Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Original Papers

OXFORD

Data and text mining

Compound-protein Interaction Prediction with End-to-end Learning of Neural Networks for Graphs and Sequences

Masashi Tsubaki [1]*, Kentaro Tomii [1,2], and Jun Sese [1,2]

[1] National Institute of Advanced Industrial Science and Technology, Artificial Intelligence Research Center, Tokyo, 135-0064, Japan.
[2] AIST- Tokyo Tech Real World Big-Data Computation Open Innovation Laboratory, Tokyo, 152-8550, Japan

* To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: In bioinformatics, machine learning-based methods that predict the compound-protein interactions (CPIs) play an important role in the virtual screening for drug discovery. Recently, end-to-end representation learning for discrete symbolic data (e.g., words in natural language processing) using deep neural networks has demonstrated excellent performance on various difficult problems. For the CPI problem, data are provided as discrete symbolic data, i.e., compounds are represented as graphs where the vertices are atoms, the edges are chemical bonds, and proteins are sequences in which the characters are amino acids. In this study, we investigate the use of end-to-end representation learning for compounds and proteins, integrate the representations, and develop a new CPI prediction approach by combining a graph neural network (GNN) for compounds and a convolutional neural network (CNN) for proteins.
Results: Our experiments using three CPI datasets demonstrated that the proposed end-to-end approach achieves competitive or higher performance as compared to various existing CPI prediction methods. In addition, the proposed approach significantly outperformed existing methods on an unbalanced dataset. This suggests that data-driven representations of compounds and proteins obtained by end-to-end GNNs and CNNs are more robust than traditional chemical and biological features obtained from databases. Although analyzing deep learning models is difficult due to their black-box nature, we address this issue using a neural attention mechanism, which allows us to consider which subsequences in a protein are more important for a drug compound when predicting its interaction. The neural attention mechanism also provides effective visualization, which makes it easier to analyze a model even when modeling is performed using real-valued representations instead of discrete features.
Availability: https://github.com/masashitsubaki
Contact: tsubaki.masashi@aist.go.jp
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Identifying interactions between compounds and proteins is important in the discovery and development of safe and effective drugs (Keiser *et al.*, 2009). Revealing unknown compound-protein interactions (CPIs) is useful for the prediction of potential side effects and finding new uses for the existing drugs, i.e., drug repositioning (Lounkine *et al.*, 2012; Medina-Franco *et al.*, 2013). However, identifying CPIs experimentally is time-consuming and expensive.

To identify potential CPIs effectively, machine learning–based prediction methods have been developed from a chemogenomics perspective (Bredel and Jacoby, 2004), which considers the chemical space, genomic space, and its interactions in a unified framework. In line with this thinking, various types of compound and protein features and algorithms to predict CPIs have been investigated. For example,

Jacob and Vert (2008) used the tensor product–based features between chemical substructures and protein families and then applied SVMs with pairwise kernels. In addition, Yamanishi *et al.* (2008) proposed a bipartite graph learning method that maps compounds and proteins into a common feature vector space and minimized the Euclidean distances between vectors linked by known interactions (otherwise maximized). Furthermore, Bleakley and Yamanishi (2009) proposed a bipartite local model (BLM) that employs the similarity measurements between chemical structures and protein sequences and then applied SVMs with known interactions. Recently, to reduce the high dimensionality of the chemogenomics space, Cheng *et al.* (2012) applied feature selection techniques prior to training an SVM, and Tabei and Yamanishi (2013) focused on improving the prediction performance of a linear SVM using a minwise hashing algorithm to obtain compact fingerprints of compound-protein pairs. van Laarhoven *et al.* (2011) used Gaussian interaction profile (GIP) kernels based on a CPI network topology, and Gönen (2012) used kernelized Bayesian matrix factorization with twin kernels.

Recently, among various machine learning methods, deep neural networks (DNNs) have achieved excellent performance for various problems, such as speech recognition (Hinton *et al.*, 2012) and visual object recognition (Krizhevsky *et al.*, 2012). In particular, *end-to-end learning* is a powerful representation learning technique widely used in natural language processing problems, such as machine translation (Sutskever *et al.*, 2014). The end-to-end representation learning technique consists of three steps: (i) embedding discrete input symbols, such as words, in a low-dimensional real-valued vector space, (ii) designing various neural networks considering data structures (e.g., sequences and graphs), and (iii) learning all network parameters by backpropagation, including the embedding vectors of discrete input symbols. Note that this technique can be used for various discrete symbolic data. On the CPI problem, data are provided as discrete symbolic data, i.e., compounds are represented as graphs in which a vertex is an atom, an edge is a chemical bond, and proteins are represented as sequences in which a character is an amino acid. Based on these observations, there is room to consider end-to-end learning of CPI representations. While DNNs have been used for compounds and proteins (Tian *et al.*, 2016; Wan and Zeng, 2016; Hamanaka *et al.*, 2017), such DNN-based methods do not apply end-to-end representation learning and depend on molecular fingerprints and protein family databases as input features, which are *fixed* in the DNN training process.

In this paper, we investigate the use of end-to-end representation learning for compounds and proteins, integrate the representations, and develop a new CPI prediction approach. Specifically, we propose the use of graph neural networks (GNNs) (Scarselli *et al.*, 2009; Kearnes *et al.*, 2016) and convolutional neural networks (CNNs) (Kim, 2014), which can learn low-dimensional real-valued vector representations of molecular graphs and protein sequences (Sections 3 and 4). Note that most existing methods use fixed input features (i.e., binary values) in the training process of e.g., SVMs. We assume that this leads to poor performance on an unbalanced CPI dataset; Indeed, handling of an unbalanced dataset, i.e., a dataset that include small positive samples (i.e., interact) and large negative samples (i.e., not interact), is a common problem in CPI prediction (Tabei and Yamanishi, 2013), and the performance of existing methods is relatively poor (Liu *et al.*, 2015). On the other hand, end-to-end representation learning, which can learn input features (i.e., real values) in the DNN training process instead of fixing them, can potentially obtain data-driven features. Thus, end-to-end representation learning can benefit from a large training dataset regardless of the balance between positive and negative samples. That is, without any chemical and biological features, we expect to achieve more robust performance on both balanced and unbalanced datasets with end-to-end representation learning. Furthermore, the combination of a GNN and a CNN can naturally handle pairs of data with different structure, i.e., molecular graphs and protein sequences, and

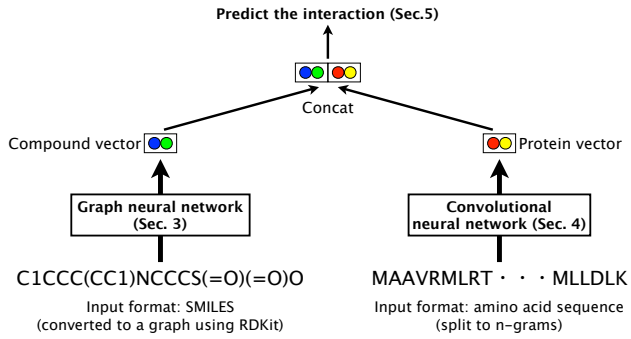

Fig. 1. An overview of the proposed CPI prediction approach. Compound and protein vectors, which are low-dimensional real-valued representations obtained using a GNN (Section 3) and a CNN (Section 4), are concatenated and input to a classifier (Section 5) to predict whether the compound and protein interact. The proposed method is based on end-to-end representation learning, which uses only raw inputs of compounds and proteins (i.e., SMILES and amino acid sequences) instead of features such as molecular fingerprints and protein domains extracted from databases. More precisely, we input molecular graphs obtained from SMILES pre-processed using the RDKit and split protein sequences based on n-gram amino acids to a GNN and CNN, respectively.

we can input them to a unified framework, i.e., end-to-end learning of chemogenomics representation space. Finally, we use the compound and protein representations, which have the same dimensionality, as input to a classifier to predict whether they interact (Section 5). An overview of the proposed CPI prediction method is shown in Fig. 1.

