SUPPLEMENTARY MATERIAL OF MULTI-VIEW SELF-ATTENTION FOR INTERPRETABLE DRUG-TARGET INTERACTION PREDICTION

1 Featurization Methods

This section presents the compound and target featurization methods that were used in our experiments. Other compound and target featurization methods which follow the final dimensions of the methods discussed next could also be used in our proposed architecture. We provide Pytorch implementation of these methods at `https://github.com/bbrighttaer/jova_baselines`.

1.1 Compound Featurization

1.1.1 Extended Connectivity Fingerprint

The ECFP algorithm is a state-of-the-art circular fingerprint scheme for numerically encoding the topological features of a compound [1]. ECFP decomposes a compound into substructures and assigns a unique identifier to each fragment. In the algorithm, larger substructures are composed through bond relations. A diameter parameter controls the extent to which these larger substructures can be composed. For instance, with a diameter of 4, (written as ECFP4), the largest substructure has a width of 4 bonds. Subsequently, the unique identifiers of all fragments are hashed to produce a fixed-length binary vector. This final representation indicates the presence of particular substructures. We used RDKit's [2] implementation of the ECFP algorithm in our study.

1.1.2 Molecular Graph Convolution

Motivated by recent progress in end-to-end representation learning, Molecular Graph Convolution (MGC) is a class of algorithms that, for a given layer, apply the same differentiable function to the atoms of a molecule to learn the features of the molecule from its raw form. This operation is akin to the use of kernels in the CNN architecture. Also, information about distant atoms is propagated radially through bonds, as found in circular fingerprints. Thus, composing several layers facilitate the learning of useful representations that are related to the learning objective. The earliest form of MGC is the work in [3]. It has been used in a notable number of studies and in various forms, such as that of [4], to model bioactivity. In the GraphConv method of [5], graph pool, and gather operations are proposed to augment the neural graph fingerprints algorithm of [3]. Recent progress in the domain has also produced other forms of MGCs [6]. In our study, we use the GraphConv algorithm proposed by [5]. Atom vectors are initialized using predefined physicochemical properties. The main operations of GraphConv are:

1. Graph convolution: applies molecular graph convolution to each atom.
2. Graph pool: applies a pooling function to an atom and its neighbors to get the updated feature vector of the atom.
3. Graph gather: takes the feature vectors of all atoms and applies a downsampling function to compute the fixed-length compound feature vector $x_{molecule} \in \mathbb{R}^d$.

We refer to the GraphConv implementation without the graph gather operation as GraphConv2D in this study. Hence, for a compound of $n \in \mathbb{R}$ atoms, where $a_i \in \mathbb{R}^d, i < n$, is the vector the ith atom, the output of GraphConv2D is $x_{molecule} \in \mathbb{R}^{n \times d}$.

1.1.3 Weave

Weave featurization, proposed in [7], is another form of MGC. In the weave algorithm, atom-atom pairs are constructed using all atoms in a molecule. The features of an atom are then updated using the information of all other atoms and their respective pairs. This form of update enables the propagation of information from distant atoms, albeit with increased complexity. While predefined physicochemical features are used to initialize atom vectors, topological properties are used to initialize atom-atom pair vectors. The following are the main operations of the weave featurization scheme:

1. Weave: applies the weave operation as described above.
2. Weave gather: computes the compound feature vector $x_{molecule} \in \mathbb{R}^d$ as a function of all atom feature vectors.

We refer to the Weave implementation without the graph gather operation as Weave2D in this study. Thus, for a compound of $n \in \mathbb{R}$ atoms, where $a_i \in \mathbb{R}^d, i < n$, is the vector the ith atom, the output of Weave2D is $x_{molecule} \in \mathbb{R}^{n \times d}$.

1.1.4 Graph Neural Network

In [8], a Graph Neural Network (GNN) was proposed for molecular graphs. GNN maps a given molecular graph to a fixed-length feature vector using two differentiable functions: transition and output functions. Atoms are depicted as nodes, and the bonds within a molecule form the edges in the molecular graph. For each entity in the graph, substructures within a specified radius are encoded to form the embedding profile of the entity. These profiles are then mapped to indices of an embedding matrix that is trained using backpropagation. The transition function is used to update the features of atoms and bonds towards determining the vector representation of the molecule. Thus, applying different transition functions hierarchically recapitulates the convolution operation in a CNN since the same transition function is applied to all entities in the graph in a layer. The output function downsamples the set of node vectors from the transition phase to get the fixed-length molecular representation, $x_{molecule} \in \mathbb{R}^d$.

In our study, we use a variant of GNN dubbed GNN2D. This variant omits the downsampling phase of the GNN operation. Thus, for a compound of $n \in \mathbb{R}$ atoms, where $a_i \in \mathbb{R}^d, i < n$, is the vector the ith atom, the output of GNN2D is $x_{molecule} \in \mathbb{R}^{n \times d}$.

1.2 Target Featurization

1.2.1 Protein Sequence Composition

As regards target quantization, PSC is a well-known predefined scheme for capturing subsequence information. It consists of Amino Acid Composition (AAC), Dipeptide Composition (DC), and Tripeptide Composition (TC). AAC provides information about the frequency of each amino acid. DC determines the frequency of every two amino acid combinations, whereas TC computes the frequency of every three amino acid combinations. The dimension of a PSC feature vector is 8420.

1.2.2 Prot2Vec

Similar to compound featurization, efforts have been made to learn protein representations directly from their raw forms. Learning protein vectors is typically achieved by learning embedding vectors using NLP techniques such as the word2vec and GloVe models [9, 10]. This approach also maintains the temporal properties in the target sequence. In [11], it was shown that the NLP approach could be used to develop rich target representations. Therefore, we construct a vocabulary of n-gram subsequences (biological words) following the splitting scheme of [11]. We set $n = 3$ in this study. In Figure 1, the approach we use to construct the 3-gram profile of a protein sequence is illustrated. The raw form of the protein is split into three non-overlapping representations. The words of all three sequences make up the vocabulary used in this study. We then move across the three splits to construct the overlapping 3-gram target profile. Each word in the dictionary D is mapped to a randomly initialized vector $x_i \in \mathbb{R}^d$, $i < |D|$, that is updated during training.

