Effectively Identifying Compound-Protein Interaction using Graph Neural Representation

Xuan Lin, Zhe Quan*, Zhi-Jie Wang, Yan Guo, Xiangxiang Zeng, *Senior Member, IEEE,* and Philip S. Yu, *Fellow, IEEE*

Abstract—Effectively identifying compound-protein interactions (CPIs) is crucial for new drug design, which is an important step in silico drug discovery. Current machine learning methods for CPI prediction mainly use one-demensional (1D) compound/protein strings and/or the specific descriptors. However, they often ignore the fact that molecules are essentially modeled by the molecular graph. We observe that in real-world scenarios, the topological structure information of the molecular graph usually provides an overview of how the atoms are connected, and the local chemical context reveals the functionality of the protein sequence in CPI. These two types of information are complementary to each other and they are both significant for modeling compound-protein pairs. Motivated by this, we propose an end-to-end deep learning framework named *GraphCPI*, which captures the structural information of compounds and leverages the chemical context of protein sequences for solving the CPI prediction task. Our framework can integrate any popular graph neural networks for learning compounds, and it combines with a convolutional neural network for embedding sequences. To compare our method with classic and state-of-the-art deep learning methods, we conduct extensive experiments based on several widely-used CPI datasets. The experimental results show the feasibility and competitiveness of our proposed method.

Index Terms—Graph neural networks, machine learning, compound-protein interaction.

1 Introduction

Effectively identifying compound-protein interactions (CPIs) is a key task in pharmacology and drug discovery [2]. In the CPIs task, compound refers to molecular compounds (instead of ionic compounds), which can be represented by a molecule graph with atoms as nodes and chemical bonds as edges; while proteins are sequences of amino acids. CPI prediction is to find the potential compound-protein pairs where a protein is targeted by at least a compound. However, the predicted CPI does not mean that a positive or negative influence on functions triggered by proteins. This may affect the disease conditions [3], [4]. Figure 1 shows a CPI example with compound-protein pair (*Aspirin-Phospholipase*). In the figure, the dotted line indicates the *Hydrophobic* interaction.

By understanding the CPI task, it can help users find out candidate compounds that are able to inhibit the protein, and it benefits many other bioinformatic applications such as drug resistance [5], and cancer research [6]. As a result, CPI prediction has received much attention in recent years [7], [8], [9]. Traditional machine learning approaches for

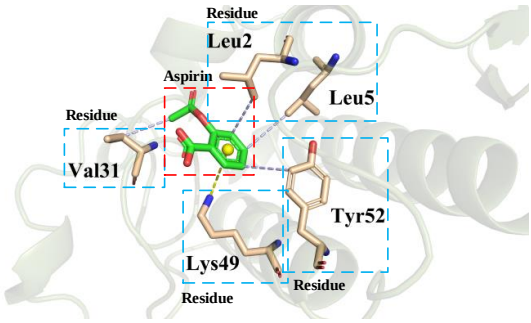


Fig. 1. Example for visualization of CPI: Aspirin-Phospholipase (PDB ID:6MQF)

CPI prediction can be roughly classified into feature-based and similarity-based methods. Generally speaking, feature-based methods construct input vector from descriptors of compounds and proteins, such as molecular docking [10] and the 3D structure-embedded of protein [11], which are often difficult to obtain. On the other hand, similarity-based methods rely on hypothesis that compounds with similar structures should have similar properties [12]. The representative works of this line of methods are like [13], [14], etc. Recently, owing to the remarkable success in various machine learning tasks (e.g., image recognition [15], natural language processing [16], [17]), deep learning methods are also exploited for CPI prediction [18], [19], [20]. In this branch, existing methods consider either label/one-hot encodings or the fingerprint of molecules. However, they have not considered the chemical bond of atoms and the local chemical context of amino acids. We observe that, in real-world scenarios, the topological structure information usually provides an overview of how the atoms are connected,

- *X. Lin is with (i) the College of Computer Science and Technology, Hunan University, Changsha 410082, China, and (ii) the Key Laboratory of Intelligent Computing and Information Processing, Ministry of Education (Xiangtan University), Xiangtan 411105, China. E-mail: jack_lin@hnu.edu.cn.*
- *Z. Quan, Y. Guo and X. Zeng are with the College of Computer Science and Technology, Hunan University, Changsha 410082, China. E-mail: {quanzhe, guoyan, xzeng}@hnu.edu.cn.*
- *Z.-J. Wang is with (i) the College of Computer Science, Chongqing University, Chongqing 400044, China, and (ii) the Ministry of Education (MOE) Key Laboratory of Dependable Service Computing in Cyber Physical Society, Chongqing University, China. E-mail: cszjwang@cqu.edu.cn.*
- *P. S. Yu is with the Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607, USA. E-mail: psyu@cs.uic.edu*

** means the corresponding author. The preliminary version of this paper was accepted at the conference BIBM'2019 [1].*

and the local chemical context reveals the functionality of the protein sequence in CPI, which is just like the semantic meaning of a word in a sentence. These two types of information are complementary to each other and they are both important for modeling compounds and proteins.

Inspired by the aforementioned facts, in this paper we attempt to develop an end-to-end deep learning framework that combines the local chemical context for sequences and the topological structure for molecules to learn the interaction between compounds and proteins. To this end, we propose a **graph** neural representation framework for **CPI** prediction, and we refer to it as *GraphCPI*. Our framework consists of two major building blocks: One of the major building blocks learns low-dimension vector representations for protein sequences using a convolutional neural network (CNN), while the other building block learns graph representations for compounds using graph neural network (GNN), respectively. Specifically, the CNN building block extracts the local chemical context information for amino acids in proteins; in the process of extracting, we propose to incorporate *Prot2Vec* [21], which was previously used for representation and feature extraction for biological sequences, to encode the amino acids to a distributed representation. In this way, we can efficiently avoid the limitation of the label/one-hot encodings of amino acids, since it often ignores the context information. Meanwhile, the GNN building block extracts the topological features for compounds by constructing a molecular graph. The GNN building block is pretty flexible, which can be replaced by any popular graph-based neural networks. The learned representations for both compounds and proteins are then passed to a dense neural network for predicting the interaction. Different from the existing feature-based and similarity-based methods, our framework needs neither molecular docking nor 3D structure-embedding of the proteins. Additionally, the proposed framework takes advantage of the topological information of atoms encoded in the graph neural representation, which differs our framework from the existing deep learning methods such as *DeepCPI* [22]. In a nutshell, the novelty and main contributions of this paper are as follows:

- We propose a framework that incorporates the advanced graph neural representation for compound and pre-trained embedding techniques for protein sequences together. To the best of our knowledge, in the CPI field this paper is the first to combine the local chemical context and topological structure to learn the interaction between compound-protein pairs.
- We conduct extensive experiments based on several widely-used CPI datasets with various imbalance ratios. The experimental results demonstrate the feasibility and competitiveness of our proposed framework, compared against the classic and state-of-the-art methods.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 introduces the proposed framework for CPI prediction task; Section 4 covers and analyzes the experimental results. Finally, we conclude the paper in Section 5.

2 Related Work

Compound-protein interactions (CPIs) prediction has been an interesting topic in drug discovery. Existing methods for CPI prediction can be roughly classified into two categories: (i) traditional methods, and (ii) deep learning-based methods. Next, we review these works, respectively.

2.1 Traditional Methods

Traditional methods focused either on simulation-based methods (e.g., descriptors), or on machine learning based models, which heavily rely on domain similarity [23]. For example, Jaroch *et al.* [24] integrated the chemical attributes of compounds, the genomic attributes of proteins and the known CPIs into a unified mathematical framework. With a variety of similarity information, Bleakley *et al.* [25] presented the bipartite local model (BLM) to predict CPIs, and they trained local support vector machine (SVM) classifiers with the help of known interactions. Later, Mei *et al.* [19] improved BLM by exploiting the already known interactions of neighbors, which compensates the lack of BLM. To predict the drug-target interaction (DTI) involving new drugs or targets for unknown interactions, Ezzat *et al.* [26] proposed the matrix factorization method that combines with graph regularization. Additionally, Cheng *et al*. [27] presented a method named *PUCPI* that employs biased-SVM to predict CPIs using positive and unlabeled examples.