Although analyzing deep learning models is difficult due to their black-box nature, we demonstrate that this problem can be solved using a *neural attention mechanism* (Bahdanau *et al.*, 2014). This mechanism allows us to consider which subsequences in a protein are important for a drug compound to predict CPIs (i.e., interaction sites) by using weights, which are also learned in the proposed neural networks (Section 5.1). Furthermore, by using the obtained weights, the neural attention mechanism provides clear visualizations, which makes models easier to analyze (Fig. 9) even when modeling is performed using real-valued vector representations rather than discrete features.

In our experiments using three CPI datasets (Liu *et al.*, 2015; Mysinger *et al.*, 2012), we demonstrated that the proposed approach based on end-to-end learning of GNN and CNN can achieve competitive or higher performance than existing approaches: feature-based classical machine learning methods such as SVMs, other methods specifically for CPI prediction, a non-machine learning method (i.e., virtual screening and docking), and a recently proposed structure-based deep learning model. In particular, the proposed method significantly outperformed these methods on an unbalanced dataset; this indicates that the data-driven representations obtained by end-to-end learning of GNNs and CNNs are more robust than traditional chemical and biological features. We believe that the data-driven features can be more flexible compared to chemical and biological features that are fixed in the training process; this provides robust performance even with large, unbalanced training datasets. Furthermore, we visualized the interaction sites between a drug and the substructures of a protein. It is important to emphasize that we can obtain the information of 3D structural interaction sites from representation learning based on the information of 2D molecular graphs and 1D protein sequences.

2 Materials

Most datasets used to evaluate machine learning–based CPI prediction methods (Tian *et al.*, 2016; Wan and Zeng, 2016; Hamanaka *et al.*, 2017) include positive samples (i.e., interact) and randomly generated negative

samples (i.e., not interact). However, such random negative samples may include unknown positive samples. Even if a classifier is trained with such a dataset and achieves high performance, it may demonstrate poor performance with real test datasets. Based on this observation, screening true negative samples is important to create highly credible CPI datasets (Ding *et al.*, 2013).

First, we experimented with two sets of CPI datasets, for human and *C.elegans*, recently created by Liu *et al.* (2015). The datasets include highly credible negative samples of compound-protein pairs obtained by using a systematic screening framework, based on the assumption that "the proteins dissimilar to any known/predicted target of a given compound are not much likely to be targeted by the compound and vice versa (Liu *et al.*, 2015)." Positive samples of the datasets were retrieved from two manually curated databases: DrugBank 4.1 (Wishart *et al.*, 2007) and Matador (Günther *et al.*, 2007). In the end, the human dataset contains 3,369 positive interactions between 1,052 unique compounds and 852 unique proteins; the *C.elegans* dataset contains 4,000 positive interactions between 1,434 unique compounds and 2,504 unique proteins.

In the above screening framework, Liu *et al.* (2015) assumed that similar compounds interact with the proteins that are similar to known proteins. Then they defined the dissimilarity rules for proteins and drugs, and applied these rules to their screening framework to identify more reliable negative samples. In addition, it is also known that chemical compounds with similar (or different) features may interact with different (or similar) target proteins. Based on the observation, they considered that the number of predicted interactions between a target protein and negative compound candidates should be larger than some predefined threshold. By setting a threshold, they got compounds and proteins used to construct the negative samples. Furthermore, they expected that the features of proteins (compounds) interact with a specific compound (protein) should be different from each other as largely as possible. Since variance is a common measurement to evaluate data divergence, they carried out a statistical test to check whether the similarity variance of proteins (or compounds) corresponding to each compound (or protein) in the negative samples is larger than the population variance. Finally, they filtered out specious candidates and obtained credible negative samples of CPIs.

In the experiments of CPI prediction, since real-world CPI datasets are typically unbalanced, we evaluated the robustness of the compared methods using the unbalanced dataset. In our experiments, the ratios of positive and negative samples (positive:negative) were 1:1, 1:3, and 1:5, whereas the number of positive samples was fixed. This experimental setting was proposed by Tabei and Yamanishi (2013) and has been used in other studies. The negative samples used in our experiments were extracted from the top candidates based on the scores obtained by Liu *et al.* (2015). As CPI prediction is a classification problem, its performance is evaluated using the AUC, precision, and recall.

In addition to various machine learning methods, we also compared the proposed method with "non-machine learning" methods on the DUD-E dataset, an enhanced and rebuilt version of DUD, a directory of useful decoys (Mysinger *et al.*, 2012). The DUD-E benchmark, which is a challenging and robust dataset for structure-based virtual screening methods, contains 102 diverse target proteins (provided as PDB files), 22,886 active pairs of proteins and compounds (provided as SMILES), and its average is 224 compounds per target protein. In this paper, we followed the experimental setting of Wallach *et al.* (2015). We randomly divided 102 DUD-E targets into 72 targets as a training dataset and 30 targets as a test dataset. Note that we use only 1D protein sequences and 2D molecular graphs in the DUD-E dataset as our model does not require 3D information of proteins, although the DUD-E targets are provided as PDB files. Note moreover that we use the balanced dataset of DUD-E, i.e., the number of training samples is 22,886 active (i.e., positive) and 22,886 decoy (i.e., negative). The prediction performance is also evaluated using the AUC.

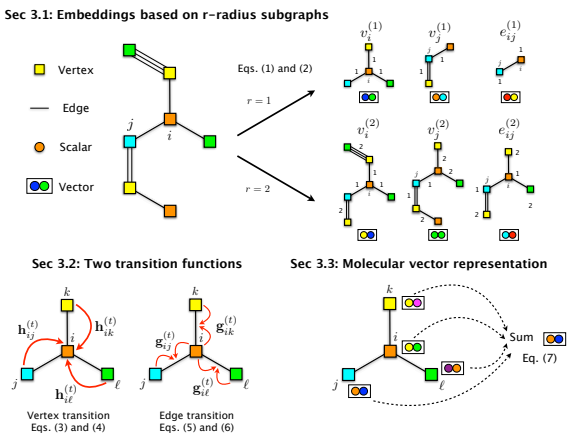

Fig. 2. Overview of the proposed GNN for molecular graph with subgraph-based vectors, two transition functions, and an output function. **Embeddings** (Section 3.1): we first consider the use of an r-radius subgraph (induced by neighboring vertices and edges within radius r from a vertex) to learn the representation. We embed the r-radius subgraphs of molecules in low-dimensional real-valued vector space. **Transition** (Section 3.2): we develop two transition functions in our GNN, i.e., vertex and edge transitions. The basic idea is that the local information of vertices and edges is propagated in the graph by (i) summing neighboring embeddings and (ii) iterating the process. **Output** (Section 3.3): we use the summation of the hidden vectors of vertices to obtain the output (i.e., a molecular vector representation).

3 Graph Neural Network for Molecular Graph

Notation. Throughout the paper, vectors are written in lowercase boldface letters (e.g., $\mathbf{v} \in \mathbb{R}^d$), matrices are written in uppercase boldface letters (e.g., $\mathbf{M} \in \mathbb{R}^{m \times n}$), and scalars and discrete symbols such as graphs, vertices, and edges are written in non-bold letters (e.g., $\mathcal{G}$, v, and e).