In order to make computations tractable, we group subsequences using a non-overlapping window approach similar to the method in [8].

Specifically, given the target profile $\mathcal{S} = \{s_1, s_2, ..., s_n | n \in \mathbb{R}, n \leq |D|\}$, we retrieve the vectors of each word to construct the set of vectors $X = \{x_1, x2, ...x_n\}$. Setting the window size to 3, for didactic purposes, we group X as:

$$X = \{[x_1, x_2, x_3], [x_4, x_5, x_6], ..., [x_{n-2}, x_{n-1}, x_n]\}$$

where $[\cdots]$ is a concatenation operator. Also, $x_{i:i+w-1}$ denotes the window $\{x_i, ..., x_{i+w-1}\}$ where w is the window size. Note that if $|x_{i:i+w-1}| < w$ by k elements, we add $z \in \mathbb{R}^d$ to the window k times. Here, z is a vector of all zeros. Thus, each window is a wd-dimensional vector. Pooling functions or RNN could then be used to process these windows/segments into a fixed-length representation of the target.

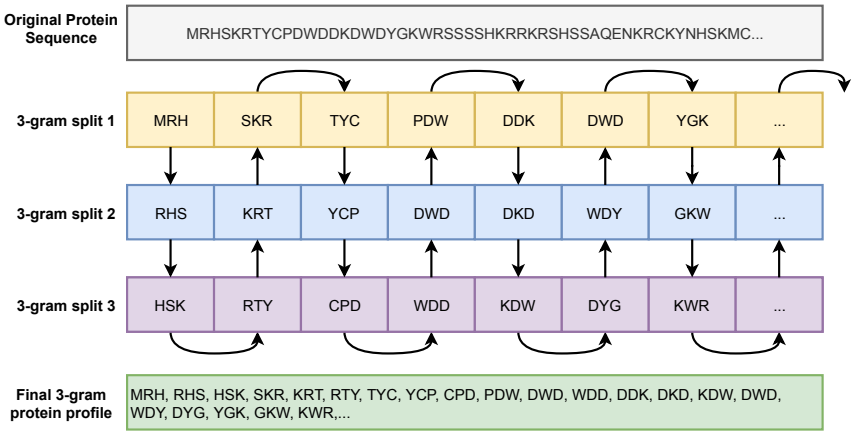


Figure 1: **Target sequence 3-gram representation.** The original target sequence is split into three non-overlapping sequences (split 1, split 2, and split 3). The overlapping 3-gram profile of the protein is constructed by moving across the three sequences as depicted by the arrow.

1.2.3 Protein Convolutional Neural Network

Protein Convolutional Neural Network (PCNN), proposed by Tsubaki et al. [8], is another end-to-end representation learning scheme for target sequences. It uses a similar approach to the Prot2Vec method (see section 1.2.2), but with overlapping windows, to construct target representations. The subsequent discussion on the PCNN uses Prot2Vec to encode target data and also has a minor variation of the convolution operation in [8]. Given $c_i^{(l-1)} = x_{i:i+w-1}$ to be the ith window of $\mathcal{C}^{(l-1)} = \{c_1^{(l-1)}, c_2^{(l-1)}, .., c_{|\mathcal{C}|}^{(l-1)}\}$, where l denotes the lth convolution layer, PCNN computes $c_i^{(l)}$ of $C^{(l)}$ as,

$$c_i^{(l)} = f(W^l c_i^{(l-1)} + b^{(l)}) \tag{1}$$

where $f(\cdot)$ is a nonlinear activation function, we let $W^l \in \mathbb{R}^{wd \times wd}$ be the kernel, and $b \in \mathbb{R}^{wd}$. Applying equation 1 multiple times enable nonlinear properties to be learned at different levels of abstraction. In order to produce a d-dimensional vector $c_i^{(L)}$ for the last PCNN layer L, we let $W^L \in \mathbb{R}^{d \times wd}$ and $b^L \in \mathbb{R}^d$. Thus, the final output is $\mathcal{C}^L \in \mathbb{R}^{|\mathcal{C}| \times d}$. We refer to the rows of C^L as segments.

To compute the vector representation of the target, [8] proposed using the average pooling function. It is easy to realize that other differential pooling functions, such as the max and sum functions, could be employed. Moreover, an attention mechanism is proposed in [8], where the compound representation is used to compute attention weights for the segments of the target representation. In this context, the compound vector dimension and the segment dimension must be equal. In this study, we refer to the attention variant as PCNN with Attention (PCNNA). We refer the reader to [8] for the exposition of PCNNA.

Additionally, we use a variant of the PCNN architecture called PCNN2D. This variant omits the downsampling and attention phases of the PCNN method.