Although classic methods show reasonable performance in CPI prediction, they are often computational expensive, require additional expert knowledge, or the 3D structure-embedded of protein, which are often difficult to obtain. Different from these classic methods, the proposed framework is able to automatically extract features from the data, and requires neither domain knowledge nor 3D structure of the target/protein. These main features make our proposed framework applicable to large scale CPI datasets.

2.2 Deep Learning-based Methods

In recent years, much attention has been devoted to applying deep learning techniques for DTI prediction (which is an alternative name of CPI prediction). For example, Gao *et al.* [28] proposed an end-to-end neural network model to predict DTIs directly from low level representations, and they provided biological interpretation by using two-way attention mechanism. Moreover, Wan *et al.* [29] developed a nonlinear end-to-end learning model named NeoDTI that integrates diverse information from heterogeneous network data, and it automatically learned topology-preserving representations of drug-target pair to facilitate DTI prediction. Moreover, Karimi *et al*. [20] presented a DTI model named DeepAffinity that represents protein and SMILES sequences based on a recurrent neural network (RNN) , and note that SMILES (Simplified Molecular-Input Line Entry System) is a single-line text representation to encode the chemical context of molecule [30]. Recently, a model called DeepCPI was proposed for CPI prediction [22].

Among the studies in this line branch, DeepCPI [22] could be the one most relevant to our work, since it addresses the problem same to ours, and uses also GNN and CNN. Specifically, DeepCPI uses a traditional GNN, based

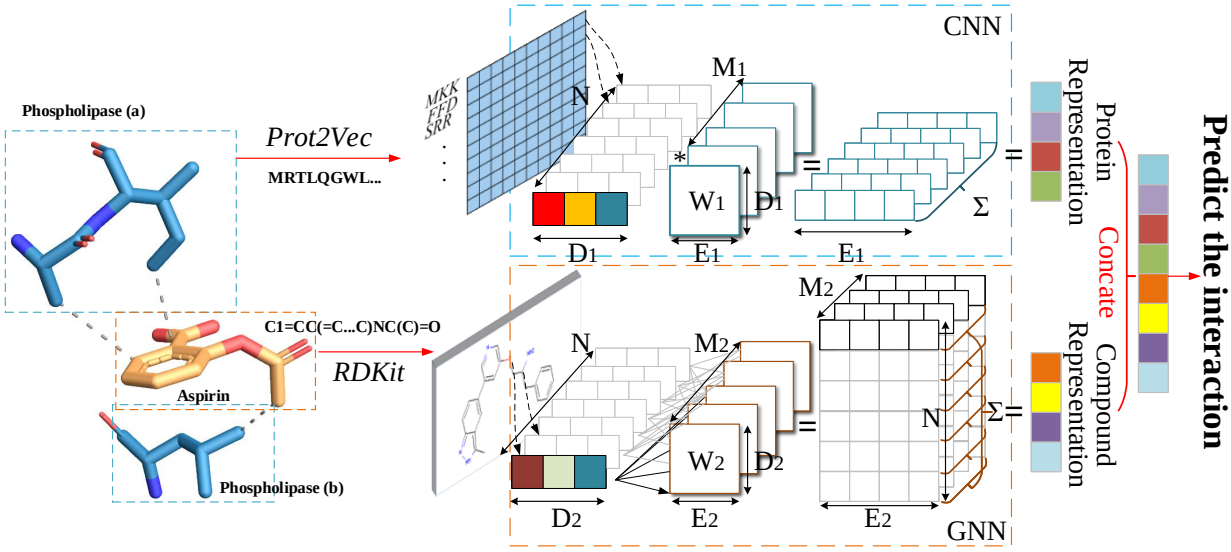


Fig. 2. An overview of GraphCPI. Note that, Phospholipase (a) and (b) refer to two residues of Phospholipase. In fact, Phospholipase has many residues, we only plot two residues, due to space limit. The top part illustrates a 3-layer CNN that learns representation for proteins, while the bottom part illustrates a 3-layer GNN that learns representation for compound. In the figure, D_i, E_i, M_i and W_i ($i \in \{1, 2\}$) denote the input feature, output feature, the number of filter and weight matrix, respectively. N denotes the batch size. $\sum$ means sum up the D convolution results and adds a constant bias to compute the output feature map.

on representation of r-radius fingerprints, to encode the molecular structure of compounds; and it uses a CNN to encode protein sequences without pre-trained embedding. Compared with DeepCPI, for compound representation our proposed deep learning framework incorporates the topological information obtained from GNNs to encode the atoms, and it uses pre-trained embedding (e.g., Prot2Vec) to encode the amino acids to boost the representation learning of proteins. Moreover, our framework could be much more flexible, since it allows us to integrate any popular GNN model.

3 The GraphCPI Framework

In this section, we firstly describe an overview of the proposed framework called *GraphCPI* (Section 3.1), and then we present the representation learning for compounds and for proteins, respectively (Sections 3.2 $\sim$ 3.3). Finally, we present the detail of CPI task prediction (Section 3.4).

3.1 Overview of GraphCPI

Figure 2 gives an overview of our proposed framework, named *GraphCPI*, for the compound-protein pair task. Generally, *GraphCPI* takes *the molecular structure of the compound* and *the symbolic sequences of the protein* as the inputs. Then, the molecular structure of the compound in SMILES [30] string is encoded into a molecular graph, while the sequence of the protein is encoded into a distributed representation (like Word2Vec [16]), forming a matrix. Later, the molecular graph is fed into graph neural networks (GNNs) for capturing the structural information of the compound, while the matrix is fed into convolutional neural networks (CNNs) to obtain local chemical context of the protein. As a result, we obtain two latent representations for compound and protein, respectively. After that, we further feed the concatenation of two latent representations into a stack of fully connected layers, and finally GraphCPI outputs a binary value for the compound-protein pair (1 means interaction, and 0 means otherwise).

3.2 Graph Representation for Compounds

As we know, compounds are often represented in the format of SMILES provided by many database (e.g, ZINC, PubChem). The molecular structure is a significant part in graph neural representation learning for compounds. To represent such a structure efficiently, most of existing methods either use similarity-based manner/strategy to infer the unknown CPI, or use molecular fingerprints and protein family databases to represent compound-protein pairs. These methods usually have fixed features, while they cannot learn more features for compound and protein representation. To alleviate such dilemmas, in this paper we propose to use the end-to-end representation learning that combines with the advanced embedding techniques for compounds and proteins. Specifically, for each input SMILES string of a chemical compound, we use RDKit [31] tool to transform it into a molecular graph, which is represented as $G = (V, E)$, where V denotes the atomic feature and E denotes the chemical bond value between adjacent atoms. For ease of understanding, we take Aspirin (*O=C(C)Oc1ccccc1C(=O)O*) as an example, as shown in Figure 3. Firstly, it is transformed into its 2D structures by using RDKit tool. Then predefined atomic features are assigned to each node based on its atomic number. In this paper, we adopt multi-dimensional binary feature vector to encode 5 types of atomic features, including atomic symbol, adjacent atoms, adjacent hydrogens, implicit value and aromaticity. Specifically, we use a binary vector of size 44, denoted by A_1 (i.e., green), to encode the atomic symbol. For example, the 12th atom '*O*' is encoded by one-hot encoding as (01

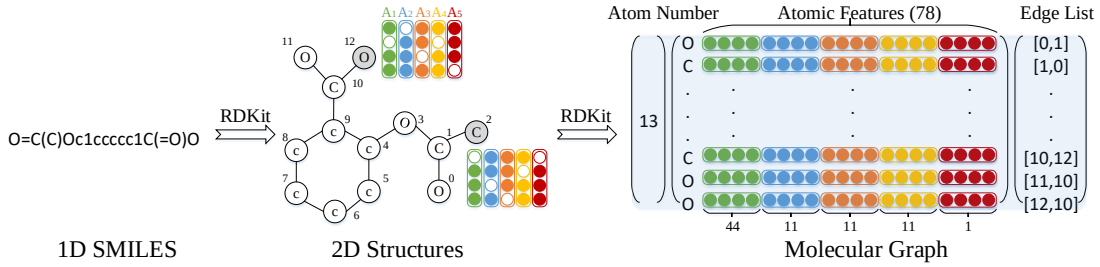


Fig. 3. An example of the molecular graph representation of Aspirin by using RDKit.

