2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)

GraphCPI: Graph Neural Representation Learning for Compound-Protein Interaction

Zhe Quan[†], Yan Guo[†], Xuan Lin[†], Zhi-Jie Wang(✉)[‡,#,§], and Xiangxiang Zeng[†]
[†] College of Information Science and Engineering, Hunan University, Changsha, China
[‡] School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, China
[#] Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, China
[§] National Engineering Laboratory for Big Data Analysis and Applications, Beijing, China
{quanzhe, guoyan, jack_lin, xzeng}@hnu.edu.cn, wangzhij5@mail.sysu.edu.cn

***Abstract*—Accurately predicting compound-protein interactions (CPIs) is of great help to increase the efficiency and reduce costs in drug development. Most of existing machine learning models for CPI prediction often represent compounds and proteins in one-dimensional strings, or use the descriptor-based methods. These models might ignore the fact that molecules are essentially structured by the chemical bond of atoms. However, in real-world scenarios, the topological structure information 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 important for modeling compounds and proteins. Motivated by this, this paper suggests an end-to-end deep learning framework called *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 nerual networks for learning compounds, and it combines with a convolutional neural network for embedding sequences. We conduct extensive experiments based on two benchmark CPI datasets. The experimental results demonstrate that our proposed framework is feasible and also competitive, comparing against classic and state-of-the-art methods.**

***Index Terms*—Compound-Protein Interaction, Graph Neural Network, Representation Learning, Drug Discovery**

I. Introduction

Development of potential drugs for new targets/proteins is a costly and time-consuming process. Accurately identifying compound-protein interactions (CPIs) is a key task in pharmacology and drug discovery [1]. 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. Interaction between compound-protein pair indicates that compounds can have a positive or negative influence on functions triggered by proteins. This may affect the disease conditions.

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 [2], side effect prediction [3]. As a result, CPI prediction has received much attention in recent years. Traditional machine learning approaches for CPI prediction can be roughly classified into feature-based and similarity-based methods. Generally, feature-based methods construct input vector from descriptors of compounds and proteins, such as molecular docking and the 3D structure-embedded of protein, which are often difficult to obtain. On the other hand, similarity-based methods rely on hypothesis that compounds with similar structures should be of similar properties [4]. Recently, owing to the remarkable success in various machine learning tasks (e.g., natural language processing [5]), deep learning methods are also exploited for CPI prediction [6]. In this branch, existing methods consider either label/one-hot encodings or the fingerprint of molecules, they have not considered the chemical bond of atoms and the local chemical context of amino acids. However, in real-world scenarios, the topological structure information usually provides an overview of how the atoms are connected, 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 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 extract the local chemical context information for amino acids in proteins; in the process of extracting, we propose to incorporate *Prot2Vec* [7] to encode the amino acids to a distributed representation, which can efficiently avoid the limitation of the label/one-hot encodings of amino acids, since

978-1-7281-1867-3/19/$31.00 © 2019 IEEE

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-embedded 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* [8]. In a nutshell, the 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, 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 two 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 II reviews the related work. Section III introduces the proposed framework for CPI prediction task. Section IV covers and analyzes the experimental results. Finally, we conclude the paper in Section V.

II. Related Work

Compound-protein interactions (CPIs) prediction has been an interesting topic in drug discovery. Prior works focused either on simulation-based methods (e.g., descriptors), or on machine learning based models, which heavily rely on domain similarity. For example, with a variety of similarity information, Bleakley *et al*. [9] presented the bipartite local model (BLM) to predict CPIs, they trained local support vector machine (SVM) classifiers with the help of known interactions. Later, Mei *et al*. [10] improved BLM by exploiting the already known interactions of neighbors, which compensates the lack of BLM. 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.

Also, much attention has been devoted to applying deep learning techniques for drug-target interactions (DTIs) prediction (which is an alternative name of CPI prediction). For

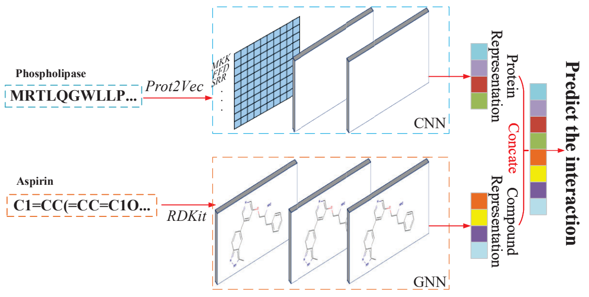


Fig. 1. The overview of GraphCPI. The top part illustrates a 3-layer CNN that learns representation for proteins, while the bottom part illustrates a 3-layer GNN that learns representation for compound.