2 Baseline Models

In this section we provide an introduction to the baseline models used in our study. The implementations of these methods can be found at `https://github.com/bbrighttaer/jova_baselines`

2.1 KronRLS

The KronRLS method proposed in [12] is a generalization of the RLS method in which the data is assumed to consist of pairs (compounds and targets, in this case). It is a kernel-based approach for predicting the binding affinity between

a compound-target pair. Specifically, given a set of compound-target pairs $X = \{x_1, x_2, ..., x_m\}$ as training data with their corresponding binding-affinity values $Y = \{y_1, y_2, ..., y_m\}$, where $i < m$ and $m \in \mathbb{R}$, KronRLS learns a real-valued function $f(x)$ that minimizes the objective,

$$J(f) = \sum_{i=1}^{m}(y_i - f(x_i))^2 + \lambda\|f\|_k^2 \tag{2}$$

where λ is a regularization parameter and $\|f\|_k$ is the norm of the minimizer f associated with the kernel k in equation 3. Basing on the representer theorem, [12] defines the minimizer as,

$$f(x) = \sum_{i=1}^{m} a_i k(x, x_i) \tag{3}$$

where the kernel function k is a symmetric similarity measure between two compound-target pairs. Given a dataset of m samples, k can be represented as $K \in \mathbb{R}^{m \times m}$ computed as $K_c \otimes K_t$ if X contains all possible compound-target pairs. Here, K_c and K_t are the kernel matrices of the compounds and targets, respectively. In this context, the parameters a_i of f can be determined in closed form by solving a system of $|C||T|$ linear equations:

$$(K + \lambda I)a = y \tag{4}$$

where C is the set of compounds, T is the set of targets, $a \in \mathbb{R}^m$, $y \in \mathbb{R}^m$, and $I \in \mathbb{R}^{|C| \times |T|}$ is an identity matrix. Equation 4 assumes that Y contains the binding affinities of all $|C| \times |T|$ pairs in order to be solved in closed form. In cases where this assumption does not hold, conjugate gradient with Kronecker algebraic optimization could be employed to determine a. Howbeit, other imputation strategies have been employed to maintain the closed-form evaluation of equation 4 [12].

2.2 SimBoost

SimBoost, proposed in [13], is a gradient boosting approach to predict the binding affinity between a compound and a target. The authors proposed three types of features to construct the feature vector of a given compound-target in training set $X = \{x_1, x_2, ..., x_m | i < m, m \in \mathbb{R}\}$:

1. Type 1: features for each compound and target based on average similarity values, and information about their frequency in the dataset.
2. Type 2: features for entities determined from their respective similarity matrices.
3. Type 3: features for each compound-target pair computed using a compound-target network.

Given compound c_i and target t_i, the feature vector $x_i \in \mathbb{R}^d$ of the pair $c_i - t_i$ is constructed by concatenating the type 1 and type 2 features of both c_i and t_i, and the type 3 features of the pair $c_i - t_i$. The corresponding binding affinity $\hat{y}_i$ of x_i is computed as,

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathbf{F} \tag{5}$$

where $\mathbf{F}$ is the space of all possible trees and K is the number of regression trees. Using the additive ensemble training approach, the set of trees $\{f_k\}$ are learned by minimizing the following regularized objective:

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{6}$$

where Ω determines model complexity to control overfitting, $l(\cdot)$ is a differentiable loss function which evaluates the prediction error and y_i is the true binding affinity corresponding to x_i.

2.3 PADME

In [14], PADME is proposed to model DTIs. The authors proposed two variants of PADME: PADME-ECFP and PADME-GraphConv. The former variant constructs feature vectors of compounds using the ECFP scheme, whereas the latter learns the representations of compounds using Molecular Graph Convolution [5]. On the other hand, targets are represented using PSC [15]. After that, for a given compound-target pair, the feature vector $x_i \in \mathbb{R}^d$ is constructed as

the concatenation of the compound and target feature vectors. This constructed feature vector then becomes an input to a Fully Connected Neural Network (FCNN) which minimizes the regularized Mean Square Error (MSE) objective,

$$\underset{\theta}{\operatorname{argmin}} \sum_{i=0}^{N} (y_i - f(x_i;\theta))^2 + \lambda \|f\|_k^2 \tag{7}$$

where $f(x_i;\theta)$ outputs $\hat{y}_i$ as the predicted value using parameters θ and λ is a regularization parameter to control overfitting.

2.4 IVPGAN

In our previous study [16], we proposed IVPGAN to predict DTIs using a multi-view approach to represent a compound and PSC to construct the target feature vector. While ECFP is used to represent predefined compound features, MGC is used to learn the representation of a compound given the graphical structure encoded in its SMILES notation. Using an Adversarial Loss (AL) training technique, the following objective is minimized:

$$L_G = L_G^{MSE} + \lambda L_G^{AL} \tag{8}$$

where

$$L_G^{MSE} = \sum_{i=0}^{N} (y_i - f([v_i^c, g(d_i;\theta^g), v_i^p];\theta^f))^2 + \beta(\|f\|_k^2 + \|g\|_k^2) \tag{9}$$

$$L_G^{AL} = \mathbb{E}_{\mathbf{x}\sim G}\left[-logD(\mathbf{x})\right] \tag{10}$$

, θ^f and θ^g are trainable parameters, $g(\cdot) \in \mathbb{R}^d$, $[\cdots]$ is a concatenation operator, $\|\cdot\|_k^2$ is a norm operator, λ is a hyperparameter that is used to control the combination of MSE and the AL objectives, and β is a regularization parameter that controls overfitting. L_G^{MSE} is the MSE objective of the DTI prediction model, which is treated as the generator of a Generative Adversarial Network (GAN). L_G^{AL} is the generator objective component of the GAN whose discriminator objective is expressed as,

$$L_D^{AL} = \mathbb{E}_{\mathbf{x}\sim p}\left[-logD(\mathbf{x})\right] + \mathbb{E}_{\mathbf{x}\sim G}\left[-log(1 - D(\mathbf{x}))\right] \tag{11}$$