... 00) while (10 ... 00) for the 2nd atom '*C*', and '*unknow*' represents the unknown atom symbol in the rare case. Then we adopt A_2 (i.e., blue) of size 11 to represent the adjacent atoms of each atom in the molecule, and A_2 is defined to be its number of directly-bonded neighbors (i.e., degree). For example, the 1st atom has 3 directly-bonded neighbors, so we have its A_2 as (00010...0). Next, the adjacent hydrogens is denoted by A_3 (i.e., orange) with a size of 11 to describe the total number of Hs (explicit and implicit) on the atom. Here "explicit" refers to atoms in the graph while "implicit" refers to atoms that are not in the graph (i.e., Hydrogens). For example, the adjacent hydrogens of atoms on the benzene ring is 1, so we encode its A_3 by (10...0), while the adjacent hydrogens of 3rd atoms is 3, then its A_3 is encoded by (00010...0). Meanwhile, we represent the implicit value of each atom as A_4 of size 11 to record the number of implicit Hs on the atom. For example, the implicit value of 12th atom is 1, so we encode its A_4 by (010...0). And we use a bool vector A_5 to encode the aromaticity that means whether the current atom is in an aromatic structure. For example, the 4th atom is in an aromatic structure so we encode its A_5 by 1 while A_5 of the 1st atom is encoded by 0. All processings can be implemented by the corresponding functions of RDKit tool[1]. Finally, we obtain the molecular graph representation of Aspirin that consists of atom number (i.e., total number of atoms), atomic features and edge features (i.e., edge list), so we can extract the structural information from a molecular graph. The list of the initial atom features is summarized in Table 1.

Once such a molecular graph is obtained, one can fed it into any popular graph neural network model (e.g., GCN [32], GAT [33], GIN [34]) to obtain the structural information of the compound. As we will show later, although different graph neural network models may exhibit their own advantages for different evaluation metrics, but their performance gaps are small.

1. https://www.rdkit.org/docs/source/rdkit.Chem.rdchem.html

3.3 Sequence Representation for Proteins

Proteins are generally represented as a string of ASCII characters that represent 25 types of amino acids. In this paper, we propose to first encode the amino acids into *d*-dimensional vector using *Prot2Vec* [21]. Different from the previous methods used one-hot or label encoding to represent the protein sequence, each amino acid type is simply represented by label encoding (i.e., integer) according to its corresponding alphabetical symbol. For example, we denote *Alanine* (A) by 1, *Glutamine* (G) by 7, *Threonine* (T) by 19 and so on, respectively. And *Alanine* (A) can be also encoded by one-hot encoding as (10...00), each bit represents one type of 25 amino acid sequences. However, a single amino acid often makes no sense, we adopt a fixed length N-gram splitting method to split the sequence into meaningful biological words. Compared with the commonly used label encoding methods, the fixed-length N-gram divides the sequence into a sequence of N-grams. Thus, each N-gram can be regarded as "biological word". Intuitively, it can generate more "word context" than the commonly used label encoding. To balance the trade-off between the computation complex (i.e., 20^N) and biological significance, here we set N as 3. Specifically, the protein sequence is divided into the first three amino acids as the initial position in order to obtain 3 group of subsequences, and then remove the duplicated subsequences among them to obtain the final subsequences. We take HTR1D (Human) Recombinant Protein as an example. As shown in Figure 4, the protein sequence totally consists of 377 amino acids. The first group of subsequences are generated with 'M' in red as the starting amino acid, while 'S' in blue as the starting point of the second group of subsequences. And the duplicated subsequences among three groups are removed (e.g., 'LIT').

TABLE 1
The list of initial atom features

Atom Feature	Size	Description
Atomic symbol	44	[C, N, O, S, F, Si, P, Cl, Br, Mg, Na, Ca, Fe, As, Al, I, B, V, K, Tl, Yb, Sb, Sn, Ag, Pd, Co, Se, Ti, Zn, H, Li, Ge, Cu, Au, Ni, Cd, In, Mn, Zr, Cr, Pt, Hg, Pb, Unknow] (One-hot)
Adjacent atoms	11	number of atoms in the molecules of an element [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10] (One-hot)
Adjacent hydrogens	11	total number of hydrogen [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10] (One-hot)
Implicit value	11	the implied value of atoms [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10] (One-hot)
Aromaticity	1	whether atoms are aromatic [0/1]

Finally, we concatenate the remaining three groups to obtain the finally 3-gram subsequences.

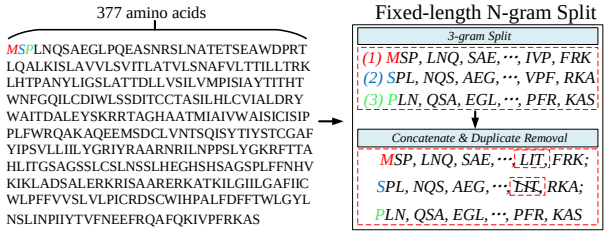


Fig. 4. Example of 3-gram split protein sequence.

After we obtain the meaningful "biological words", then for each such a word we map it to an embedding vector by looking up a pre-trained embedding dictionary that has 9048 words and a 100-dimension vector per word, which is obtained from Swiss-Prot[2]. One can set the fixed length to a specific value (e.g., 1000), and then the sequence will be truncated when its length is over the default value (e.g., 1000); otherwise, it will be padded with 0. As a result, we transform each sequence of amino acids to a matrix, where each row is the embedding of a biological word. Algorithm 1 shows the detailed steps of the embedding process. In general, a protein sequence with fixed length l is split into a set of subsequence or "biological word" (Line 1). Then it maps subsequence by looking up each of the sequence embedding from the pretrained dictionary D, while it randomly generates values if it is not in D (Line 2-6). Finally, it constructs a protein matrix A by aggregating embedding vectors (Line 7), where each line represents the pretrained vector of subsequence. The matrix shall be fed into a CNN to obtain the local chemical context of the protein.

Algorithm 1: Prot2Vec Embedding

Input: a protein sequence P, dictionary D, amino acid x, subsequence set s, fixed length l, vector dimension d.

Output: protein matrix A

1 $s = (x_1x_2x_3, ..., x_jx_{j+1}x_{j+2})(1 \leq j < l) \longleftarrow split(P)$;
2 **for** *j=1 to l* **do**
3 **if** $x_j \notin dictionary$ **then**
4 embedding $a_j \in \Re^d \longleftarrow randomgenerate(x_j)$;
5 $a_j \xleftarrow{map} x_j$ // by using D ;
6 protein matrix $A \longleftarrow \sum_{j=1}^{m} a_j$;
7 return $A \in \Re^{l \times d}$;

3.4 Compound-Protein Interaction Prediction

It is easy to understand that, one can view the compound-protein pair prediction as a binary classification problem by predicting the interaction value. With the representation learned from the previous subsections, in what follows, we are ready to integrate all features from compounds and proteins to predict the interaction.

2. https://www.uniprot.org/

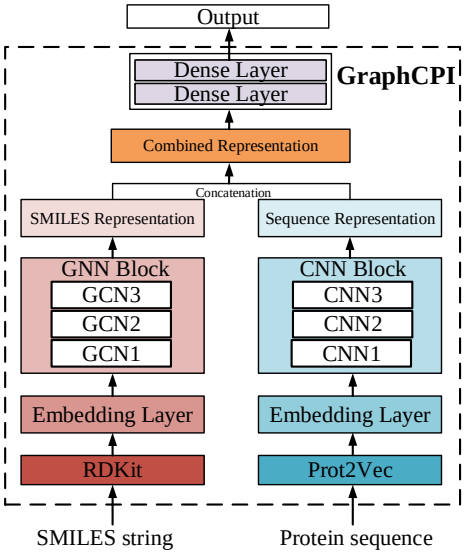


Fig. 5. Workflow for CPI prediction and main building blocks.