As shown in Fig. 1, we use a GNN for molecular graphs. In this section, we describe the GNN, which can obtain low-dimensional real-valued vector representations of molecular graphs. A GNN maps a graph $\mathcal{G}$ to a vector $\mathbf{y} \in \mathbb{R}^d$ with two functions, i.e., *transition* and *output* functions, originally proposed by Scarselli *et al.* (2009). Here, the transition function updates each vertex (i.e., an atom in a molecule) information in consideration of its neighboring vertices and edges (i.e., the chemical bonds in a molecule) in $\mathcal{G}$, and the output function maps the set of vertices to vector $\mathbf{y}$. Note that both functions are implemented using neural networks. In our GNN, a combined transition and output function $\mathbf{y} = f(\mathcal{G})$ is differentiable, and all parameters in f, including the input features, are learned by backpropagation. This is end-to-end learning and this differentiable function f requires invariance in symmetry and isomorphism for arbitrary sized, shaped, and complex structured graphs.

3.1 Input: embeddings based on r-radius subgraphs

A graph is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of edges. In a molecule, $v_i \in \mathcal{V}$ is the i-th atom and $e_{ij} \in \mathcal{E}$ is the chemical bond between the i-th and j-th atoms. Given a graph $\mathcal{G}$, we first embed all atoms and chemical bonds in a d-dimensional real-valued vector space in consideration of these types. However, in molecules, there are very few types of atoms (e.g., hydrogen and carbon) and chemical bonds (e.g., single and double); thus, representation learning is ineffective due to the small number of learning parameters in the model.

To address this problem, we use r-radius subgraphs (Costa and De Grave, 2010), which are induced by the neighboring vertices and edges within radius r from a vertex. This r value is also assumed to equal the number of hops from a vertex. More precisely, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we represent a set of all neighboring vertex indices within radius r from the i-th vertex as $\mathcal{N}(i, r)$. Note that $\mathcal{N}(i, 0) = \{i\}$. Then, we define the r-radius subgraph for vertex v_i as follows:

$$v_i^{(r)} = \left(\mathcal{V}_i^{(r)}, \mathcal{E}_i^{(r)}\right), \tag{1}$$

where

$$\mathcal{V}_i^{(r)} = \{v_j \mid j \in \mathcal{N}(i, r)\},$$
$$\mathcal{E}_i^{(r)} = \{e_{mn} \in \mathcal{E} \mid (m, n) \in \mathcal{N}(i, r) \times \mathcal{N}(i, r-1)\}.$$

In this paper, we refer to $v_i^{(r)}$ as the *r-radius vertex*. For example, CH_3-CHO (acetaldehyde) contains 1-radius vertices, e.g., CH_3-C and C-CHO. In addition, we define the r-radius subgraph for edge e_{ij} as follows:

$$e_{ij}^{(r)} = \left(\mathcal{V}_i^{(r-1)} \cup \mathcal{V}_j^{(r-1)}, \mathcal{E}_i^{(r)} \cap \mathcal{E}_j^{(r)}\right). \tag{2}$$

In this paper, we refer to $e_{ij}^{(r)}$ as the *r-radius edge*. For example, CH_3-CHO contains 1-radius edges, e.g., C-H and C=O. Then, for each type of r-radius vertex and r-radius edge, we assign an embedding (i.e., vector) depending on the type, i.e., $\mathbf{v}_i^{(r)} \in \mathbb{R}^d$ and $\mathbf{e}_{ij}^{(r)} \in \mathbb{R}^d$, which is randomly initialized and subsequently trained by backpropagation during supervised learning (Section 5.2). The left of Fig. 2 shows examples of r-radius vertices, r-radius edges, and their assigned embeddings.

For simplicity, vertex embedding $\mathbf{v}_i = \mathbf{v}_i^{(0)}$ and edge embedding $\mathbf{e}_{ij} = \mathbf{e}_{ij}^{(0)}$ are used to describe our GNN procedure in the following for explanatory purposes, and embeddings based on r-radius subgraphs are used for our implementation and experiments (Sections 6 and 7).

3.2 Two transition functions in GNN

Here, we first describe the transition function for vertices (i.e., vertex transition) in our GNN. In addition, we consider the transition function for edges (i.e., edge transition) because edges are also represented by vector embeddings, as described in Section 3.1.

Vertex transition: Given a graph $\mathcal{G}$ and the randomly initialized embeddings of vertices and edges, we represent the i-th vertex embedding at time step t as $\mathbf{v}_i^{(t)} \in \mathbb{R}^d$. We then update $\mathbf{v}_i^{(t)}$ using the following transition function:

$$\mathbf{v}_i^{(t+1)} = \sigma\left(\mathbf{v}_i^{(t)} + \sum_{j \in \mathcal{N}(i)} \mathbf{h}_{ij}^{(t)}\right), \tag{3}$$

where σ is the element-wise sigmoid function: $\sigma(x) = 1/(1 + e^{-x})$, $\mathcal{N}(i)$ is the set of neighboring indices of i, and $\mathbf{h}_{ij}^{(t)} \in \mathbb{R}^d$ is the hidden neighborhood vector. This hidden vector can be computed by considering the neighboring vertex v_j and edge e_{ij} using the following neural network:

$$\mathbf{h}_{ij}^{(t)} = f\left(\mathbf{W}_{neighbor}\begin{bmatrix}\mathbf{v}_j^{(t)} \\ \mathbf{e}_{ij}^{(t)}\end{bmatrix} + \mathbf{b}_{neighbor}\right), \tag{4}$$

where f is a non-linear activation function such as ReLU (LeCun *et al.*, 2015): $f(x) = \max(0, x)$, $\mathbf{W}_{neighbor} \in \mathbb{R}^{d \times 2d}$ is the weight matrix, $\mathbf{b}_{neighbor} \in \mathbb{R}^d$ is the bias vector, and $\mathbf{e}_{ij}^{(t)} \in \mathbb{R}^d$ is the edge embedding between the i-th and j-th vertices at time step t. Thus, by summing the neighboring hidden vectors and iterating them over time steps, vertex embeddings can gradually gather more global information on the graph.

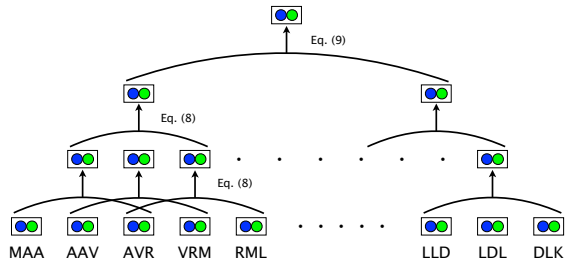

Fig. 3. Overview of our CNN for protein sequence with hierarchical filter functions.

Edge transition: The above iterative procedure can also be applied to edge embeddings in a similar manner. Here, we update $\mathbf{e}_{ij}^{(t)}$ using both side vertex embeddings $\mathbf{v}_i^{(t)}$ and $\mathbf{v}_j^{(t)}$ as follows:

$$\mathbf{e}_{ij}^{(t+1)} = \sigma\left(\mathbf{e}_{ij}^{(t)} + \mathbf{g}_{ij}^{(t)}\right), \tag{5}$$

$$\mathbf{g}_{ij}^{(t)} = f\left(\mathbf{W}_{side}\left(\mathbf{v}_i^{(t)} + \mathbf{v}_j^{(t)}\right) + \mathbf{b}_{side}\right), \tag{6}$$

where $\mathbf{W}_{side} \in \mathbb{R}^{d \times d}$ is the weight matrix and $\mathbf{b}_{side} \in \mathbb{R}^d$ is the bias vector. Note that we compute $\mathbf{v}_i^{(t)} + \mathbf{v}_j^{(t)}$ in Eq. (6) because edges are undirected in a molecular graph, e.g., C=O and O=C. In other words, C=O and O=C are identical, and these vectors are the same representations.

