Received 25 June 2023, accepted 20 July 2023, date of publication 27 July 2023, date of current version 2 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3299267

METHODS

Co-Attention Graph Pooling for Efficient Pairwise Graph Interaction Learning

JUNHYUN LEE[1], BUMSOO KIM[2], MINJI JEON[3], AND JAEWOO KANG[1]
[1]Department of Computer Science and Engineering, Korea University, Seoul 02841, South Korea
[2]LG AI Research, Seoul 07796, South Korea
[3]Department of Medicine, Korea University College of Medicine, Seoul 02708, South Korea

Corresponding authors: Jaewoo Kang (kangj@korea.ac.kr) and Minji Jeon (mjjeon@korea.ac.kr)

The work of Junhyun Lee and Jaewoo Kang was supported in part by the National Research Foundation of Korea under Grant NRF-2023R1A2C3004176, in part by the Ministry of Science and ICT of Korea Supervised by the Institute of Information and Communications Technology Planning and Evaluation under Grant IITP-2023-2020-0-01819, and in part by the Korea Health Industry Development Institute under Grant HR20C0021(3). The work of Minji Jeon was supported in part by the National Research Foundation of Korea under Grant NRF-2022R1F1A1070111, and in part by the Ministry of Science and ICT of Korea Supervised by the Institute of Information and Communications Technology Planning and Evaluation under Grant IITP-2022-RS-2022-00156439.

ABSTRACT Graph Neural Networks (GNNs) have proven to be effective in processing and learning from graph-structured data. However, previous works mainly focused on understanding single graph inputs while many real-world applications require pair-wise analysis for graph-structured data (e.g., scene graph matching, code searching, and drug-drug interaction prediction). To this end, recent works have shifted their focus to learning the interaction between pairs of graphs. Despite their improved performance, these works were still limited in that the interactions were considered at the node-level, resulting in high computational costs and suboptimal performance. To address this issue, we propose a novel and efficient graph-level approach for extracting interaction representations using co-attention in graph pooling. Our method, Co-Attention Graph Pooling (CAGPool), exhibits competitive performance relative to existing methods in both classification and regression tasks using real-world datasets, while maintaining lower computational complexity.

INDEX TERMS Graph neural networks, graph pooling, pairwise graph interaction, drug-drug interaction, graph edit distance.

I. INTRODUCTION

Recent advancements in both aggregation [1], [2], [3], [4], [5] and pooling operations [6], [7], [8], [9], [10], [11], [12] have significantly improved the capabilities of Graph Neural Networks (GNNs), enabling for more robust learning of complex graph representations and enhancing performance in downstream tasks like graph classification, node classification, and link prediction. However, the scope of these approaches are limited on a single graph input while many real-world tasks (e.g., scene graph matching, code search, and drug-drug interaction prediction) require pair-wise analysis of graph-structures. Therefore, recent studies in GNNs have shifted their focus to representation learning over pairs of input graphs.

One of the earliest approaches for paired graph representation learning using GNNs is the graph convolutional Siamese network [13]. In Graph Convolutional Siamese network, the input pairs must share identical graph topology, and the paired training is done by simply concatenating the individual graph representations. However, since it does not take the interaction between the graphs during the embedding process into account, each graph is embedded into a single *static* representation regardless of its pair. This static representation can limit the expressiveness of the many-to-many relationships between pairwise graphs [14]. For instance, when predicting interactions between chemical compounds, each molecular graph can have multiple functional groups, which are important sub-graphs for the task. Since the contribution of each

The associate editor coordinating the review of this manuscript and approving it for publication was Chao Tong.

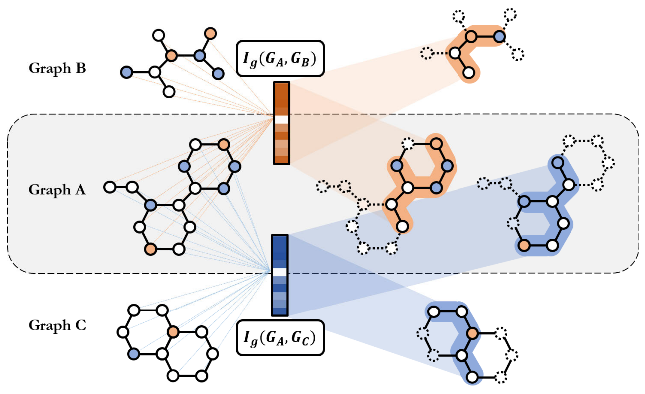


FIGURE 1. **Illustration of the concept of co-attention graph pooling. The function $I_g(\cdot,\cdot)$ represents the interaction representation between two graphs, as described in Section III. Our proposed pooling method is able to extract different sub-graphs from the same graph representation when considering different pairs. Graph A has representations from different sub-graphs when paired with Graph B and Graph C.**

functional group to the interaction depends on its pair, representing molecules with a single static representation may be limited in terms of expressiveness. To overcome this limitation, it is necessary to consider the interaction between the input pair of graphs.