Generally, we concatenate two kinds of representations, and feed them to two fully-connected layers to output the interaction value. Figure 5 shows the workflow for CPI prediction and the main building blocks. More precisely, for the compound SMILES, we first convert it to a molecular graph by RDKit tool. And we use GCN model [32] for learning on graph representation of compound, we use 78-dimensional atomic features as the node feature of molecular graph. In our GCN model, we adopt three consecutive GCN layers, each followed by a ReLU activation function. Then a global max pooling layer is applied to obtain the final graph representation vector. For the protein sequence, we use a string of amino acid sequences and adopt 1D CNN layers to learn a sequence representation vector. Specifically, the protein sequence is first split by fixed length 3-gram splitting method, then an embedding layer is applied to the generated subsequences where each 3-gram subsequences is represented by a 100-dimensional pre-trained word vector. Next, three 1D convolutional layers are adopted to learn different levels of features. Finally, a max pooling layer is used to get a final representation vector of the protein sequence. Here the Rectified Linear Unit (ReLU) [35] is selected as the activation function. Then, given a set of compound-protein pairs and the ground-truth interaction values in the training set, its objective is essentially to minimize the loss function L [22] as follows:

$$L(\Theta) = -\sum_{i=1}^{K} logP_{t_i} + \frac{\lambda}{2}\|\Theta\|_2^2, \quad (1)$$

where Θ denotes the set of all weight matrices, bias vectors in our framework (e.g., GCN and CNN), and the embeddings of N-gram words; K is the total number of compound-protein pairs, t_i is the i-th label, and λ represents an L2 regularization hyper-parameter. Here, we adopt back-propagation to train Θ.

TABLE 2
The detailed description of datasets.

	negative ratio	compounds	proteins	positive	negative
human	1	1052	852	3369	3369
	3	1052	852	3369	10107
	5	1052	852	3369	16845
C.elegans	1	1434	2504	4000	4000
	3	1434	2504	4000	12000
	5	1434	2504	4000	20000

4 EXPERIMENT

In this section, we first describe the experimental settings including datasets, evaluation metrics and baseline approaches (Section 4.1). Then, we examine the stability of our framework by using various graph neural networks (Section 4.2). Afterwards, we compare our proposed framework GraphCPI with classic and state-of-the-art methods (Section 4.3 and 4.4). Finally, we demonstrate the impact of different parts integrated in our proposed framework (Section 4.5).

4.1 Experimental Setup

Dataset. Following prior work [36], in our experiments we used several publicly available datasets called *human*, *C.elegans* and *DUD-E* for compound-protein interaction prediction.

- The human dataset contains 3369 positive interactions between 1052 unique compounds and 852 unique proteins.
- The C.elegans dataset contains 4000 positive interactions between 1434 unique compounds and 2504 unique proteins.
- The DUD-E dataset consists of 102 targets, 22,886 active compounds and their affinities against 102 targets. On average, each target has approximately 224 ligands.

Evaluation Metrics. We adopted three kinds of metrics, widely used in CPI task, to evaluate the performance. They are *precision*, *recall* and AUC [36]. The *precision* indicates how many of the samples predicted to be positive are correct. It is computed as:

$$precision = \frac{TP}{TP + FP}, \tag{2}$$

where TP (true positive) means the model correctly predict a positive class, FP (false positive) means the model incorrectly predict a negative class.

The *recall* indicates how many positive samples are correctly predicted by the model. It is can be computed as:

$$recall = \frac{TP}{TP + FN}, \tag{3}$$

where FN (false negative) means the model incorrectly predict a positive class.

The AUC refers to the probability that a randomly chosen positive sample is ranked higher than a negative one [36]. It is computed as:

$$AUC = \frac{\sum I(P_{pos}, P_{neg})}{P * N}, \tag{4}$$

where P and N denote the number of positive and negative samples, respectively; P_{pos} and P_{neg} are the probability of obtaining positive and negative samples by the prediction model, respectively; and $I(P_{pos}, P_{neg})$ is computed as:

$$I(P_{pos}, P_{neg}) = \begin{cases} 1, P_{pos} > P_{neg} \\ 0.5, P_{pos} = P_{neg} \\ 0, P_{pos} < P_{neg} \end{cases}. \tag{5}$$

Baseline Methods. We compared our proposed framework[3] against both classic and state-of-the-art methods. As for classic models, in our experiments we compared four traditional machine learning models[4], including *k-NN*, *random forest (RF)*, *L2-logistic (L2)*, and *SVM*. The results of these models are obtained in [22]. They take multiple similarity measures from different features for both drugs and proteins as the input of classifier. Specifically, the drug similarity is computed from features of chemical structure and side effect, respectively, and the protein similarity is derived from sequence similarity, functional annotation semantic similarity and protein domain similarity, respectively. As for the state-of-the-art model, we directly compared a recently published model called *DeepCPI*[5] [22]. In brief, this method uses the representation of *r*-radius fingerprint to encode the structural information in a chemical compound, and learns node and edge features using a graph neural network (GNN). Meanwhile, it uses the no pre-trained embedding of amino acids to encode protein sequences, and learns the chemical context using a convolutional neural network (CNN). The results of DeepCPI are reported from the original paper. The performance is achieved with the following experimental setting: r-radius is 2, n-gram is 3, window size is 11, vector dimensionality is 10, number of time steps in GNN is 3, and the number of layers in CNN is 3. In contrast, DeepCPI leverages a GNN to map a graph $G = (V, E)$ to a vector $y \in R^d$, by using two transition functions: (1) vertex transition (Eq. 6), and (2) edge transition (Eq. 7).

$$v_i^{(t+1)} = \sigma(v_i^{(t)} + \sum_j h_{ij}^{(t)}), \tag{6}$$

$$e_{ij}^{(t)} = \sigma(e_{ij}^t + g_{ij}^t), \tag{7}$$

where σ is the element-wise sigmoid function (*e.g.*, $1/(1 + e^{-x})$), f is a non-linear activation function, $v_i^{(t)}$ and e_{ij}^t denote the node and edge embeddings between the i-th and j-th nodes at iteration t respectively, $h_{ij}^{(t)}$ denotes the hidden vector, which is obtained by combining node $v_j (j \in neighbor(v_i))$ with edge e_{ij}, and $b_h \in R^d$ is the bias vector, as shown in Eq. 8. The parameter g_{ij}^t is updated by node v_i^t and v_j^t, and $b_g \in R^d$ is the bias vector, as shown in Eq. 9.

$$h_{ij}^{(t)} = f(W \begin{bmatrix} v_j^{(t)} \\ e_{ij}^{(t)} \end{bmatrix} + b_h), \tag{8}$$

$$g_{ij}^t = f(W(v_i^{(t)} + v_j^{(t)}) + b_g). \tag{9}$$

3. https://github.com/jacklin18/GraphCPI
4. These models were obtained from http://admis.fudan.edu.cn/negative-cpi/.
5. https://github.com/masashitsubaki

For ease of comparison, we report the main results of four classic and DeepCPI models from Table 1 (resp. Table 2) for human (resp. C.elegans) dataset in [22]. As for human and C.elegans dataset, an 8/1/1 training/validation/testing random split is adopted by DeepCPI. As for DUD-E dataset, we followed the same training and evaluating strategies as DeepCPI [22]. Specifically, we retrain DeepCPI model with the same parameter settings as in the original paper, where detailed pre-processing on DUD-E dataset refers to the following part (i.e., Other Experimental Details).

Implementation of Our Framework. As mentioned in Section 3.2, once the molecular graph is obtained, one can feed it into any popular graph neural network model to obtain the topological information of compounds. In order to examine the robustness of our proposed framework, we employed respectively three kinds of popular graph neural networks, and used each of them as the building block of the proposed framework. Specifically, these three types of graph neural networks are described as follows.

(1) *GCN*[6] [32]: This approach introduces a graph Laplacian regularization and proposes a 2-layer Graph Convolutional Network (GCN) with the following layer-wise propagation rule:

$$H^{(l+1)} = \sigma(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}), \tag{10}$$

where $\widetilde{A} = A + I_N$ is the adjacency matrix of the undirected graph G, I_N is the identity matrix, $D_{ii} = \sum_j \widetilde{A}_{ij}$, $W^{(l)}$ is a layer-specific trainable weight matrix; $\sigma(.)$ denotes an activation function (*e.g.*, ReLU [37]), and $H(l) \in R^{N\times D}$ is the matrix of activations in the *l*-th layer.