Thus, with the vertex and edge transition functions, both embeddings are considered equally and updated simultaneously in our GNN. The right of Fig. 2 illustrates both transition functions.

3.3 Output: molecular vector representation

To obtain the final output $\mathbf{y}_{molecule} \in \mathbb{R}^d$ from the set of vertex vectors obtained by the transition function, i.e., $\mathbf{V} = \{\mathbf{v}_1^{(t)}, \mathbf{v}_2^{(t)}, \cdots, \mathbf{v}_{|\mathcal{V}|}^{(t)}\}$, we use the average of the vertex vectors as follows:

$$\mathbf{y}_{molecule} = \frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}\mathbf{v}_i^{(t)}, \tag{7}$$

where $|\mathcal{V}|$ is the number of vertices in the molecular graph. While this is the simplest operation to obtain a molecular vector, we propose another output function considering a protein to interact (Section 5).

4 Convolutional Neural Network for Protein Sequence

As shown in Fig. 1, we use a CNN for protein sequences. In this section, we describe the CNN, which can obtain low-dimensional real-valued vector representations of protein sequences. CNNs for sequences map a sequence $\mathcal{C}$ to a vector $\mathbf{y} \in \mathbb{R}^d$ with multiple filter functions. Note that the dimensionality d of protein sequences is the same as that of the molecular graphs described in Section 3.3. Using a filter function, our CNN computes a hidden vector from the subsequences of $\mathcal{C}$ (input features) and weight matrix (learning parameter), applies the filter function in a hierarchical manner, and then produces the output vector $\mathbf{y}$. Note that all filter functions are implemented by neural networks. In our CNN, the total function $\mathbf{y} = f(\mathcal{C})$ is differentiable and all parameters in f, including the input features, are learned by backpropagation. Similar to our GNN, this CNN is also end-to-end learning.

4.1 Input: embeddings based on n-gram amino acids

To apply the CNN to proteins, we first define "words" in protein sequences as n-gram amino acids (Dong *et al.*, 2006). We then split the protein sequences into overlapping n-gram amino acids. Since there are 20 types of amino acids, the total number of possible n-grams is 20^n. In this study, to keep the vocabulary size tractable and avoid low-frequency words in the learning representations, we set a relatively small n-gram number $n = 3$. For example, we can split a protein sequence into an overlapping 3-gram amino acid sequence as follows: *MAAVRM* $\cdots$ *LDLK* $\rightarrow$ *"MAA", "AAV", "AVR",* $\cdots$ *, "LDL", "DLK."*

Given a protein sequence $\mathcal{S} = x_1, x_2, \cdots, x_{|\mathcal{S}|}$, where x_i is the i-th word and $|\mathcal{S}|$ is the sequence length, we first translate all words to randomly initialized embeddings, which we refer to as "word embeddings." Here, let the sequence of word embeddings be:

$$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \cdots, \mathbf{x}_{|\mathcal{S}|-1}, \mathbf{x}_{|\mathcal{S}|},$$

where $\mathbf{x}_i \in \mathbb{R}^d$ is the d-dimensional embedding of the i-th word. Alternatively, we can also consider a sequence whose elements comprise concatenated word embeddings. For example, the sequence composed of a concatenation of three contiguous embeddings is as follows:

$$[\mathbf{x}_1; \mathbf{x}_2; \mathbf{x}_3], [\mathbf{x}_2; \mathbf{x}_3; \mathbf{x}_4], \cdots, [\mathbf{x}_{|\mathcal{S}|-2}; \mathbf{x}_{|\mathcal{S}|-1}; \mathbf{x}_{|\mathcal{S}|}],$$

where $[\mathbf{x}_i; \mathbf{x}_{i+1}; \mathbf{x}_{i+2}] \in \mathbb{R}^{3d}$ is the concatenation of $\mathbf{x}_i$, $\mathbf{x}_{i+1}$, and $\mathbf{x}_{i+2}$. Here, $\mathbf{x}_{i:i+w-1}$ refers to $[\mathbf{x}_i; \cdots; \mathbf{x}_{i+w-1}]$, where w is the window size to be concatenated. In the proposed approach, such a sequence is input to the CNN.

4.2 Filter function in CNN

Our CNN uses a filter function, where the input is $\mathbf{x}_{i:i+w-1} = \mathbf{c}_i^{(0)} \in \mathbb{R}^{dw}$ and the output is a hidden vector $\mathbf{c}_i^{(1)} \in \mathbb{R}^d$ expressed as follows:

$$\mathbf{c}_i^{(1)} = f(\mathbf{W}_{conv}\mathbf{c}_i^{(0)} + \mathbf{b}_{conv}), \tag{8}$$

where f is a non-linear activation function (e.g., ReLU), $\mathbf{W}_{conv} \in \mathbb{R}^{d \times dw}$ is the weight matrix, and $\mathbf{b}_{conv}$ is the bias vector. Note that this filter function allows us to obtain a d-dimensional hidden vector from a dw-dimensional input vector. As a result, we can apply the function hierarchically, i.e., we compute the t-th hidden vector as $\mathbf{c}_i^{(t)} = f(\mathbf{W}_{conv}\mathbf{c}_i^{(t-1)} + \mathbf{b}_{conv})$ to obtain a set of hidden vectors: $\mathbf{C} = \left\{\mathbf{c}_1^{(t)}, \mathbf{c}_2^{(t)}, \cdots, \mathbf{c}_{|\mathcal{C}|}^{(t)}\right\}$. Fig. 3 illustrates the filter functions in our CNN.

4.3 Output: protein vector representation

To obtain the final output $\mathbf{y}_{protein} \in \mathbb{R}^d$ from the set of hidden vectors $\mathbf{C} = \{\mathbf{c}_1^{(t)}, \mathbf{c}_2^{(t)}, \cdots, \mathbf{c}_{|\mathcal{C}|}^{(t)}\}$, we use the average of $\mathbf{C}$ as follows:

$$\mathbf{y}_{protein} = \frac{1}{|\mathcal{C}|}\sum_{i=1}^{|\mathcal{C}|}\mathbf{c}_i^{(t)}. \tag{9}$$

Note that $\mathbf{y}_{protein}$ has the same dimensionality as $\mathbf{y}_{molecule}$ (Section 3). While this is the simplest operation to obtain a protein vector, in Section 5, we propose another output function considering a molecule to interact.

5 Compound-protein Interaction Prediction

In this section, using the obtained vector representations of compounds and proteins (Sections 3.3 and 4.3, respectively), we propose a CPI prediction model. In the proposed model, we consider the interactions between a compound and the subsequences in a protein sequence. To model such interactions, we use the recently proposed *neural attention mechanism*, which is widely used in machine learning (Bahdanau *et al.*, 2014). We describe this mechanism, the classifier, and the training process in the following.