example, Xie *et al*. [11] modeled DTI prediction as a binary classification and developed a deep-learning based model to predict potential interactions. Yasuo *et al*. [12] developed a method named CoDe-DTI that combines content-based and collaborative filtering to avoid some issues (e.g., lack of diversity and cold-start). Moreover, Wan *et al*. [13] 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. Recently, a model called DeepCPI was proposed for CPI perdition [8]. Among the studies in this line branch, DeepCPI 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 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 GNN to encode the atoms using Prot2Vec, and it uses pre-trained embedding 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.

III. The GraphCPI Framework

In this section, we firstly describe an overview of the proposed framework called *GraphCPI* (Section III-A), and then we present the representation learning for compounds and for proteins, respectively (Sections III-B $\sim$ III-C). Finally, we present the detail of CPI task prediction (Section III-D).

A. Overview of GraphCPI

Figure 1 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 string is encoded into a molecular graph, while the sequence of the protein is encoded into a distributed representation (like

Word2Vec [14]), forming a matrix. Later, the molecular graph is fed into graph neural networks (GNN) for capturing the structural information of the compound, while the matrix is fed into convolutional neural networks (CNN) to obtain local chemical context of the protein. As a result, we obtain two latent representation 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).

B. Graph Representation for Compounds

As we known, compounds are often expressed in the format of SMILES (Simplified Molecular-Input Line Entry System) [15]. 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 combine with the advanced embedding techniques for compounds and proteins. Specifically, for each input SMILES string of a chemical compound, we use RDKit [16] 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. In our method, we adopt 5 kinds of atom feature used in DeepChem (https://github.com/deepchem/deepchem). These atom properties constitute a binary feature vector with size 78. The first 44 entities (starting from 0) represent the atomic symbols, the 44-54 entities represent the atomicity, the 55-65 entities represent the total number of hydrogen, the 66-75 entities represent the implied value of atoms, and the last 2 entities represent whether atoms are aromatic. The list of the initial atom features is summarized in Table I.

Once such a molecular graph is obtained, one can fed it into any popular graph neural network model (e.g., GCN [17], GAT [18], GIN [19]) 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.

TABLE I
THE LIST OF INITIAL ATOM FEATURES

Atom Type	Description
Atomic symbols	C, N, O, S, F, Si, P ...
Atomicity	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
The total number of hydrogen	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
The implied value of atoms	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
whether atoms are aromatic	0, 1

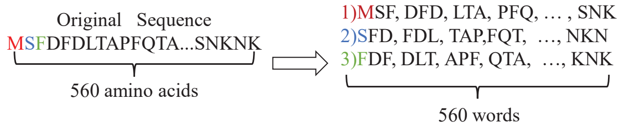

Original Sequence
MSFDFDLTAPFQTA...SNKNK
560 amino acids
1)MSF, DFD, LTA, PFQ, ... , SNK
2)SFD, FDL, TAP,FQT, ..., NKN
3)FDF, DLT, APF, QTA, ..., KNK
560 words

Fig. 2. Example of split protein sequence.

C. Sequence Representation for Proteins

Proteins are generally represented as a sequence of amino acids (e.g., ”MSFDFDLT...”). Similar to the SMILES string, we propose to first encode the amino acids into d-dimensional vector using *Prot2Vec* [7]. Since a single aimno acid often makes no sense, we adopt a fixed length N-gram splitting method to partition 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 method based on one-hot. Figure 2 illustrates an example of fixed-length N-gram splitting method. In this example, 560 amino acids can generate 560 words by N-gram splitting, while only 20 kinds of amino acids to be encoded by label encoding.

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 (https://www.uniprot.org/). One can set the fixed length to a specific value (e.g., 1000), then the sequence will be truncated when its length 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. The matrix is then fed into a CNN to obtain the local chemical context of the protein.

D. 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.

Generally, we concatenate two kinds of representations, and feed them to two fully-connected layers to output the interaction value. More precisely, for the GNN building block, we adopt three consecutive layers to update the node vectors in a molecular graph with regard to their neighbor nodes. For the CNN building block, we adopt three consecutive 1D-convolutional layers. Here the Rectified Linear Unit (ReLU) [20] 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 [8] as follows:

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

where Θ denotes the set of all weight matrices, bias vectors in our framework (e.g., GNN 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.