(2) *GAT*[7] [33]: This method uses a graph convolution model based on self-attention mechanism. It adds a *graph attention layer* (GAT) in its component. A set of node features $x \in R^F$ is regarded as input of GAT layer, and then it applies a linear transformation to each node based on a weight matrix $W \in R^{F\times F^{'}}$, where F and $F^{'}$ denote the dimension of input and output nodes, respectively. Additionally, *attention coefficients* between nodes and its 1-hop neighbors are used to compute the output node. That is,

$$e_{ij} = \alpha(W\vec{h}_i, W\vec{h}_j), \tag{11}$$

where e_{ij} denotes the importance degree of node j to node i. To make coefficients easily comparable across different nodes, it normalizes them across all choices of j using the softmax function as follows:

$$\alpha_{ij} = softmax_j(e_{ij}), \tag{12}$$

at last, a non-linearity σ is applied to compute the output node $\vec{h}_i^{'}$ as follows:

$$\vec{h}_i^{'} = \sigma(\sum_{j\in N_i} a_{ij}W\vec{h}_j). \tag{13}$$

(3) *GIN*[8] [34]: This method uses a graph isomorphism network to achieve the maximum discriminative power among GNNs. In particular, multi-layer perceptrons (MLPs) are used in GIN for modelling and parameter updating. It updates the node representation as follows:

$$h_v^{(k)} = MLP^{(k)}((1+\epsilon_{(k)})\cdot h_v^{(k-1)} + \sum_{u\in N(v)} h_u^{(k-1)}), \tag{14}$$

where ϵ is either a learnable parameter or a fixed scalar, $h \in R^F$ is the node feature vector, and $N(i)$ is the neighbors of node i.

For ease of presentation, we refer to our proposed framework integrating GCN [32] as *GraphCPI_GCN*. Similarly, we refer to other three methods integrating GAT [33], GIN [34] as *GraphCPI_GAT*, and *GraphCPI_GIN*, respectively. In what follows, when we mention a method *GraphCPI* without any suffix (e.g., *_GAT*, or *_GCN*), it refers to *GraphCPI_GCN*, unless stated otherwise.

Other Experimental Details. For GNN block, we used an initial atom vector with size 78 as the input of GNN model. For *Prot2Vec*, we used 100-dimension pre-trained embedding representation for *N*-gram words. We constructed matrices with (1000×100) dimensions for protein, where 1000 refers to the fixed length of the protein sequence. The proposed framework was implemented using PyTorch[9] with Tensorflow [38] backend. Our experiments were run on Linux 16.04.10 with Intel(R) Xeon(R) CPU E5-2678 v3@2.50GHz and GeForce GTX 1080Ti (11GB). Table 3 shows the main training parameters, while other omitted parameters were set to default values.

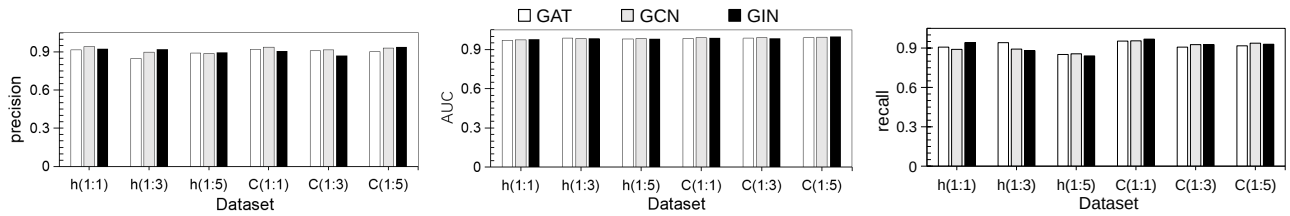
TABLE 3
The main parameter setting

Parameter	Setting	Parameter	Setting
Optimizer	Adam	Learning rate	0.0005
Epoch	1000	Batch size	512
Kernel size	8	Vector dimension	100
Sequence length	1000		

As for imbalanced datasets (e.g., human and C.elegans) in our experiments, we used a python library called *PubChemPy*[10] to obtain the SMILES format of compounds, and we extracted the protein sequence from *Uniprot*[11]. Since the ratio of positive and negative samples may affect the performance, we used three different ratios (1:1, 1:3 and 1:5) to validate the performance of the proposed method. A more detailed description is summarized in Table 2. Regarding the extraction of positive and negative samples, the reader can refer to [36] for details.

As for DUD-E dataset, the original data were obtained from the DUD-E site[12]. It originally consists of 102 targets and 22,886 active compounds (an average of 224 actives per target). We preprocessed the dataset to apply our model. Specifically, we discarded some compounds that can not be implemented by RDKit, and we used *.ism* file to obtain the SMILES string of ligands and extracted the protein sequence

6. https://github.com/tkipf/gcn
7. https://github.com/PetarV-/GAT
8. https://github.com/weihua916/powerful-gnns
9. https://github.com/pytorch/pytorch
10. https://github.com/mcs07/PubChemPy
11. https://www.uniprot.org/uploadlists/
12. http://dude.docking.org/



(a) precision (b) AUC (c) recall

Fig. 6. Testing the stability of our framework using various GNN models, based on human and C.elegans datasets with various imbalance ratios. Note that, for short, in this figure the dataset human is shorten as h, and C.elegans is shorten as C; in addition, the method *GraphCPI_GAT*, *GraphCPI_GIN* and *GraphCPI_GCN* are shorten as GAT, GIN and GCN, respectively.

by the corresponding PDB ID of the target. After preprocessing, we constructed a *.csv* file including 102 targets and 22,806 active compounds/clustered ligands. In the experiment, we evaluated our proposed model using randomly 72 targets as the train set and the rest 30 targets as the test set, we used the balanced dataset for model training. Note that the number of training samples is 22,806 active (i.e. positive sample) and 22,806 decoy (i.e., negative sample), and we randomly chose the active compounds that interact with 30 targets as the test set. Besides, we follow the previous work [39] to train and evaluate our proposed GraphCPI model using 3-fold cross-validation. The folds were split between targets, where all ligands of the same target belong to the same fold. To avoid the impact of homologous proteins, targets belonging to the same protein families were strictly kept in the same fold. Other settings are kept the same as in [22]. The detailed results of chosen 30 target in DUD-E dataset are as shown in Supplementary.

Effectively tuning hyper-parameters is a challenging process during deep neural network modeling, especially for complicated model architectures. It is vital to find out the appropriate sets of hyper-parameters, with respect to its efficiency and effectiveness. Specifically, for each dataset, we performed a grid search, in which we optimized the following hyper-parameters simultaneously: n (the number of "biological word" for protein sequence splitting), k (the number of convolutional layers for sequence embedding), l (the number of graph attention layers for molecular embedding), learning rate, and dropout rate.

4.2 Stability of Our Framework

To examine the performance when various graph neural networks are employed, we conducted extensive experiments based on human and C.elegans datasets with various imbalance ratios (recall Table 2). Figure 6 shows the comparison results of three methods including *GraphCPI_GAT*, *GraphCPI_GCN*, and *GraphCPI_GIN*. Note that, as for these methods, the other parts remain the same, except that they use various neural networks (GCN, GIN, or GAT).

From this figure, one can see that these methods achieved good performance in all these three metrics (precision, AUC, recall) under these benchmark datasets with various imbalance ratios. These results indicate that (i) our proposed framework is feasible, and (ii) our proposed framework may have competitiveness (notice: more experimental results reported later also validate this potential). On the other hand, one can see that the performance gap among these three methods is very small, which can be understood from 18 cases (3 metrics $\times$ 3 imbalance ratios $\times$ 2 datasets), as shown in Figures 6(b)-6(c). In this regard, it indicates that the stability or robustness of our framework.

4.3 Comparison with Classic and State-of-the-Art Methods

Table 4 compares our proposed framework with classical and state-of-the-art methods. In general, GraphCPI outperforms the classic and state-of-the-art deep learning methods on 10 out of 18 situations (cf., the last column in the table). Although SVM achieves some good performance in term of *precision* and *recall* on the human dataset (which is a relatively small dataset, recall Table 2), it does not perform well on the larger dataset C.elegans, and this characteristic is even more obvious when the number of negative samples increases. This is because these classic models (e.g., SVM) heavily rely on fixed hand-crafted features and the similarity matrices of compounds and proteins (e.g., PubChem fingerprints and Pfam domains), which results in poor stability and relatively poor performance.

On the other hand, when we compared with DeepCPI, we found that our method has comparable performance to DeepCPI on human dataset, and particularly it almost fully dominant in all metrics on the C.elegans dataset. This indicates that our proposed method is much more robust when the dataset is large, or even when the dataset is imbalanced. Meanwhile, this also demonstrates the superiority of our proposed method.