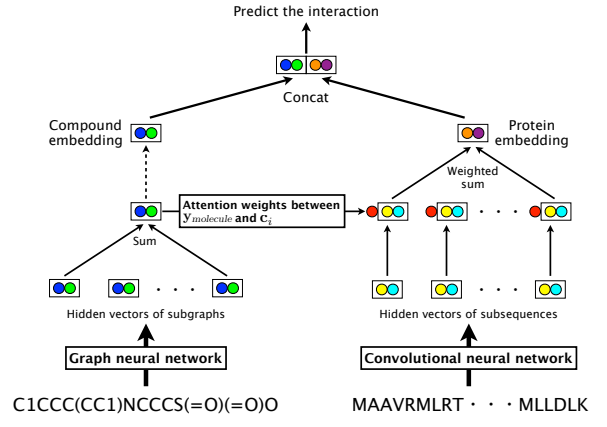

Fig. 4. CPI prediction model using weighted sums of the compound and protein embeddings with the neural attention mechanism, which considers interactions between subgraphs in a molecular graph and subsequences in a protein sequence.

5.1 Capturing interaction sites between compound and protein with neural attention mechanism

Given a molecular vector $\mathbf{y}_{molecule}$ and a set of hidden vectors of subsequences in a protein $\mathbf{C} = \left\{\mathbf{c}_1^{(t)}, \mathbf{c}_2^{(t)}, \cdots, \mathbf{c}_{|\mathcal{C}|}^{(t)}\right\}$, we wish to weight for $\mathbf{c}_i^{(t)}$ considering $\mathbf{y}_{molecule}$. In other words, we compute which subsequences in the protein are more important for the molecule by assigning greater weights to the subsequences. Such weights can be modeled using neural networks, i.e., the neural attention mechanism (Bahdanau *et al.*, 2014).

More precisely, given $\mathbf{y}_{molecule}$ and $\mathbf{c}_i^{(t)}$, we compute the following dot product–based scalar values as the weights:

$$\begin{aligned}
\mathbf{h}_{molecule} &= f(\mathbf{W}_{inter}\mathbf{y}_{molecule} + \mathbf{b}_{inter}),\\
\mathbf{h}_i &= f(\mathbf{W}_{inter}\mathbf{c}_i + \mathbf{b}_{inter}),\\
\alpha_i &= \sigma\left(\mathbf{h}_{molecule}^{\top}\mathbf{h}_i\right),
\end{aligned}$$

where $\mathbf{W}_{inter}$ is the weight matrix and $\mathbf{b}_{inter}$ is the bias vector. The weight value α_i, which is called attention, can be assumed to represent the interaction strength between a molecule and the subsequence of a protein. Using the attention weights, we obtain the weighted sum of $\mathbf{h}_i$ as follows:

$$\mathbf{y}_{protein} = \sum_{i=1}^{|\mathcal{C}|}\alpha_i\mathbf{h}_i. \tag{10}$$

Thus, using the attention mechanism, we have the flexibility to model the interactions between compounds and proteins rather than obtaining a simple summation. Fig. 4 illustrates the attention mechanism for CPI prediction. Note that the attention mechanism allows us to analyze CPIs by visualizing the weight values (Section 7.2, Figs. 9).

5.2 Classifier

We concatenate $\mathbf{y}_{molecule}$ and $\mathbf{y}_{protein}$, i.e., $[\mathbf{y}_{molecule}; \mathbf{y}_{protein}]$, and obtain an output vector $\mathbf{z} \in \mathbb{R}^2$, which is the input to the CPI classifier:

$$\mathbf{z} = \mathbf{W}_{output}[\mathbf{y}_{molecule}; \mathbf{y}_{protein}] + \mathbf{b}_{output},$$

where $\mathbf{W}_{output} \in \mathbb{R}^{2\times 2d}$ is the weight matrix and $\mathbf{b}_{output} \in \mathbb{R}^2$ is the bias vector. Finally, a softmax layer is added on top of the output vector $\mathbf{z} = [y_0, y_1]$ to model the CPI probability as follows:

$$p_t = \frac{\exp(y_t)}{\sum_i \exp(y_i)},$$

where $t \in \{0, 1\}$ is the binary label (i.e., interact or not) and p_t is the probability of t. Thus, we use a softmax classifier for CPI prediction.

5.3 Training

Given a set of all compound-protein pairs and the labels in a training dataset, the training objective is to minimize the loss function $\mathcal{L}$, given as the cross-entropy loss as follows:

$$\mathcal{L}(\Theta) = -\sum_{i=1}^{N} \log p_{t_i} + \frac{\lambda}{2}||\Theta||_2^2,$$

where Θ is the set of all weight matrices and bias vectors in our GNN, CNN, the embeddings of r-radius vertices and edges, and the embeddings of n-gram amino acids, and N is the total number of molecule-protein pairs in the training dataset, t_i is the i-th label, and λ is an L2 regularization hyper-parameter. Then, we use backpropagation to train Θ.

6 Experiments

6.1 Implementation and training

Our GNN takes SMILES as input, which is a string encoding of a molecule. Note that SMILES was converted to a graph representation using RDKit. We then extracted various information of the molecular graph, such as atom types, chemical bonds, and the adjacency list of atoms. For proteins, the pre-processing is not required because our CNN takes a raw amino acid sequence as input.

We implemented our GNN and CNN using Chainer (version 3.2.0) (Tokui *et al.*, 2015), and the training details of these neural networks are as follows: optimization: ADAM (Kingma and Ba, 2014), which is one of the SGD-based algorithms; radius r: 0 (i.e., each atom and chemical bond), 1, or 2; n-gram: 1 (i.e., 20 amino acids), 2, or 3; window size: 11 (fixed); vector dimensionality of vertices, edges, and n-grams: 5, 10, 20, and 30; number of time steps (i.e., depth) in GNN: 2, 3, or 4; number of layers (i.e., depth) in CNN: 2, 3, or 4; regularization λ: 1e-5, 1e-6, and 1e-7. Note that the batch size is 1; a batch contains a molecule, and the molecule contains a relatively large number of vertices and edges. In addition, using the batch size 1, we achieved the best performance in terms of the convergence of accuracy.

We perform grid search over a combination of the above hyperparameters using five-fold cross-validation. While deep learning models generally require careful hyperparameter tuning, we can obtain high performance with a relatively small range about hyperparameter tuning. In Section 6, We analyzed the effects of these hyperparameters relative to improving CPI prediction performance.

6.2 Main results: AUC, precision, and recall on balanced and unbalanced datasets

Tables 1 and 2 show the AUC, precision, and recall results of various traditional machine learning methods and the proposed method obtained using the balanced and unbalanced datasets from Liu *et al.* (2015). First, on the balanced and small dataset (i.e., positive:negative is 1:1), the proposed method achieved lower (on the human dataset) or higer (on the *C.elegans* dataset) performance compared to other methods: k-NN, random forest (RF), L2-logistic (L2), and SVM, which the results are obtain in Liu *et al.* (2015) and the experimental settings are as follows: k-NN and RF were run by Weka 3.7, L2 was run by liblinear 1.94, and the SVM was run by libsvm 3.17. Each classifier was learned by default setting of each software, e.g., SVM uses the radial basis function (RBF) as a non-linear kernel function. On the other hand, on the unbalanced and large dataset (i.e., positive:negative is 1:5), the proposed method outperformed the existing methods on both human and C.elegans datasets. This suggests that the proposed method is robust even if the dataset is unbalanced and when using a large dataset. This also implies that the end-to-end learned representations obtained using the GNN and CNN, i.e., the data-driven features, can reflect the useful properties of compounds and proteins for predicting interactions. Note that the proposed GNN and CNN do not rely on any chemical and biological features of molecular graphs and protein sequences used in the existing methods, such as PubChem fingerprints and Pfam domains.