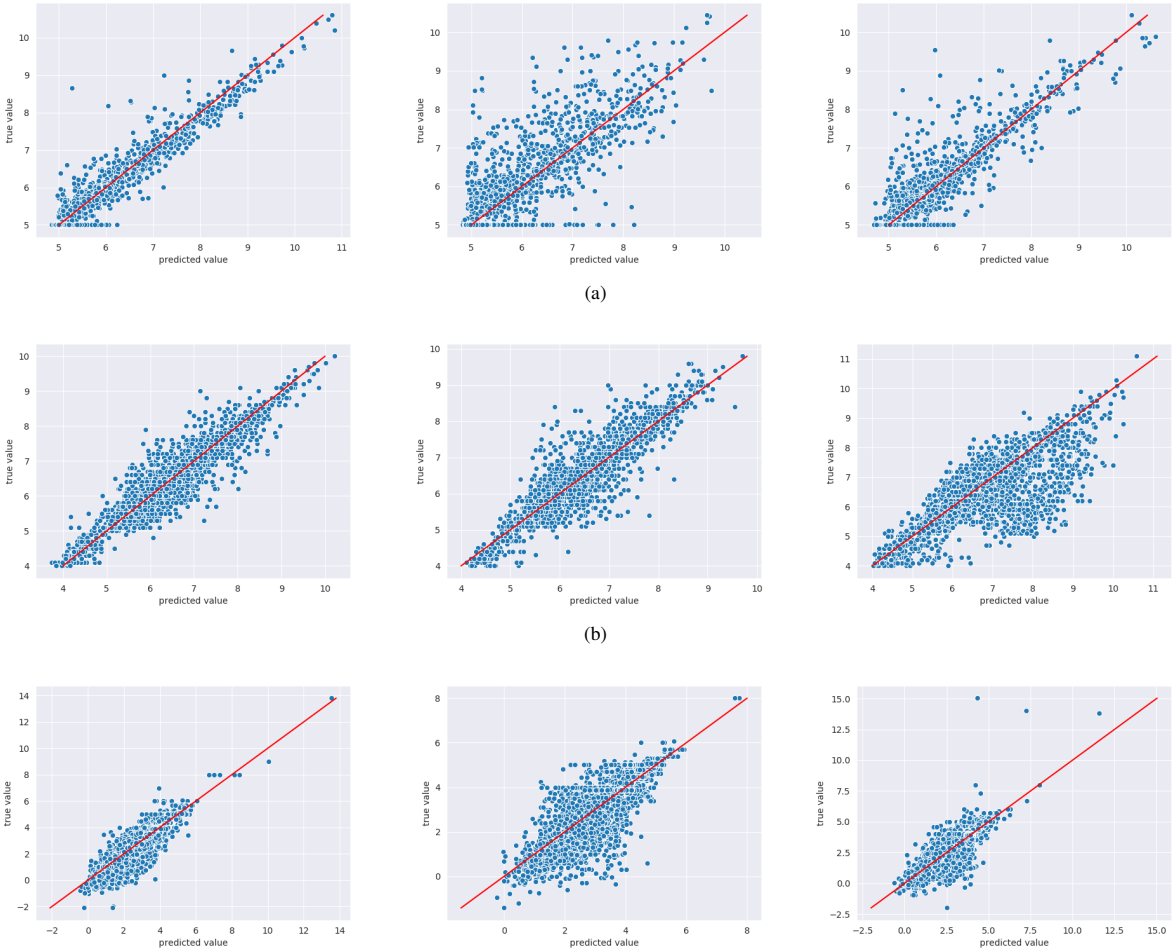
where the distributions p and G of equations 10 and 11 are derived from the neighborhood alignment matrices constructed from the labels and predicted values, respectively, as explained in [16].

2.5 CPI for Interaction Prediction

The authors of [8] proposed a Compound-Protein Interaction (CPI) prediction model that uses end-to-end representation learning to encode compound graphs and protein sequences for bioactivity prediction. The graph neural network and protein convolutional neural network featurization methods used in [8] are described in Sections 1.1.4 and 1.2.3. While the CPI model in [8] was trained using Binary Cross-Entropy (BCE) loss, we omit the Sigmoid endpoint to train the model using the MSE objective for predicting binding affinities.

3 Prediction Plots

In this section we present the scatter plots of the baseline and the two JoVA models of the study. Columns 1, 2, and 3 show the warm split, cold-drug split, and cold-target split results, respectively. The first, second, and third rows correspond to the Davis, Metz, and KIBA datasets, respectively.