4.4 Other Comparison Results

To further demonstrate the good performance of our proposed method, we also compared it with AutoDock Vina and Smina as the non-machine learning methods, and AtomNet, 3D-CNN and DeepCPI as the deep learning models. AutoDock Vina [40] is an open-source program for molecular docking and virtual screening, and Smina [41] is a version of AutoDock Vina specially optimized for high-throughput scoring. AtomNet [42] is the first structure-based deep convolutional neural network method designed to predicting the bioactivity of small molecules, and it combines information about the ligand and the structure of the target, and requires that the locations of each atom in the binding site of the target, while 3D-CNN [39] is also a 3D-structured CNN method, which uses a 3D grid representation generated by docking, and predicts the protein-ligand interaction, and the DeepCPI can be recalled in

TABLE 4
Comparison results of proposed models and baselines on human and C.elegans dataset.

Measure	Negative ratio	k-NN	RF	L2	SVM	DeepCPI	GraphCPI
AUC	1	0.860	0.940	0.911	0.910	0.970	**0.973**
	3	0.904	0.954	0.920	0.942	0.950	**0.983**
	5	0.913	0.967	0.920	0.951	0.970	**0.983**
recall	1	0.927	0.897	0.913	**0.950**	0.918	0.890
	3	0.882	0.824	0.773	0.883	**0.913**	0.892
	5	0.844	0.825	0.666	0.861	**0.975**	0.856
precision	1	0.798	0.861	0.891	**0.966**	0.923	0.940
	3	0.716	0.847	0.837	**0.969**	0.949	0.898
	5	0.684	0.830	0.804	**0.969**	**0.969**	0.886
AUC	1	0.858	0.902	0.892	0.894	0.978	**0.989**
	3	0.892	0.926	0.896	0.901	0.971	**0.989**
	5	0.897	0.928	0.906	0.907	0.971	**0.994**
recall	1	0.827	0.844	0.877	0.818	0.929	**0.955**
	3	0.743	0.705	0.681	0.576	0.921	**0.926**
	5	0.690	0.639	0.582	0.519	0.836	**0.937**
precision	1	0.801	0.821	0.890	0.785	**0.938**	0.937
	3	0.787	0.836	0.875	0.837	**0.916**	0.914
	5	0.774	0.830	0.863	0.896	0.920	**0.930**

* Note: The top (resp. bottom) part refers to the comparison results on human (resp. C.elegans) dataset. The main results of four classic and DeepCPI models are obtained from Table 1 (resp. Table 2) for human (resp. C.elegans) dataset in [22]. The above performance of DeepCPI is achieved using 5-fold cross-validation with the following experimental setting: r-radius is 2, n-gram is 3, window size is 11, vector dimensionality is 10, number of time steps in GNN is 3, and number of layers in CNN is 3.

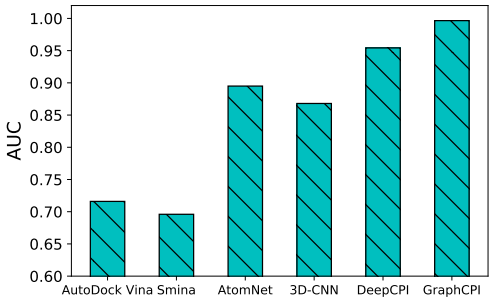


Fig. 7. Comparison of the proposed GraphCPI model to different types of baselines on DUD-E dataset. Note: The AUC scores of AutoDock Vina, Smina, AtomNet, 3D-CNN and DeepCPI model are obtained from Figure 7 in [22].

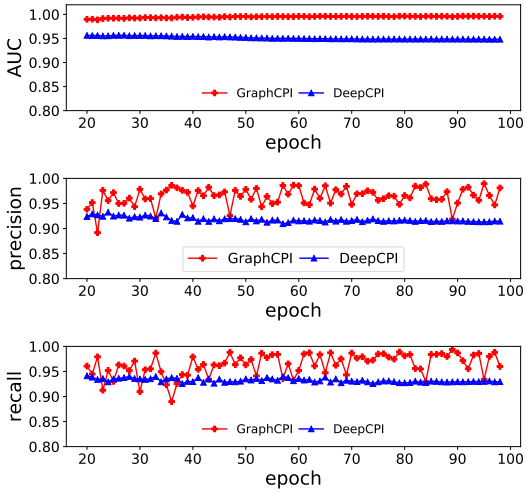


Fig. 8. The AUC, precision, recall scores of DeepCPI and GraphCPI on DUD-E dataset.

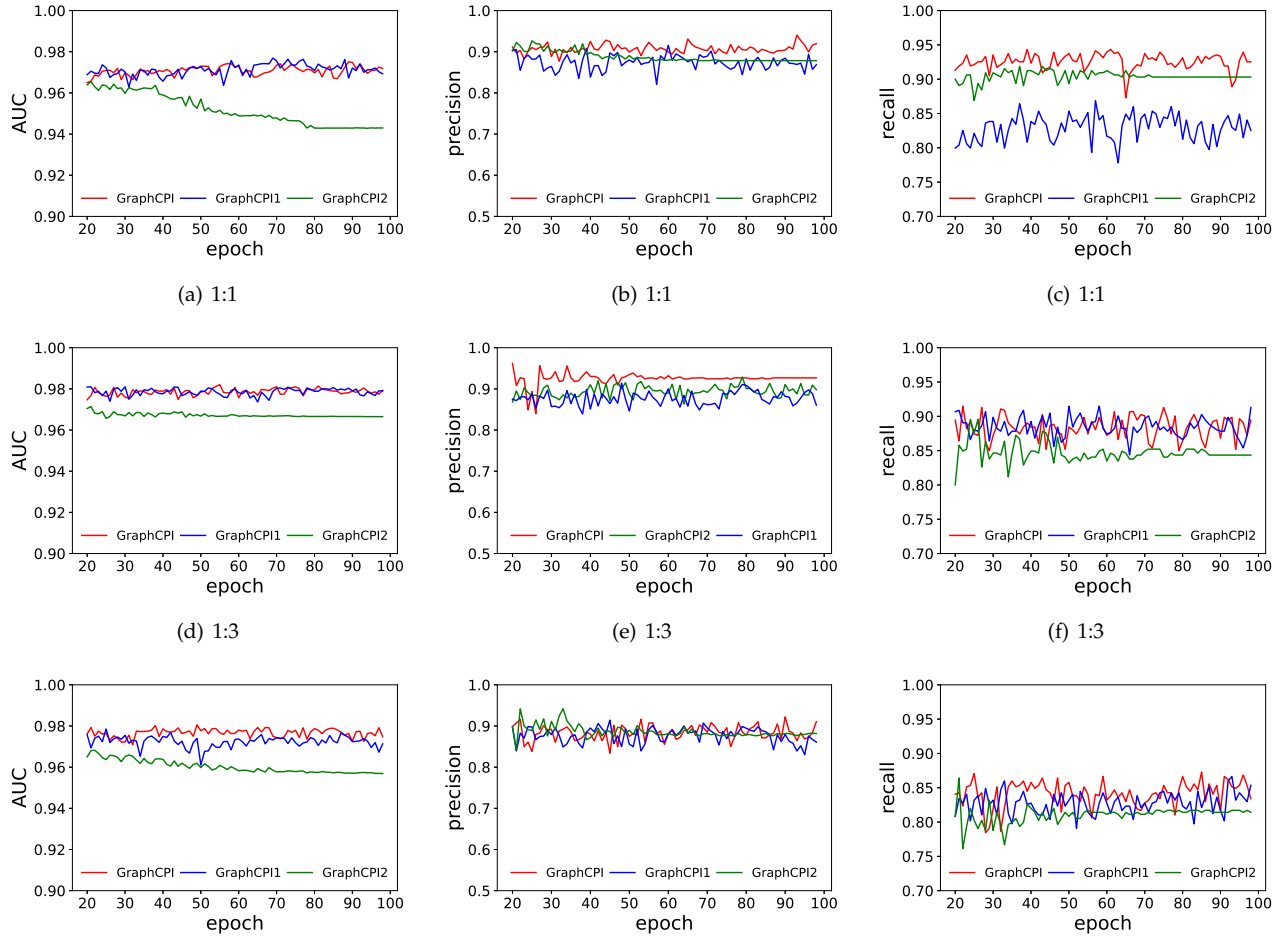
Section 4.1. Figure 7 reports the comparison results on the DUD-E dataset (notice that DeepCPI is also included in the figure, for ease of later discussion and analysis). And the AUC scores of these comparison baselines are reported from Figure 7 of [22]. Meanwhile, we follow [22] to implement our model on the same dataset to obtain the AUC values by using 5-fold cross-validation. One can easily see that our method performed the best among these competitors.