Subsequent works [14], [15], [16] proposed architectures considering the interaction between the input pair using the co-attention mechanism, which is an intuitive way to contemplate pairwise interactions. While co-attention has improved the predictive power of these methods, they are still limited to obtaining the interaction representation at the *node-level*. This not only leads to increased complexity but also generates redundant output as every node pair of the two input graphs must be considered.

In this paper, we propose an efficient method for considering interactions between graphs at the *graph-level* by applying co-attention to graph pooling. Our Co-attention Graph Pooling (CAGPool) dynamically represents each input graph based on its interaction with the opposite graph in the pair (as illustrated in Figure 1), while adding minimal computation complexity. Our model outperforms baselines even without using the additional information commonly used in baseline methods on real-world public benchmark datasets for both classification and regression tasks in paired graph representation learning: drug-drug interaction classification and graph similarity regression. The implementation is publicly available.[1]

II. RELATED WORK

The overall pipeline of our model is divided into two components: Graph Convolution and Graph Pooling with Co-attention. In this section, we provide a detailed overview of the previous works related to each component.

[1]https://github.com/LeeJunHyun/CoAttentionGraphPooling

A. GRAPH CONVOLUTION NETWORKS

Researchers have been actively working on using convolutional neural networks to process graph-structured data. However, the conventional convolution operation, which is defined on grids, is difficult to be directly applied on graphs due to irregular structure. To address this issue, previous researches have defined graph convolution by using the Fourier transform (i.e. spectral graph convolution) [1], [17] or spatial graph connectivity (i.e. non-spectral graph convolution) [2], [18]. The spectral approaches can be represented by the work of Kipf and Welling [1], where graph convolution is redefined on the Fourier domain with a localized first-order approximation. Non-spectral graph convolutions align to the work of Hamilton et al. [18],where graph convolution is defined as an aggregation function of a node's neighborhood representations. Building upon these works, subsequent studies have aimed to improve the expressive power of both node-level and graph-level representations [3], [4], [5].

B. GRAPH POOLING

Pooling is a technique that prevents overfitting in modern neural network models with a large number of parameters by reducing the size of the representations. This allows the model to generalize better to new data. There are two main categories of graph pooling methods: global pooling and hierarchical pooling. Hierarchical pooling methods can be further grouped into *pooling by graph transformation* and *pooling by node selection*, based on how they produce the reduced node and edge sets.

Global graph pooling methods are techniques that transform the variable-sized representation of nodes produced by GNNs into a fixed-sized vector representation. This can then be used as input for downstream tasks such as prediction. Some common global graph pooling methods include using simple aggregation functions such as summation, average, and max to combine the node representations. The SortPool method [19] involves selecting the top k nodes from the graph based on a certain criterion, and using their representations as the pooled output. SimGNN [16] proposes using global context-aware attention to weight the contributions of different nodes, and producing a pooled representation as a weighted sum of all the node representations in the graph. These methods allow for the use of GNN output as input for downstream tasks, and have been applied to a range of tasks including graph classification and regression.

Pooling by graph transformation is one of the hierarchical graph pooling methods where nodes are clustered and merged under certain criteria [6], [8]. The main purpose of *Pooling by graph transformation* is to learn the assignment matrix that transforms input nodes into new cluster nodes. The new adjacency matrix of cluster nodes is re-defined according to the assignment matrix. The *Pooling by graph transformation* has the advantage of keeping all node information. However, there is a computational complexity issue when calculating the new adjacency matrix for cluster nodes.

And also, the output of several pooling layers is not easily interpreted through the transformed graph.

Pooling by node selection creates hierarchical graph representations via trainable indexing method. TopKPool [20] uses a trainable projection vector to calculate node scores and accordingly select nodes with the top-k score. SAGPool [7] improves upon TopKPool by using GNNs to consider the topology of the graph along with the node features to calculate node scores. Our method aligns with the work of *pooling by node selection* as it has comparably lower complexity than pooling by graph transformation. Although information loss might occur during node discarding, we propose that this helps the model to dynamically focus on different sub-graphs for different graph pairs by eliminating the representations of the nodes that are irrelevant to the interaction.