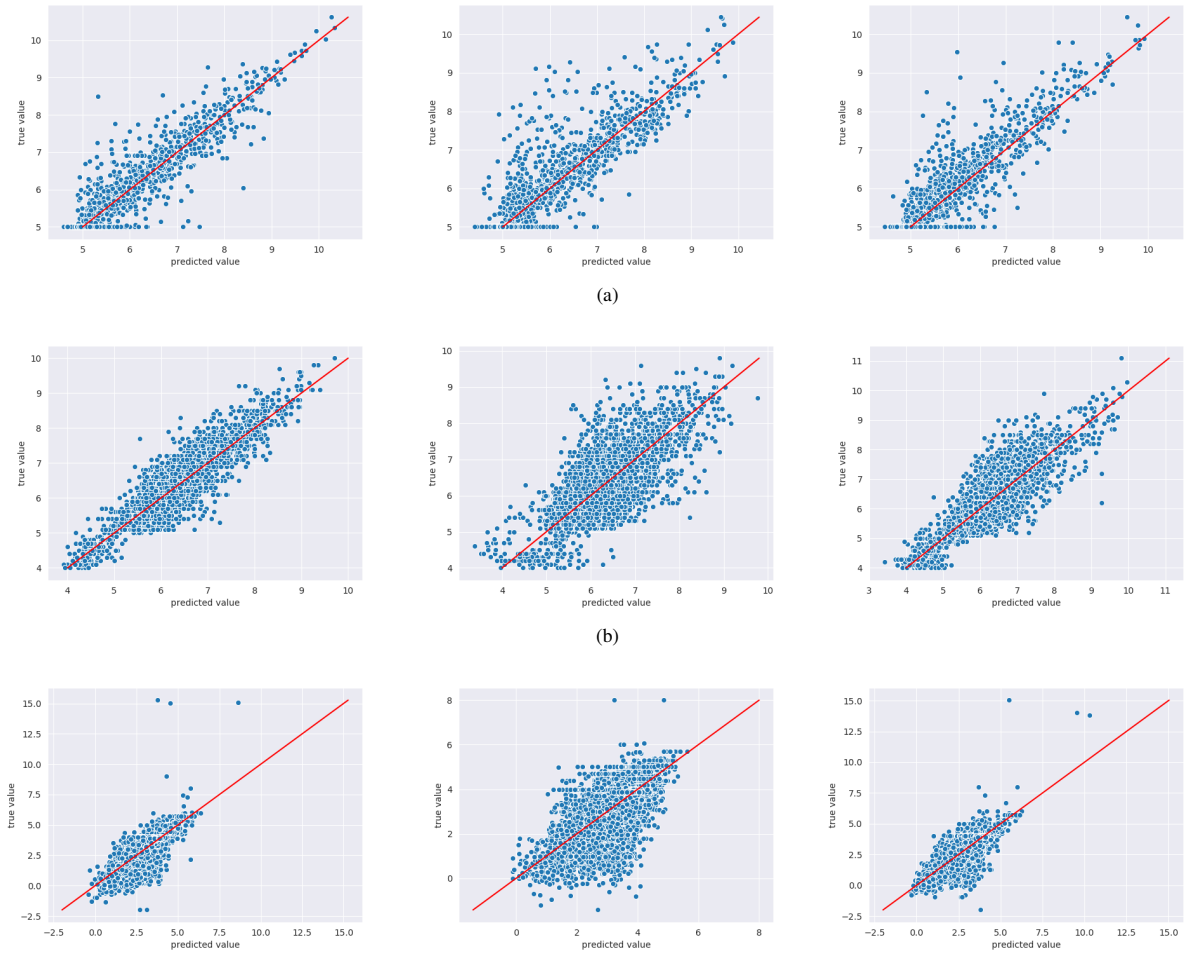


(a)

(b)

(c)

Figure 2: Predictions of ECFP8-PSC on the benchmark dataset.

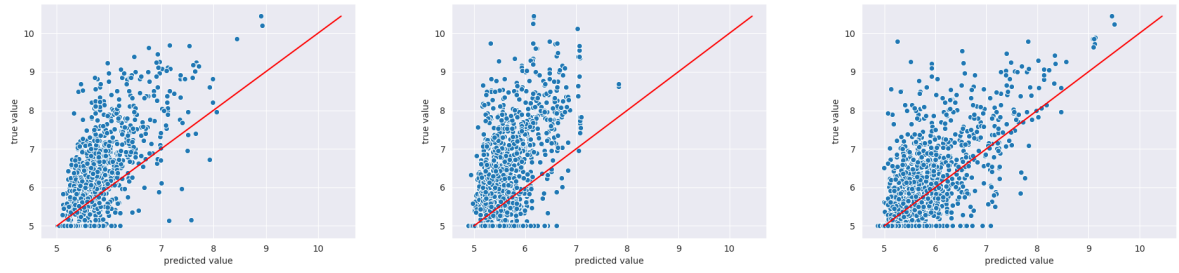


(a)

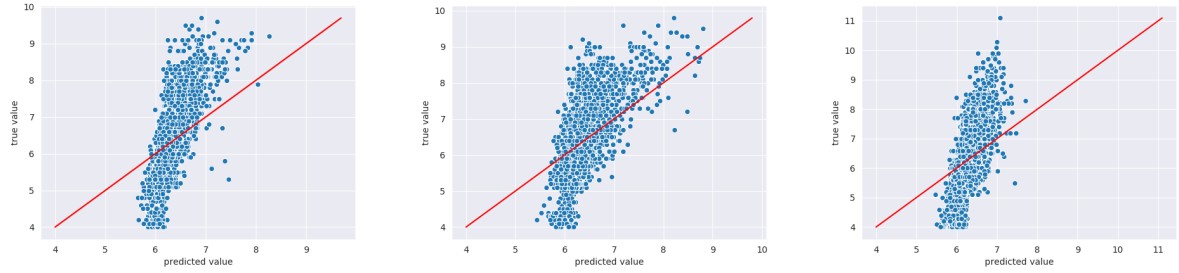
(b)

(c)

Figure 3: Predictions of GraphConv-PSC on the benchmark dataset.



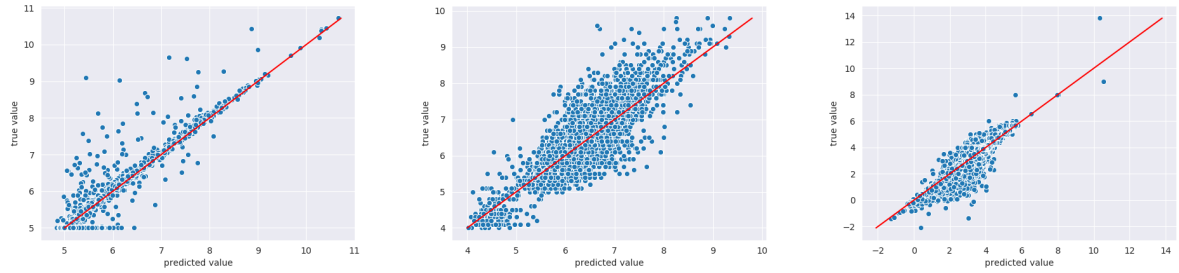
(a)



(b)