IV. Experiment

In this section, we first describe the experimental settings including datasets, evaluation metrics and baseline approaches (Section IV-A). Then, we compare our proposed framework GraphCPI with classic and state-of-the-art methods (Section IV-C).

A. Experimental Setup

Dataset. Following prior work [6], in our experiments we used two publicly available datasets called *human* and *C.elegans* 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.

In our experiments, we used a python library called *PubChemPy* (https://github.com/mcs07/PubChemPy) to obtain the SMILES format of compounds, and we extract the protein sequence from *Uniprot* (https://www.uniprot.org/uploadlists/).

In addition, since the ratio of positive and negative samples may affect the performance, we followed [6] and also used three different ratios (1:1, 1:3 and 1:5) to validate the performance. A more detailed description is summarized in Table II.

Evaluation metrics. We use three kinds of metrics, widely used in CPI task, to evaluate the performance. They are *precision*, *recall* and AUC [6]. Here, the AUC refers to the probability that a randomly chosen positive sample is ranked higher than a negative one [6]. It is computed as:

$$AUC = \frac{\sum I(P_{pos}, P_{neg})}{P * N} \tag{2}$$

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:

$$I(P_{pos}, P_{neg}) = \begin{cases} 1, P_{pos} > P_{neg} \\ 0.5, P_{pos} = P_{neg} \\ 0, P_{pos} < P_{neg} \end{cases} \tag{3}$$

Baseline methods. We compared our proposed framework against both classic and state-of-the-art methods. As for classic models, in our experiments we compared four traditional machine learning models[1], including *k-NN*, *random forest (RF)*, *L2-logistic (L2)*, and *SVM*. For these models, all parameter settings were kept as the same as in [6]. As for the state-of-the-art model, we directly compared a recently published model called *DeepCPI* [8]. 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). In DeepCPI, a GNN maps a graph $G = (V, E)$ to a vector $y \in R^d$ by two transition functions: (i) $v_i^{(t+1)} = \sigma(v_i^{(t)} + \sum_j h_{ij}^{(t)})$, and (ii) $e_{ij}^{(t)} = \sigma(e_{ij}^t + g_{ij}^t)$. Here 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 time step 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}, as shown in Eq. 4. The parameter g_{ij}^t is updated by node v_i^t and v_j^t, as shown in Eq. 5.

$$h_{ij}^{(t)} = f(W \begin{bmatrix} v_j^{(t)} \\ e_{ij}^{(t)} \end{bmatrix} + b) \tag{4}$$

$$g_{ij}^t = f(W(v_i^{(t)} + v_j^{(t)}) + b) \tag{5}$$

For ease of comparison, in our experiments all the above parameter settings were consistent with the original paper.

Implementation of our framework. As mentioned in Section III-B, once the molecular graph is obtained, one can feed it into any popular graphic 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* [17](https://github.com/tkipf/gcn): This approach introduces a graph Laplacian regularization and proposes a 2-layer Graph Convolutional Network (GCN) with the following layer-wise propagation rule: $H^{(l+1)} = \sigma(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$, where $\widetilde{A} = A + I_N$ is the adjacency matrix of the undirected graph G, I_N is the identity matrix,

TABLE II
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

[1]These models were obtained from http://admis.fudan.edu.cn/negative-cpi/.

TABLE III
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		

TABLE IV
COMPARISON RESULTS WITH CLASSIC AND STATE-OF-THE-ART METHODS ON HUMAN AND C.ELEGANS DATASETS, RESPECTIVELY.

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**
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
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
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**
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**
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**

$D_{ii} = \sum_j \widetilde{A}_{ij}$, $W^{(l)}$ is a layer-specific trainable weight matrix; $\sigma(.)$ denotes an activation function (*e.g.*, ReLU [21]), and $H(l) \in R^{N \times D}$ is the matrix of activations in the *l*-th layer.

(2) *GAT* [18](https://github.com/PetarV-/GAT): 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, $e_{ij} = \alpha(W\vec{h}_i, W\vec{h}_j)$, 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: $\alpha_{ij} = softmax_j(e_{ij})$. At last, a non-linearity σ is applied to compute the output node $\vec{h}_i^{'}$ as follows: $\vec{h}_i^{'} = \sigma(\sum_{j \in N_i} a_{ij} W\vec{h}_j)$.

