the ICLR, Scottsdale, AZ, USA, 2–4 May 2013.
16. Krizhevsky, A.; Ilya, S.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, 1097–1105. [CrossRef]
17. Khawaldeh, S.; Pervaiz, U.; Rafiq, A.; Alkhawaldeh, R. Noninvasive Grading of Glioma Tumor Using Magnetic Resonance Imaging with Convolutional Neural Networks. *Appl. Sci.* **2018**, *8*, 27. [CrossRef]
18. Dong, J.; Gao, Y.; Lee, H.; Zhou, H. Action Recognition Based on the Fusion of Graph Convolutional Networks with High Order Features. *Appl. Sci.* **2020**, *10*, 1482. [CrossRef]
19. Collobert, R.; Weston, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 5–9 July 200; ACM: New York, NY, USA; pp. 160–167.
20. Rios, A.; Kavuluru, R. Convolutional neural networks for biomedical text classification: Application in indexing biomedical articles. In Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics, Atlanta, GA, USA, 30 August–2 September 2015; ACM: New York, NY, USA; pp. 258–267.
21. Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; Kuksa, P. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* **2011**, *12*, 2493–2537.
22. Kalchbrenner, N.; Grefenstette, E.; Blunsom, P. A convolutional neural network for modelling sentences. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, MD, USA, 22 June 2014; pp. 655–665.
23. Kim, Y. Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; Association for Computational Linguistics: Stroudsburg, PA, USA; pp. 1746–1751.
24. Hua, L.; Quan, C. A shortest dependency path based convolutional neural network for protein-protein relation extraction. *BioMed Res. Int.* **2016**. [CrossRef] [PubMed]
25. Quan, C.; Hua, L.; Sun, X.; Bai, W. Multichannel convolutional neural network for biological relation extraction. *BioMed Res. Int.* **2016**, 1–10. [CrossRef]
26. Peng, Y.; Lu, Z. Deep learning for extracting protein-protein interactions from biomedical literature. In Proceedings of the BioNLP, Vancouver, Canada, 4 August 2017; pp. 29–38.
27. Tu, Z.; Hu, B.; Lu, Z.; Li, H. Context-dependent translation selection using convolutional neural network. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, Beijing, China, 26–31 July 2015; pp. 536–541.
28. Funahashi, K.-I.; Nakamura, Y. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Netw.* **1993**, *6*, 801–806. [CrossRef]
29. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]

30. Lai, S.; Xu, L.; Liu, K.; Zhao, J. Recurrent convolutional neural networks for text classification. In Proceedings of the AAAI, Austin, TX, USA, 25–30 January 2015; Volume 333, pp. 2267–2273.
31. Hsieh, Y.L.; Chang, Y.-C.; Chang, N.W.; Hsu, W.L. Identifying Protein-protein Interactions in Biomedical Literature using Recurrent Neural Networks with Long Short-Term Memory. In Proceedings of the Eighth International Joint Conference on Natural Language Processing, Taipei, Taiwan, 27 November–1 December 2017; pp. 240–245.
32. Yadav, S.; Ekbal, A.; Saha, S.; Kumar, A.; Bhattacharyya, P. Feature Assisted bi-directional LSTM Model for Protein-Protein Interaction Identification from Biomedical Texts. *Knowl. Based Syst.* **2019**, *166*, 18–29. [CrossRef]
33. Chen, X.; Xie, H.; Cheng, G. Trends and Features of the Applications of Natural Language Processing Techniques for Clinical Trials Text Analysis. *Appl. Sci.* **2020**, *10*, 2157. [CrossRef]
34. Zhou, C.; Sun, C.; Liu, Z.; Lau, F.C.M. A C-LSTM Neural Network for Text Classification. *arXiv* **2015**, arXiv:1511.08630.
35. Zhang, D.; Tian, L.; Chen, Y. Combining Convolution Neural Network and Bidirectional Gated Recurrent Unit for Sentence Semantic Classification. *IEEE Access* **2018**, *6*, 73750–73759. [CrossRef]
36. Stanford Parser. Available online: https://nlp.stanford.edu/software/lex-parser.shtml (accessed on 27 December 2019).
37. CoNLL-U Viewer. Available online: http://www.let.rug.nl/kleiweg/conllu/ (accessed on 27 December 2019).
38. Pyysalo, S.; Ginter, F.; Moen, F.; Salakoski, T. Distributional semantics resources for biomedical text processing. In Proceedings of the Languages in Biology and Medicine (LBM '13), Tokyo, Japan, 12–13 December 2013; pp. 39–44.
39. Razvan, C.B.; Ruifang, G.; Rohit, J.K.; Edward, M.M.; Raymond, J.M.; Arun, K.R.; Yuk, W.W. Comparative experiments on learning information extractors for proteins and their interactions. *Artif. Intell. Med.* **2005**, *33*, 139–155.
40. Pyysalo, S.; Ginter, F.; Heimonen, J.; Björne, J.; Boberg, J.; Järvinen, J.; Salakoski, T. BioInfer: A corpus for information extraction in the biomedical domain. *BMC Bioinform.* **2007**, *8*, 1–24. [CrossRef] [PubMed]
41. Turku BioNLP group. Available online: http://bionlp.utu.fi/ (accessed on 27 December 2019).
42. Gensim–Deep Learning with Word2vec. Available online: https://radimrehurek.com/gensim/models/word2vec.html (accessed on 5 October 2019).
43. Jolliffe, I.T. *Principal Component Analysis*, 2nd ed.; Springer-Verlag: New York, NY, USA, 2002.
44. Maaten, L.V.D.; Hinton, G.E. Visualizing High-Dimensional Data Using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.