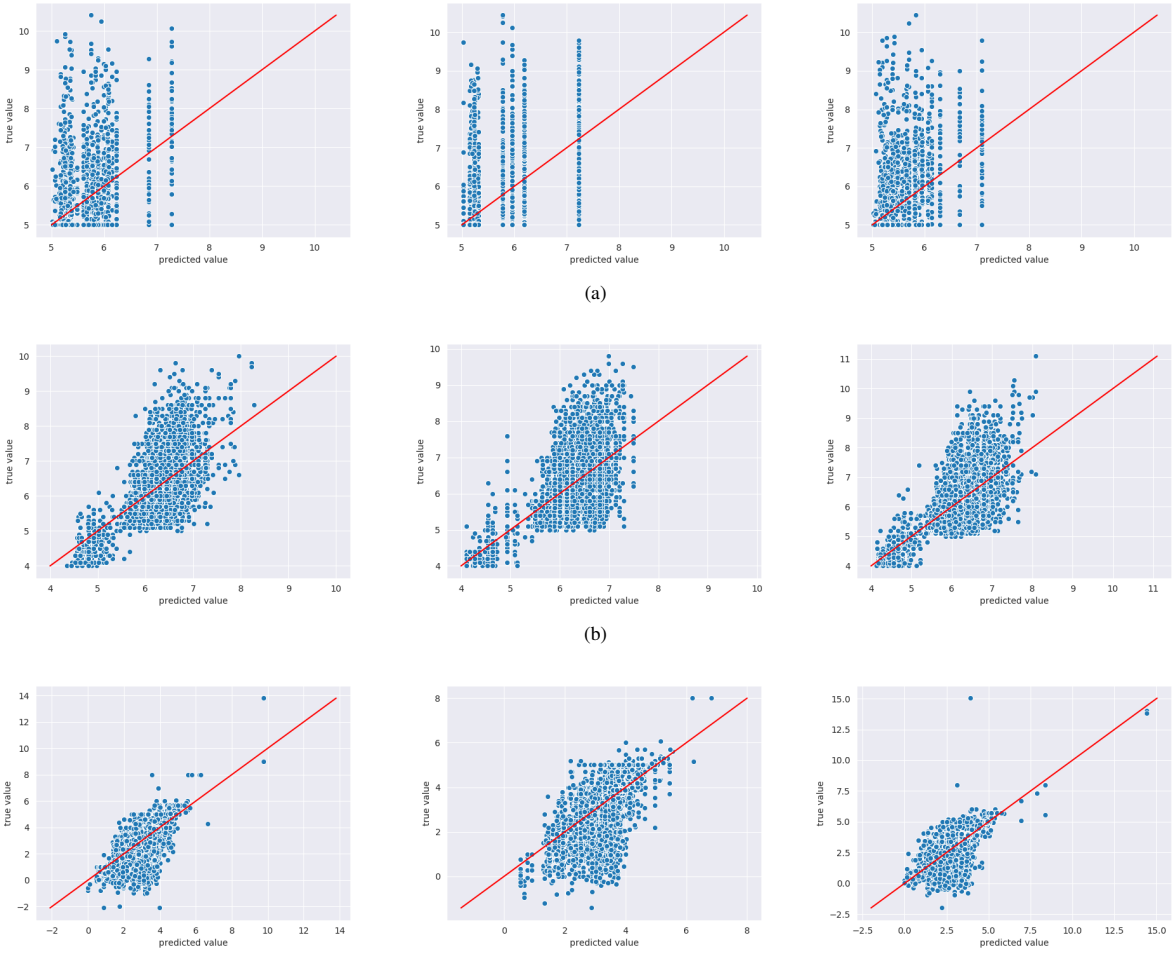
(c)

Figure 4: Predictions of KronRLS on the benchmark dataset.



(a)

Figure 5: Predictions of SimBoost on the benchmark dataset.