The reader might argue that, the comparisons in Figure 7 might not be very fair, since the input features are different. For example, some methods (e.g., 3D-CNN) used 3D structured feature of proteins, while others (e.g., DeepCPI and our method) used only 1D protein sequence. To alleviate this concern, we conducted a fairer comparison. Specifically, we made a deeper comparison between DeepCPI and our method, since both methods used 1D protein sequence. Figure 8 reports the comparison results in three metrics. Note that we trained the DeepCPI network to obtain a model by using the same processed DUD-E dataset adopted in our paper. By zooming in these figures, one can easily see that our method outperforms the competitor (most similar to our method) in all metrics. These results further demonstrate the superiority of our method. The reason could be that (i) our method incorporates GNN model to obtain the topological information of compound graph, which can obtain more high-order structures than a traditional GNN method based on representation of r-radius fingerprints; (ii) our model jointly considers the pre-trained embedding (e.g., Prot2Vec) of amino acid sequences, which contributes positively to the performance improvement. Moreover, our framework performs much more flexible to integrate any popular GNN model in terms of CPI prediction. Next, we shall further verify this observation by ablation study.

4.5 Ablation Study

To better understand the effect of each part in the proposed framework, we conducted an ablation study for our proposed framework. Specifically, we adopted two variants as follows. Firstly, to investigate the effect of *Prot2Vec*, we removed the local chemical context information obtained by *Prot2Vec* from the framework. This obtains a model named *GraphCPI1*. Note that, for the protein representation, we used the traditional one-hot/label encoding method as the alternative. Secondly, we removed the structural information captured by GNN, obtaining another variant named *GraphCPI2*. Different from *GraphCPI*, *GraphCPI2* uses one-hot/label encoding for representation learning of compounds. The detailed configurations of the variants of our model are listed in Table 5. To speedup the test efficiency, we set the number of epoch to 100, other experimental settings are the same as that in Table 3.

Figure 9 shows the AUC, precision and recall scores of all the variants on the human dataset with various imbalance ratios. Firstly, when the imbalance ratio is 1:1 (positive samples vs. negative samples), we can see from the figure that, *GraphCPI1* performs close to *GraphCPI* on the AUC metric (cf., Figure 9(a)), but it always performs worse than



(a) 1:1 (b) 1:1 (c) 1:1

(d) 1:3 (e) 1:3 (f) 1:3

(g) 1:5 (h) 1:5 (i) 1:5

Fig. 9. Ablation study (AUC, precision and recall) for our proposed model on human dataset with various imbalance ratios. (a) GraphCPI1 is the variant that replaces the Prot2Vec embedding with one-hot/label encoding for the protein; (b) GraphCPI2 is the variant that replaces the GNN block with one-hot/label encoding for the compound.

TABLE 5
The detailed descriptions of the variants

Model	Compound Representation	Protein Representation
GraphCPI	RDKit + GNN	Prot2Vec+CNN
GraphCPI1	RDKit + GNN	One-hot/Label + CNN
GraphCPI2	One-hot/Label + CNN	Prot2Vec+CNN

GraphCPI with regard to *precision* and *recall* from 20 to 100 epochs (cf., Figures 9(b) and 9(c)). These results imply that, only using the structural information of the compounds is not effective enough to learn a good representation for compound-protein pairs. In other words, without the distributed embeddings learned by *Prot2Vec*, the performance of our framework could drop a lot. This is because the one-hot/label encoding lacks the feature engineering on the amino acids. On the other hand, one can see also from Figures 9(a), 9(b) and 9(c) that, *GraphCPI2* performs poorer than *GraphCPI* on all metrics; this implies the limitation of one-hot/label encoding used for compound representation. Meanwhile, it demonstrates that the structural information obtained from GNN plays an important role in compound representation learning. Secondly, when the imbalance ratio is 1:3 (cf., Figures 9(d)-9(f)) and or even 1:5 (cf., Figures 9(g)-9(i)), we can see that *GraphCPI1* performs less satisfactorily than GraphCPI on at least one of the measures, while *GraphCPI2* performs less satisfactorily on at least two of the measures. This further demonstrates the importance of GNN in compound representation learning and of the distributed embeddings learned by *Prot2Vec*. Meanwhile, it also indicates that the proposed framework is robust even if the dataset is imbalanced. In summary, all these results show us that (i) the main components contained in our framework are important and/or effective; and (ii) for the CPI prediction task, our proposed framework *GraphCPI* is robust for both balanced and imbalanced data.

5 CONCLUSION

In this paper, we have proposed a new framework named GraphCPI for Compound Protein Interaction task. GraphCPI uses graph neural representation for compounds and the embedding representation for proteins. Our framework can integrate any popular graph neural networks to obtain the topological information of compounds, and it

combines with a convolutional neural networks to extract the chemical context of protein sequences. We have conducted extensive experiments to compare our method and existing methods, based on several benchmark datasets. The experimental results consistently demonstrate that our proposed is not only feasible but also very competitive, compared against the state-of-the-art methods for CPI tasks.

ACKNOWLEDGMENT

The authors would like to thank all the reviewers for their insightful and valuable suggestions, which significantly improve the quality of this paper. The work was supported in part by the National Natural Science Foundation of China (No. 62122025, 62102140, 61872309, 61972138, 61972425, U1811264), the Hunan Provincial Natural Science Foundation of China under Grants (No. 2020JJ4215, 2021JJ10020 and 2022JJ40451), and the NSF under Grants III-1763325, III-1909323, III-2106758 and SaTC-1930941.

REFERENCES

[1] Zhe Quan, Yan Guo, Xuan Lin, Zhi-Jie Wang, and Xiangxiang Zeng. Graphcpi: Graph neural representation learning for compound-protein interaction. In *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1–6, 2019.

[2] Michael J Keiser, Vincent Setola, John J Irwin, Christian Laggner, Atheir I Abbas, Sandra J Hufeisen, Niels H Jensen, Michael B Kuijer, Roberto C Matos, Thuy B Tran, et al. Predicting new molecular targets for known drugs. *Nature*, 462(7270):175, 2009.

[3] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 2018.

[4] Qinhu Zhang, Lin Zhu, and De-Shuang Huang. High-order convolutional neural network architecture for predicting dna-protein binding sites. *IEEE/ACM transactions on computational biology and bioinformatics*, pages 1184–1192, 2018.

[5] Zoya Khalid and Osman Ugur Sezerman. Prediction of hiv drug resistance by combining sequence and structural properties. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 15(3):966–973, 2018.

[6] Gregor Urban, Kevin M Bache, Duc Phan, Agua Sobrino, Alexander Konstantinovich Shmakov, Stephanie J Hachey, Chris Hughes, and Pierre Baldi. Deep learning for drug discovery and cancer research: Automated analysis of vascularization images. *IEEE/ACM transactions on computational biology and bioinformatics*, pages 1029–1035, 2018.

[7] Monica Campillos, Michael Kuhn, Anne-Claude Gavin, Lars Juhl Jensen, and Peer Bork. Drug target identification using side-effect similarity. *Science*, 321(5886):263–266, 2008.

[8] Kejian Wang, Jiazhi Sun, Shufeng Zhou, Chunling Wan, Shengying Qin, Can Li, Lin He, and Lun Yang. Prediction of drug-target interactions for drug repositioning only based on genomic expression similarity. *PLoS computational biology*, 9(11):e1003315, 2013.

[9] Peng Zhang, Lin Tao, Xian Zeng, Chu Qin, Shangying Chen, Feng Zhu, Zerong Li, Yuyang Jiang, Weiping Chen, and Yu-Zong Chen. A protein network descriptor server and its use in studying protein, disease, metabolic and drug targeted networks. *Briefings in bioinformatics*, 18(6):1057–1070, 2016.

[10] Laura Ascherl, Torben Sick, Johannes T Margraf, Saul H Lapidus, Mona Calik, Christina Hettstedt, Konstantin Karaghiosoff, Markus Döblinger, Timothy Clark, Karena W Chapman, et al. Molecular docking sites designed for the generation of highly crystalline covalent organic frameworks. *Nature Chemistry*, 8(4):310, 2016.