Measure	Negative ratio	k-NN	RF	L2	SVM	Ours
AUC	1	0.860	0.940	0.911	0.910	**0.970**
	3	0.904	**0.954**	0.920	0.942	0.950
	5	0.913	0.967	0.920	0.951	**0.970**
Precision	1	0.798	0.861	0.891	**0.966**	0.923
	3	0.716	0.847	0.837	**0.969**	0.949
	5	0.684	0.830	0.804	**0.969**	**0.969**
Recall	1	0.927	0.897	0.913	**0.950**	0.918
	3	0.882	0.824	0.773	0.883	**0.913**
	5	0.844	0.825	0.666	0.861	**0.975**

Table 1. Main results on the human dataset: AUC, precision, and recall of k-NN, random forest (RF), L2 logistic (L2), SVM, and the proposed method on the balanced and unbalanced datasets created by Liu et al. (2015). The existing methods use features based on PubChem fingerprints and Pfam domains. The proposed method uses features obtained by end-to-end learning of GNNs and CNNs, i.e., data-driven feature representations of compounds and proteins. The above 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 number of layers in CNN is 3.

Measure	Negative ratio	k-NN	RF	L2	SVM	Ours
AUC	1	0.858	0.902	0.892	0.894	**0.978**
	3	0.892	0.926	0.896	0.901	**0.971**
	5	0.897	0.928	0.906	0.907	**0.971**
Precision	1	0.801	0.821	0.890	0.785	**0.938**
	3	0.787	0.836	0.875	0.837	**0.916**
	5	0.774	0.830	0.863	0.896	**0.920**
Recall	1	0.827	0.844	0.877	0.818	**0.929**
	3	0.743	0.705	0.681	0.576	**0.921**
	5	0.690	0.639	0.582	0.519	**0.836**

Table 2. Main results on the *C.elegans* dataset. The experimental setting is the same as that of above human dataset.

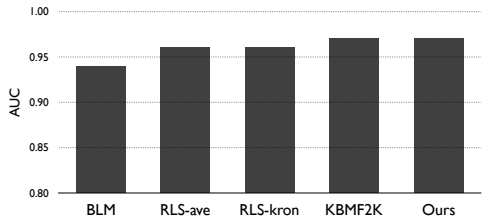

Fig. 5. The AUC scores of various methods specifically for CPI prediction on the human dataset. We achieved higher or competitive AUC score.

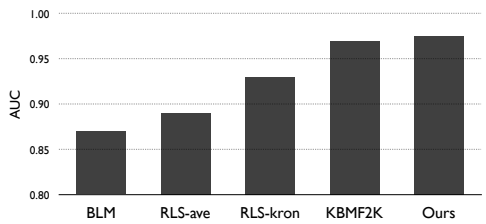

Fig. 6. The AUC scores of various methods specifically for CPI prediction on the *C.elegans* dataset. We achieved higher or competitive AUC score.

6.3 Comparison with various methods specifically for CPI prediction

In additional experiments, we compared the proposed method to other existing methods specifically for CPI prediction, i.e., BLM Bleakley and Yamanishi (2009), the RLS-avg and RLS-Kron classifiers with GIP kernels van Laarhoven *et al.* (2011), and KBMF2K-classification and KBMF2K-regression Gönen (2012), which were run using the same experimental settings as Liu *et al.* (2015). Note that these methods take a chemical structure similarity matrix, protein sequence similarity matrix, and CPI matrix as inputs. Figs. 5 and 6 show the AUC scores on the human and *C.elegans* datasets. As can be seen, on both datasets, the proposed method also achieved competitive or greater performance compared to these methods.

6.4 Comparison with non-machine learning methods and other deep learning models on the DUD-E dataset

As the final experiment, we compared with some non-machine learning methods and recently proposed deep learning models. In this experiment, we used the DUD-E benchmark, according to the experimental setting of Wallach *et al.* (2015) as described in Section 2. We compared our model with AutoDock Vina (Trott and Olson, 2010) and Smina (Koes *et al.*, 2013) as the non-machine learning methods, and AtomNet (Wallach *et al.*, 2015) and 3D-CNN (Ragoza *et al.*, 2017) as the deep learning models. AutoDock Vina is an open-source molecular docking program of small ligands to target proteins, and Smina is also a molecular docking program, which is a fork of the Autodock Vina and adds better control of scoring function and minimization. AtomNet (Wallach *et al.*, 2015) is a 3D structure-based CNN, which incorporates 3D information of proteins and compounds, for predicting the bioactivity of small molecules for drug discovery. AtomNet combines the information of ligand and the structure of target protein, which requires the coordinates of each atom in the binding site of the target protein. 3D-CNN (Ragoza *et al.*, 2017) also a 3D structure-based CNN, which discretizes a protein-ligand structure into a grid, incorporates 3D information of protein and ligand based on the grid, and predicts the interaction. This is similar to the CNNs that take images as inputs, where a scene is discretized into pixels with red, green, and blue values (RGB). When we evaluated our proposed model using 72 targets as a training dataset and 30 targets as a test dataset based on the DUD-E dataset, we found that, as shown in Fig. 7 in terms of the AUC scores, our model achieved greater performance compared to these methods.

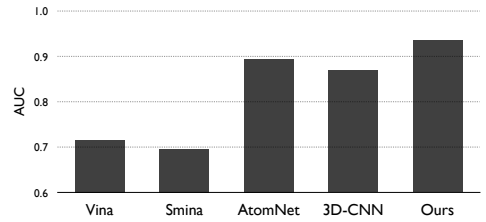

Fig. 7. The AUC scores of AutoDock Vina, Smina, AtomNet, 3D-CNN, and our proposed model. The proposed method, which requires only 2D molecular graphs and 1D protein sequences as inputs, achieved higher AUC score compared to the four methods, which require 3D information of proteins and compounds, on the DUD-E dataset. Note that the AUC scores of AutoDock Vina, Smina, AtomNet, and 3D-CNN are derived from Wallach et al. (2015) and Ragoza et al. (2017).

Note that this is not a complete comparison due to the difference in terms of input information, i.e., Vina, Smina, AtomNet, and 3D-CNN use 3D structure information of proteins, whereas we use only 1D sequence information of proteins. We believe that our approach, based on only protein sequences, is reasonable for applications such as large-scale CPI prediction and screening.

7 Discussion and Conclusion

7.1 Model analysis

In this section, we analyze the effects of the model hyperparameters on CPI prediction performance using learning curves, where the x-axis is the epoch and the y-axis is the AUC obtained with the test data (Fig. 8). To describe the learning curves, we use the DUD-E dataset. In addition, we visualize CPI sites by highlighting high-value weights obtained by the neural attention mechanism (Fig. 9).

Effects of r-radius subgraphs and n-gram amino acids: As shown in the first learning curve in Fig. 8, when $r = 0$ and $n = 1$, i.e., when we use atoms, chemical bonds, and amino acids instead of r-radius vertices, edges, and n-gram amino acids, these vector representations cannot be learned correctly. Note that the AUC score with $r = 0$ and $n = 1$ is about 0.8; this is lower than the AUC score of 3D-CNN (see Fig. 7). On the other hand, $r = 2$ radius vertices, edges, and $n = 3$ gram amino acids can improve the performance significantly; the AUC score is about 0.95 and this is higher than that of 3D-CNN. Interestingly, among the various components of our GNN and CNN, r and n values are the most important hyperparameters; that is, subgraph- and subsequence-based representation learning is crucial relative to improving CPI prediction performance. As shown in the learning curves, high performance is achieved early in the training process.

Effect of vector dimensionality: As shown in the second learning curve in Fig. 8, high AUC scores were obtained when relatively low-dimensional vectors for compounds and proteins were used. In addition, higher dimensional vectors did not yield further improvement. Surprisingly, when we use extremely low-dimensional (five-dimensional) vectors, we achieved high AUC scores. These results show that end-to-end representation learning–based CPI prediction can be modeled using small dimensional neural networks.