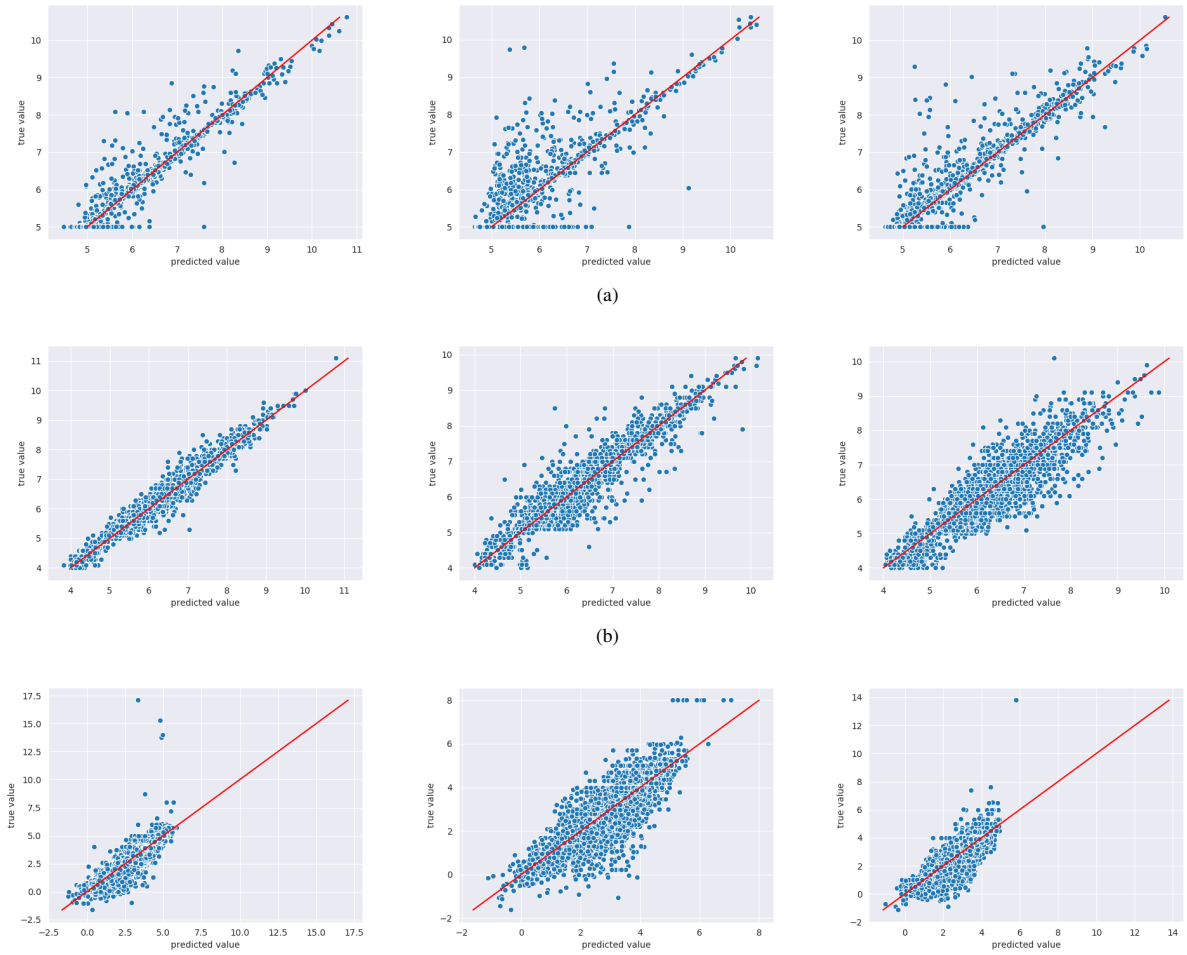


(a)

(b)

(c)

Figure 6: Predictions of CPI-Reg on the benchmark dataset.

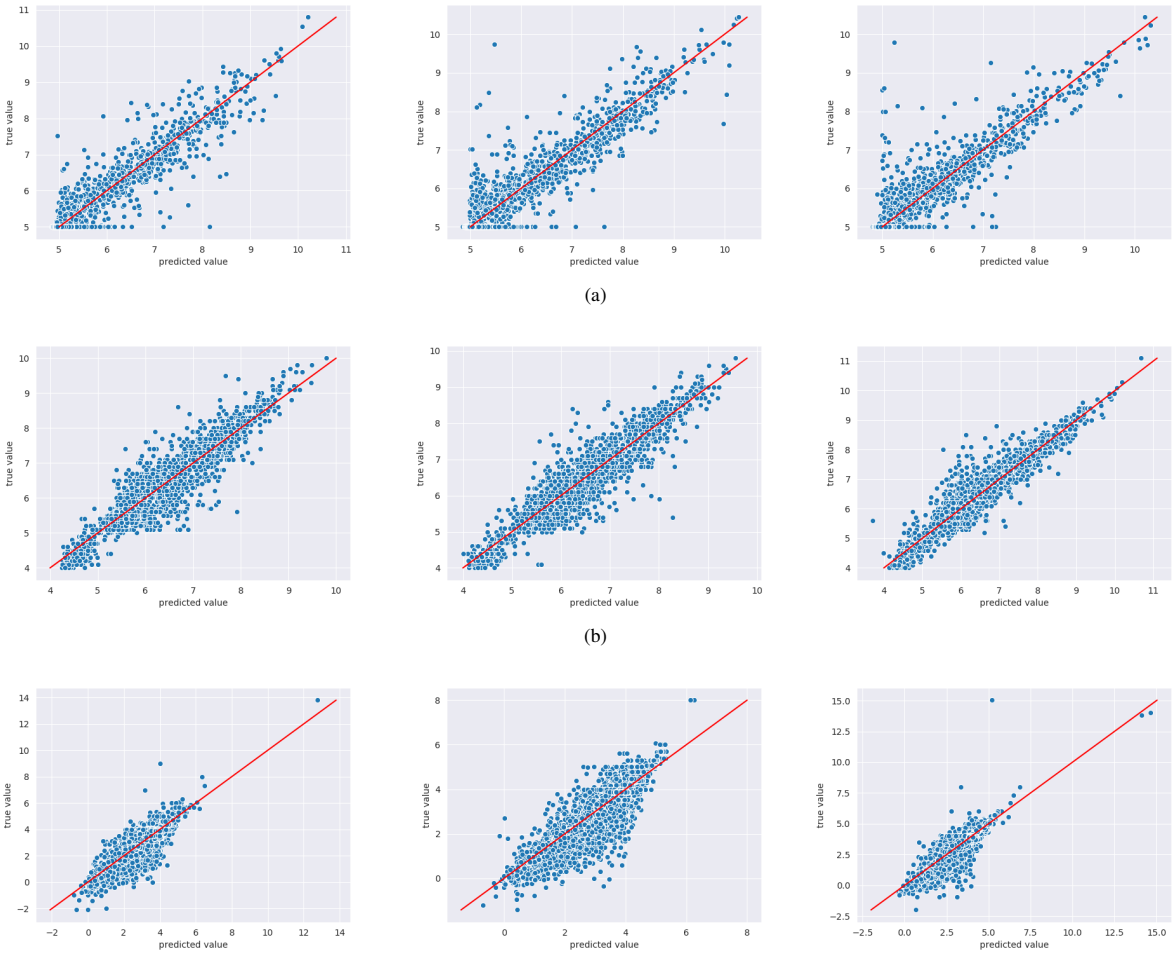


(a)

(b)

(c)

Figure 7: Predictions of IVPGAN on the benchmark dataset.