(3) *GIN* [19](https://github.com/weihua916/powerful-gnns): 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: $h_v^{(k)} = MLP^{(k)}((1 + \epsilon_{(k)}) \cdot h_v^{(k-1)} + \sum_{u \in N(v)} h_u^{(k-1)})$, 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 [17] as *GraphCPI_GCN*. Similarly, we refer to other three methods integrating GAT [18], GIN [19], and GAT-GCN 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 implementation 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 (https://github.com/pytorch/pytorch) with Tensorflow [22] 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 III shows the main training parameters, while other omitted parameters were set to default values.

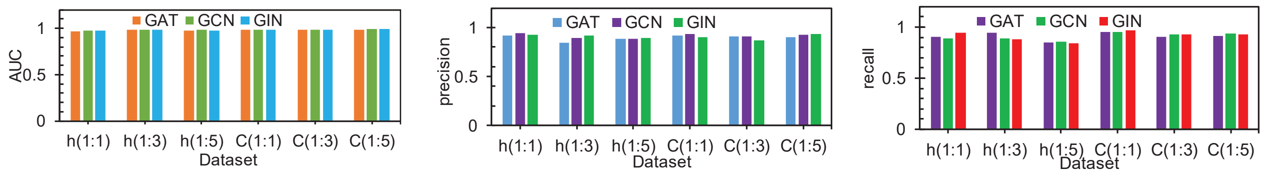
B. Stability of our framework

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 II). Figure 3 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 (AUC, precision, 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 × 3 imbalance ratios × 2 datasets), as shown in Figures 3(a)-3(c). In this regard, it indicates that the stability or robustness of our framework.

C. Comparing with classic and state-of-the-art methods

Table IV compares our proposed framework with classical and state-of-the-art methods. In general, *GraphCPI_GAT* 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 II), it performs not well on the larger dataset C.elegans, and this characteristic is even more obvious when the number of negative samples increases.



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

Fig. 3. 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.

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.

V. Conclusion

In this paper, we proposed an end-to-end framework, GraphCPI, for predicting compound-protein interaction. GraphCPI uses the graph representation for compounds and the embedding representation for proteins. Our framework can integrate any popular graph neural networks for learning compounds, and it combines with a convolutional neural network for embedding sequences of proteins. Experimental results based on benchmark datasets demonstrated its feasibility and competitiveness, compared with the competitors.

Acknowledgment

We thank anonymous reviewers very much for their efforts in evaluating our paper. This work was supported in part by the National Key R&D Program of China (2018YFB0204100), in part by the NSFC (No. 61602166, U1811264, U1501252, U1611264, U1711261, U1711262, U1711263, 61972425), in part by the Key R&D Program of Guangdong Province (No. 2018B010107005, 2019B010120001).

References

[1] 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.
[2] Zoya Khalid and Osman Ugur Sezerman. Prediction of hiv drug resistance by combining sequence and structural properties. *TCBB*, 15(3):966–973, 2018.
[3] Pengwei Hu, Zhu-Hong You, Tiantian He, Shaochun Li, Shuhang Gu, and Keith CC Chan. Learning latent patterns in molecular data for explainable drug side effects prediction. In *BIBM*, pages 1163–1169, 2018.
[4] Dhanya Sridhar, Shobeir Fakhraei, and Lise Getoor. A probabilistic approach for collective similarity-based drug–drug interaction prediction. *Bioinformatics*, 32(20):3175–3182, 2016.
[5] Zhe Quan, Zhi-Jie Wang, Yuquan Le, Bin Yao, Kenli Li, and Jian Yin. An efficient framework for sentence similarity modeling. *TASLP*, 27(4):853–865, 2019.
[6] 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.
[7] Ehsaneddin Asgari and Mohammad RK Mofrad. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one*, 10(11):e0141287, 2015.
[8] 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, 2018.
[9] Kevin Bleakley and Yoshihiro Yamanishi. Supervised prediction of drug–target interactions using bipartite local models. *Bioinformatics*, 25(18):2397–2403, 2009.
[10] 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.
[11] Lingwei Xie, Zhongnan Zhang, Song He, Xiaochen Bo, and Xinyu Song. Drug-target interaction prediction with a deep-learning-based model. In *BIBM*, pages 469–476, 2017.
[12] Nobuaki Yasuo, Nakashima Yusuke, and Masakazu Sekijima. Code-dti: Collaborative deep learning-based drug-target interaction prediction. In *BIBM*, pages 792–797, 2018.
[13] 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.
[14] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.
[15] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.
[16] Greg Landrum et al. Rdkit: Open-source cheminformatics, 2006.
[17] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.
[18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.
[19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.
[20] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.
[21] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.
[22] 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 *OSDI*, pages 265–283, 2016.