Effect of number of time steps/layers in GNN/CNN: As shown in the third learning curve in Fig. 8, the effect of the number of time-steps/layers in the GNN/CNN is relatively small, i.e., the depths of these two neural

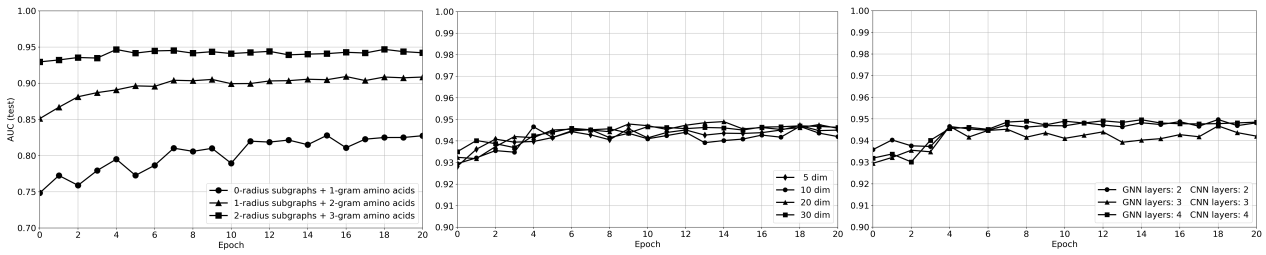

Fig. 8. Learning curves with various hyperparameters on the DUD-E dataset. In all learning curves, unless otherwise noted, we use the following hyperparameters: r-radius is 2, n-gram is 3, window size is 11, vector dimensionality is 10, number of time steps in GNN is 3, and number of layers in CNN is 3.

networks are not crucial to improving performance. Indeed, in quantum chemistry research, Gilmer *et al.* (2017) showed that the number of time steps in a GNN is not significant for predicting the quantum mechanical properties of molecules. In addition, Schütt *et al.* (2017) showed that the performance of a GNN is saturated at time step 3 when predicting molecular energies. As shown by the third learning curve in Fig. 8, we agree with these findings and draw the same conclusions. Furthermore, since we consider the global information of graphs and sequences using r-radius subgraphs and n-gram subsequences at the point of inputs in the GNN and CNN, the depths of the neural networks may not be required.

7.2 Visualization of CPIs with attention weights

The goals of this study were to achieve high performance with deep learning and demonstrate that deep learning can be analyzed even with a black-box model (i.e., modeling with real-valued vectors). One of the most interesting aspects of the neural attention mechanism (Section 5.1) is that the obtained attention weights allow us to gain greater insight into the output and interpret what the model has learned from the data. In other words, the neural attention mechanism can provide a helpful clue to considering which regions in a protein are important for interactions between a drug compound and a protein by highlighting high-value attention weights.

To exemplify this, we visualize such regions with high weight values calculated from the neural attention mechanism by mapping them onto a known 3D protein structure. Here, Fig. 9 shows examples of interactions between a drug compound and a protein, which are highlighted with the weights obtained by the neural attention mechanism. Fig. 9(a) shows the complex of imatinib and Syk (spleen tyrosine kinase) (PDB ID: 1XBB (Atwell *et al.*, 2004)), and Fig. 9(b) shows the complex of aspirin and group II phospholipase A2 (PDB ID: 1TGM). Note that hyperparameters used to describe these figures are as follows: r-radius: 2; n-gram: 3; window size: 11; vector dimensionality: 10; number of time steps in GNN: 3; and number of layers in CNN: 3. In both cases, regions with weights higher than 0.8 overlap substantially with the interaction sites between a drug compound and a protein. In the case of 1XBB (Fig. 9(a)), there are four such regions. Among them, two regions are located in the binding sites including residues which form hydrogen bonds and van der Waals interactions with imatinib. Another one contains two phosphorylation sites, S579 and T582, although the role of those residues is not obvious. In the case of 1TGM (Fig. 9(b)), there are two such regions which almost correspond to the part of ligand-recognition site (Singh *et al.*, 2007), i.e., subsites 1 and 2 including L2, A18, and I19 that form van der Waals interactions with aspirin.

In addition, we also examined whether the neural attention mechanism could capture the promiscuous domain using the human urokinase-type plasminogen activator (u-PA) catalytic domain. We found that the

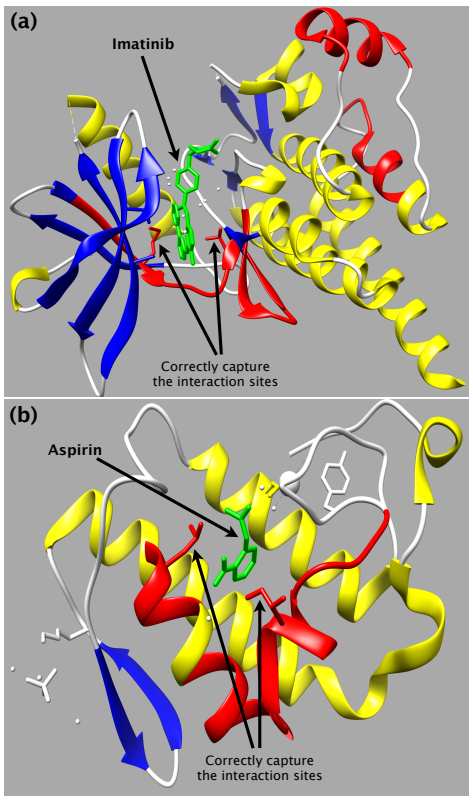

Fig. 9. Examples for visualization of CPIs with attention weights. Two CPIs, imatinib-Syk (a; PDB ID 1XBB) and aspirin-phospholipase A2 (b; PDB ID 1TGM), whose complex structures have already known, are shown. Drug compounds are highlighted in green, and the regions in proteins, which have high weight values obtained by the neural attention mechanism, are highlighted in red. The secondary structures, α-helix and β-strand are displayed in yellow and blue, respectively. (Directly) contacted residues with compounds are depicted as stick models.

neural attention mechanism could highlight unique binding sites for each compound, when we used the two different compounds and u-PA as inputs for the proposed model trained with the human dataset (Fig. S1). This result suggests that the neural attention mechanism in the proposed model is helpful for finding promiscuous domains and recognizing the unique binding sites for each compound. Again, we emphasize that the model highlights these 3D structural interaction sites between the compounds and proteins, obtained from 2D-graph and 1D-sequence information and the proposed end-to-end representation learning.

7.3 Conclusion and future work

In this paper, we have proposed end-to-end representation learning of a GNN and CNN to predict CPIs. The experimental results have demonstrated that a relatively low-dimensional and shallow neural network has the potential to outperform various existing methods on both balanced and unbalanced datasets. In addition, our attention mechanism has provided clear visualizations that make real-valued vectors easier to analyze. We believe that our study will provide new insights into end-to-end representation learning to construct a general machine learning in bioinformatics rather than using feature engineering.

It is also worth noting that we can also apply the GNN to proteins represented as 3D structured data in the DUDE dataset, whereas we have applied the CNN to proteins represented as 1D sequential data in this work. The reason is that, in practice (i) the number of 3D structured data is smaller than that of 1D sequential data, (ii) deep learning requires relatively large number of training data samples, and (iii) the model based on only protein sequences is attractive as a practical application such as a large-scale CPI prediction and screening. In addition, chemical compounds are often provided as SMILES, which can be easily transformed to 2D molecular graphs using RDKit. Based on these observations, in this paper we have proposed the use of CNN for protein sequences and GNN for molecular graphs, and achieved high prediction performance. However, the development of GNN for 3D structured proteins is an important challenge; in particular, we believe that such a "3D GNN" will allow us to achieve higher performance, provide more detailed analysis, and obtain more useful information for 3D interaction sites between compounds and proteins derived from the perspective of data-driven machine learning approach. We leave the exploration of GNN extension to future work.