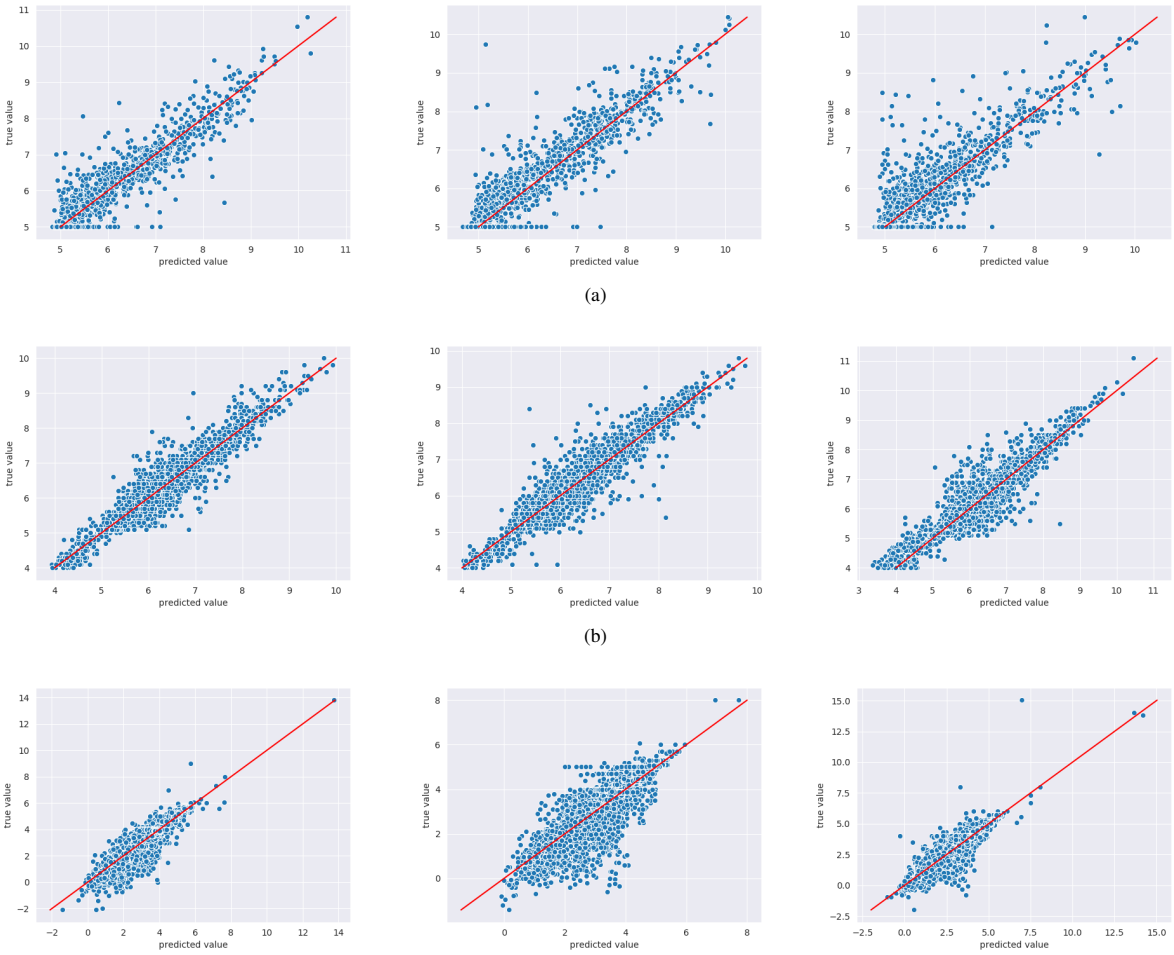


(a)

(b)

(c)

Figure 8: Predictions of JoVA1 on the benchmark dataset.



(a)

(b)

(c)

Figure 9: Predictions of JoVA2 on the benchmark dataset.

References

[1] David Rogers and Mathew Hahn. Extended-Connectivity Fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010.

[2] Greg Landrum. RDKit: Open-source Cheminformatics, 2006.

[3] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional Networks on Graphs for Learning Molecular Fingerprints. pages 1–9, 2015.

[4] Kyle Yingkai Gao, Achille Fokoue, Heng Luo, Arun Iyengar, Sanjoy Dey, and Ping Zhang. Interpretable drug target prediction using deep neural representation. *IJCAI International Joint Conference on Artificial Intelligence*, 2018-July:3371–3377, 2018.

[5] Han Altae-Tran, Bharath Ramsundar, Aneesh S. Pappu, and Vijay Pande. Low Data Drug Discovery with One-Shot Learning. *ACS Central Science*, 2017.

[6] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: A benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018.

[7] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*, 30(8):595–608, 2016.

[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, 2019.

[9] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, pages 1–12, 2013.

[10] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. In *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2014.

[11] Ehsaneddin Asgari and Mohammad R.K. Mofrad. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PLoS ONE*, 10(11):1–15, 2015.

[12] Tapio Pahikkala, Antti Airola, Sami Pietilä, Sushil Shakyawar, Agnieszka Szwajda, Jing Tang, and Tero Aittokallio. Toward more realistic drug-target interaction predictions. *Briefings in Bioinformatics*, 16(2):325–337, 2015.

[13] Tong He, Marten Heidemeyer, Fuqiang Ban, Artem Cherkasov, and Martin Ester. SimBoost: a read-across approach for predicting drug-target binding affinities using gradient boosting machines. *Journal of Cheminformatics*, 9(1):1–14, 2017.

[14] Qingyuan Feng, Evgenia Dueva, Artem Cherkasov, and Martin Ester. PADME: A Deep Learning-based Framework for Drug-Target Interaction Prediction. pages 1–21, 2018.

[15] Dong Sheng Cao, Qing Song Xu, and Yi Zeng Liang. Propy: A tool to generate various modes of Chou's PseAAC. *Bioinformatics*, 29(7):960–962, 2013.

[16] B. Agyemang, W. Wu, M. Y. Kpiebaareh, and E. Nanor. Drug-target indication prediction by integrating end-to-end learning and fingerprints. In *2019 16th International Computer Conference on Wavelet Active Media Technology and Information Processing*, pages 266–272, 2019.