[11] Brice Hoffmann, Mikhail Zaslavskiy, Jean-Philippe Vert, and Véronique Stoven. A new protein binding pocket similarity measure based on comparison of clouds of atoms in 3d: application to ligand prediction. *BMC bioinformatics*, 11(1):99, 2010.

[12] Dhanya Sridhar, Shobeir Fakhraei, and Lise Getoor. A probabilistic approach for collective similarity-based drug–drug interaction prediction. *Bioinformatics*, 32(20):3175–3182, 2016.

[13] Didier Rognan. Chemogenomic approaches to rational drug design. *British journal of pharmacology*, 152(1):38–52, 2007.

[14] Laurent Jacob and Jean-Philippe Vert. Protein-ligand interaction prediction: an improved chemogenomics approach. *Bioinformatics*, 24(19):2149–2156, 2008.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–778, 2016.

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[17] Zhe Quan, Zhi-Jie Wang, Yuquan Le, Bin Yao, Kenli Li, and Jian Yin. An efficient framework for sentence similarity modeling. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 27(4):853–865, 2019.

[18] Richard J Brennan, Tatiana Nikolskya, and Svetlana Bureeva. Network and pathway analysis of compound–protein interactions. In *Chemogenomics*, pages 225–247. Springer, 2009.

[19] Jian-Ping Mei, Chee-Keong Kwoh, Peng Yang, Xiao-Li Li, and Jie Zheng. Drug-target interaction prediction by learning from local information and neighbors. *Bioinformatics*, 29(2):238–245, 2012.

[20] Mostafa Karimi, Di Wu, Zhangyang Wang, and Yang Shen. Deepaffinity: interpretable deep learning of compound–protein affinity through unified recurrent and convolutional neural networks. *Bioinformatics*, 35(18):3329–3338, 2019.

[21] Ehsaneddin Asgari and Mohammad RK Mofrad. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one*, 10(11):e0141287, 2015.

[22] Masashi Tsubaki, Kentaro Tomii, and Jun Sese. Compound–protein interaction prediction with end-to-end learning of neural networks for graphs and sequences. *Bioinformatics*, 35(2):309–318, 2019.

[23] Tapio Pahikkala, Antti Airola, Sami Pietilä, Sushil Shakyawar, Agnieszka Szwajda, Jing Tang, and Tero Aittokallio. Toward more realistic drug–target interaction predictions. *Briefings in bioinformatics*, 16(2):325–337, 2014.

[24] Stefan Jaroch and Hilmar Weinmann. *Chemical genomics: small molecule probes to study cellular function*, volume 58. Springer Science & Business Media, 2007.

[25] Kevin Bleakley and Yoshihiro Yamanishi. Supervised prediction of drug–target interactions using bipartite local models. *Bioinformatics*, 25(18):2397–2403, 2009.

[26] Ali Ezzat, Peilin Zhao, Min Wu, Xiao-Li Li, and Chee-Keong Kwoh. Drug-target interaction prediction with graph regularized matrix factorization. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 14(3):646–656, 2017.

[27] Zhanzhan Cheng, Shuigeng Zhou, Yang Wang, Hui Liu, Jihong Guan, and Yi-Ping Phoebe Chen. Effectively identifying compound-protein interactions by learning from positive and unlabeled examples. *IEEE/ACM transactions on computational biology and bioinformatics*, 15(6):1832–1843, 2016.

[28] Kyle Yingkai Gao, Achille Fokoue, Heng Luo, Arun Iyengar, Sanjoy Dey, and Ping Zhang. Interpretable drug target prediction using deep neural representation. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 3371–3377, 2018.

[29] Fangping Wan, Lixiang Hong, An Xiao, Tao Jiang, and Jianyang Zeng. Neodti: neural integration of neighbor information from a heterogeneous network for discovering new drug–target interactions. *Bioinformatics*, 35(1):104–111, 2018.

[30] David Weininger, Arthur Weininger, and Joseph L Weininger. Smiles. 2. algorithm for generation of unique smiles notation. *Journal of chemical information and computer sciences*, 29(2):97–101, 1989.

[31] Greg Landrum et al. Rdkit: Open-source cheminformatics, 2006.

[32] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[33] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[34] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

[35] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th inter-*

national conference on machine learning (ICML-10), pages 807–814, 2010.

[36] Hui Liu, Jianjiang Sun, Jihong Guan, Jie Zheng, and Shuigeng Zhou. Improving compound–protein interaction prediction by building up highly credible negative samples. *Bioinformatics*, 31(12):i221–i229, 2015.

[37] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[38] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[39] Matthew Ragoza, Joshua Hochuli, Elisa Idrobo, Jocelyn Sunseri, and David Ryan Koes. Protein-ligand scoring with convolutional neural networks. *Journal of chemical information and modeling*, 57(4):942–957, 2017.

[40] Oleg Trott and Arthur J Olson. Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry*, 31(2):455–461, 2010.

[41] David Ryan Koes, Matthew P Baumgartner, and Carlos J Camacho. Lessons learned in empirical scoring with smina from the csar 2011 benchmarking exercise. *Journal of chemical information and modeling*, 53(8):1893–1904, 2013.

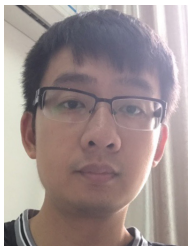
[42] Izhar Wallach, Michael Dzamba, and Abraham Heifets. Atomnet: a deep convolutional neural network for bioactivity prediction in structure-based drug discovery. *arXiv preprint arXiv:1510.02855*, 2015.

Zhi-Jie Wang received the PhD degree in computer science from the Shanghai Jiao Tong University, Shanghai, China. He is currently an associate professor at the College of Computer Science, Chongqing University (CQU), Chongqing, China. His current research interests include data mining, artificial intelligence, databases, machine learning, etc. He has published a set of research papers in these fields including IEEE TKDE, IEEE TMM, IJCAI, AAAI, etc. He is a member of CCF, IEEE, and ACM.

Yan Guo received the bachelors degree from the College of food science & technology, Nanchang University, Nanchang, China. She is currently working toward the masters degree with the Department of Computer Science and Engineering, Hunan University, Changsha, China. Her main research interests include machine learning, and bioinformatics, etc. One of her research paper has been accepted by international conference on Bioinformatics & Biomedicine (BIBM'19).

Xuan Lin is currently a lecturer at the College of Computer Science, Xiangtan University, Xiangtan, China. Before joining Xiangtan University, he received the PhD degree in computer science from Hunan University, Changsha, China, in 2021. He was visiting scholar in University of Illinois at Chicago, from 2019 to 2020. His main research interests include machine learning, graph neural networks and bioinformatics. He has published several research papers in these fields including IEEE TKDE, Briefings in Bioinformatics, IJCAI, AAAI, ECAI, BIBM, etc.

Xiangxiang Zeng is an Yuelu distinguished Professor with the College of Information Science and Engineering, Hunan University, Changsha, China. Before joining Hunan University in 2019, he was with Department of Computer Science in Xiamen University. He received his Ph.D. degree in system engineering from Huazhong University of Science and Technology, China, in 2011. He was visiting scholar in Indiana University, The Chinese University of Hongkong, Oklahoma State University, etc. His main research interests include computational intelligence, graph neural networks and bioinformatics. He is a senior member of the IEEE.



Zhe Quan received the PhD degree in computer science from the University de Picardie Jules Verne, France. He is currently an associate professor at the College of Computer Science and Technology, Hunan University (HNU), Changsha, China. Before joining HNU, he worked at the National University of Defense Technology, Changsha, China, and worked as a postdoctoral research fellow at the Berkeley and Livermore Lab of the University of California, United Stated. His main research interests include machine learning, artificial intelligence, parallel and high-performance computing, etc. He has published a set of research papers in venues such as AAAI, IJCAI, ICSC, BIBM, etc.

Philip S Yu received the B.S. Degree in E.E. from National Taiwan University, the M.S. and Ph.D. degrees in E.E. from Stanford University, and the M.B.A. degree from New York University. He is a Distinguished Professor in Computer Science at the University of Illinois at Chicago and also holds the Wexler Chair in Information Technology. His research interest is on big data, including data mining, data stream, database and privacy. He has published more than 1,000 papers in refereed journals and conferences. He holds or has applied for more than 300 US patents. He received the ICDM 2013 10-year Highest-Impact Paper Award, and the EDBT Test of Time Award (2014). He is a Fellow of the ACM and the IEEE.