Acknowledgements

This research was supported by NEDO, Japan, JSPS KAKENHI Grant Number JP17H07392, Platform Project for Supporting Drug Discovery and Life Science Research (Basis for Supporting Innovative Drug Discovery and Life Science Research (BINDS)) from AMED under Grant Number JP18am0101110, and JST CREST Grant Number JPMJCR1689 and JSPS KAKENHI Grant Number 15H01717 to JS.

References

Atwell, S., Adams, J. M., Badger, J., Buchanan, M. D., Feil, I. K., Froning, K. J., Gao, X., Hendle, J., Keegan, K., Leon, B. C., *et al.* (2004). A novel mode of gleevec binding is revealed by the structure of spleen tyrosine kinase. *Journal of Biological Chemistry*, **279**(53), 55827–55832.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *ICLR*.

Bleakley, K. and Yamanishi, Y. (2009). Supervised prediction of drug–target interactions using bipartite local models. *Bioinformatics*, **25**(18), 2397–2403.

Bredel, M. and Jacoby, E. (2004). Chemogenomics: an emerging strategy for rapid target and drug discovery. *Nature Reviews Genetics*, **5**(4), 262–275.

Cheng, F., Zhou, Y., Li, J., Li, W., Liu, G., and Tang, Y. (2012). Prediction of chemical–protein interactions: multitarget-qsar versus computational chemogenomic methods. *Molecular BioSystems*, **8**(9), 2373–2384.

Costa, F. and De Grave, K. (2010). Fast neighborhood subgraph pairwise distance kernel. In *International Conference on Machine Learning*.

Ding, H., Takigawa, I., Mamitsuka, H., and Zhu, S. (2013). Similarity-based machine learning methods for predicting drug–target interactions: a brief review. *Briefings in Bioinformatics*, **15**(5), 734–747.

Dong, Q.-W., Wang, X.-L., and Lin, L. (2006). Application of latent semantic analysis to protein remote homology detection. *Bioinformatics*, **22**(3), 285–290.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*.

Gönen, M. (2012). Predicting drug–target interactions from chemical and genomic kernels using bayesian matrix factorization. *Bioinformatics*, **28**(18), 2304–2310.

Günther, S., Kuhn, M., Dunkel, M., Campillos, M., Senger, C., Petsalaki, E., Ahmed, J., Urdiales, E. G., Gewiess, A., Jensen, L. J., *et al.* (2007). Supertarget and matador: resources for exploring drug-target relationships. *Nucleic acids research*, **36**(suppl_1), D919–D922.

Hamanaka, M., Taneishi, K., Iwata, H., Ye, J., Pei, J., Hou, J., and Okuno, Y. (2017). Cgbvs-dnn: Prediction of compound-protein interactions based on deep learning. *Molecular informatics*, **36**(1-2).

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., *et al.* (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, **29**(6), 82–97.

Jacob, L. and Vert, J.-P. (2008). Protein-ligand interaction prediction: an improved chemogenomics approach. *Bioinformatics*, **24**(19), 2149–2156.

Kearnes, S., McCloskey, K., Berndl, M., Pande, V., and Riley, P. (2016). Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, **30**(8), 595–608.

Keiser, M. J., Setola, V., Irwin, J. J., Laggner, C., Abbas, A. I., Hufeisen, S. J., Jensen, N. H., Kuijer, M. B., Matos, R. C., Tran, T. B., *et al.* (2009). Predicting new molecular targets for known drugs. *Nature*, **462**(7270), 175–181.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koes, D. R., Baumgartner, M. P., and Camacho, C. J. (2013). Lessons learned in empirical scoring with smina from the csar 2011 benchmarking exercise. *Journal of chemical information and modeling*, **53**(8), 1893–1904.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, **521**(7553), 436–444.

Liu, H., Sun, J., Guan, J., Zheng, J., and Zhou, S. (2015). Improving compound–protein interaction prediction by building up highly credible negative samples. *Bioinformatics*, **31**(12), i221–i229.

Lounkine, E., Keiser, M. J., Whitebread, S., Mikhailov, D., Hamon, J., Jenkins, J. L., Lavan, P., Weber, E., Doak, A. K., Côté, S., *et al.* (2012). Large-scale prediction and testing of drug activity on side-effect targets. *Nature*, **486**(7403), 361–367.

Medina-Franco, J. L., Giulianotti, M. A., Welmaker, G. S., and Houghten, R. A. (2013). Shifting from the single to the multitarget paradigm in drug discovery. *Drug discovery today*, **18**(9), 495–501.

Mysinger, M. M., Carchia, M., Irwin, J. J., and Shoichet, B. K. (2012). Directory of useful decoys, enhanced (dud-e): better ligands and decoys for better benchmarking. *Journal of medicinal chemistry*, **55**(14), 6582–6594.

Ragoza, M., Hochuli, J., Idrobo, E., Sunseri, J., and Koes, D. R. (2017). Protein–ligand scoring with convolutional neural networks. *Journal of chemical information and modeling*, **57**(4), 942–957.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, **20**(1), 61–80.

Schütt, K. T., Arbabzadah, F., Chmiela, S., Müller, K. R., and Tkatchenko, A. (2017). Quantum-chemical insights from deep tensor neural networks. *Nature Communications*, **8**.

Singh, N., Somvanshi, R. K., Sharma, S., Dey, S., Kaur, P., and Singh, T. P. (2007). Structural elements of ligand recognition site in secretory phospholipase a2 and structure-based design of specific inhibitors. *Current topics in medicinal chemistry*, **7**(8), 757–764.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Tabei, Y. and Yamanishi, Y. (2013). Scalable prediction of compound-protein interactions using minwise hashing. *BMC systems biology*, **7**(6), S3.

Tian, K., Shao, M., Wang, Y., Guan, J., and Zhou, S. (2016). Boosting compound-protein interaction prediction by deep learning. *Methods*, **110**, 64–72.

Tokui, S., Oono, K., Hido, S., and Clayton, J. (2015). Chainer: a next-generation open source framework for deep learning. In *Proceedings of workshop on machine learning systems (LearningSys) in the conference on neural information processing systems*.

Trott, O. and Olson, A. J. (2010). 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.

van Laarhoven, T., Nabuurs, S. B., and Marchiori, E. (2011). Gaussian interaction profile kernels for predicting drug–target interaction. *Bioinformatics*, **27**(21), 3036–3043.

Wallach, I., Dzamba, M., and Heifets, A. (2015). Atomnet: A deep convolutional neural network for bioactivity prediction in structure-based drug discovery. *arXiv*

preprint arXiv:1510.02855.

Wan, F. and Zeng, J. (2016). Deep learning with feature embedding for compound-protein interaction prediction. *bioRxiv*, page 086033.

Wishart, D. S., Knox, C., Guo, A. C., Cheng, D., Shrivastava, S., Tzur, D., Gautam, B., and Hassanali, M. (2007). Drugbank: a knowledgebase for drugs, drug actions and drug targets. *Nucleic acids research*, **36**(suppl_1), D901–D906.

Yamanishi, Y., Araki, M., Gutteridge, A., Honda, W., and Kanehisa, M. (2008). Prediction of drug–target interaction networks from the integration of chemical and genomic spaces. *Bioinformatics*, **24**(13), i232–i240.