C. CO-ATTENTION MECHANISM

Attention mechanisms, which allow neural networks to assign trainable importance weights to input [21], have been widely used in deep learning research. There have been several works that extend the use of attention to input in the form of pairs. For example, Seo et al. [22] proposed a bidirectional attention flow model that includes both query-to-context and context-to-query attentions (i.e., co-attention) for machine comprehension tasks. Additionally, Deac et al. [14] demonstrated that co-attention obtained from pairs of graphs can significantly improve the predictive power of GNNs. In this work, we leverage the co-attention mechanism for graph pooling to select nodes in paired graphs that should be aware of each other.

III. METHODS

In this section, we first describe the basic notations for general pairwise graph representation learning. Then, we introduce our proposed method: Co-Attention Graph Pooling. We denote the goal of our Co-Attention Graph Pooling, i.e., CAGPool, followed by detailed explanations for each components of our overall pipeline for pair-wise graph representation learning using CAGPool (see Figure 2 for detailed illustration).

A. PROBLEM SETTING

Let $G = (V, E)$ denote the undirected graph, where V is the set of vertices and E is the set of edges. Its adjacency matrix A can be constructed with $A_{i,j} = 1$ if $(i,j) \in E$ and $A_{i,j} = 0$ otherwise. Node attributes can be represented as a matrix form: $X^{(0)} \in \mathbb{R}^{N\times F}$, where $N = |V|$ is the number of nodes and F is the feature dimension. Our main task is to predict the labels for a pair of input graphs $(G_{\mathcal{A}}, G_{\mathcal{B}})$, thus designing a model that learns the function $f : G \times G \mapsto R \in \mathbb{R}^o$, where o denotes the output dimension.

While learning f, it has been studied that considering the interaction between the input graph pairs leads to better prediction [14], [16]. For clear comparison with previous methods, we define $I(G)$ as the representation of the graph G and the interaction representation of two graphs as $I(G_{\mathcal{A}}, G_{\mathcal{B}})$, where $I_n(G_{\mathcal{A}}, G_{\mathcal{B}})$ and $I_g(G_{\mathcal{A}}, G_{\mathcal{B}})$ each denotes the interaction at the node-level and graph-level, respectively. Since interactions were utilized at the node-level $(I_n(G_{\mathcal{A}}, G_{\mathcal{B}}))$ in previous works [14], [16], [23], they suffer from high complexity (see Figure 3 and Section V-B).

The main goal of this paper is to consider the interaction representation at graph-level (i.e., $I_g(G_{\mathcal{A}}, G_{\mathcal{B}})$) instead of node-level (i.e., $I_n(G_{\mathcal{A}}, G_{\mathcal{B}})$). To this end, we propose a graph pooling module that learns a mapping function $g : G \times G \mapsto G' \times G'$ that improves the predictive power of f, where $G' = (V', E')$ is the sub-graph with $V' \subset V$ and $E' \subset E$. The individual representations $I(G_{\mathcal{A}}), I(G_{\mathcal{B}})$ for each graph are then refined to $I(G'_{\mathcal{A}}), I(G'_{\mathcal{B}})$ respectively by using the interaction representation $I_g(G_{\mathcal{A}}, G_{\mathcal{B}})$.

B. NODE EMBEDDING WITH GRAPH CONVOLUTION

Our overall pipeline follows the basic architecture of graph convolution Siamese network [24]. We receive a pair of graphs as input and apply the graph convolution with shared weights to each graph of the pair. In the graph convolution layer, we update node representations by neighborhood aggregation. We use the graph convolution suggested by Kipf and Welling [1],

$$X^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X^{(l)}\Theta^{(l)}) \tag{1}$$

where $X^{(l)}$ is the node representation of l-th layer. $\tilde{A} \in \mathbb{R}^{N\times N}$ is the adjacency matrix with self-loops. $\tilde{D} \in \mathbb{R}^{N\times N}$ denotes the degree matrix of $\tilde{A}$. $\Theta^{(0)} \in \mathbb{R}^{F\times F'}$ and $\Theta^{(l)} \in \mathbb{R}^{F'\times F'}(l >= 1)$ are learnable convolution weights. σ is a non-linear function that follows each graph convolution layer. Afterwards, we concatenate the output of each output layer. The final node representation $X^{cat} \in \mathbb{R}^{N\times nF'}$ is obtained by concatenating each of the n convolution blocks.

$$X^{cat} = X^{(1)} \| X^{(2)} \| \ldots \| X^{(n)} \tag{2}$$

where $\|$ denotes the concatenation. Because we have two graphs as a pairwise input, two node representations $(X^{cat}_{\mathcal{A}}, X^{cat}_{\mathcal{B}})$ are obtained by a graph convolution Siamese network.

C. CO-ATTENTION GRAPH POOLING

After individually obtaining the representations for the pair of input graphs, our CAGPool layer takes the two graph representations $((X^{cat}_{\mathcal{A}}, A_{\mathcal{A}})$ and $(X^{cat}_{\mathcal{B}}, A_{\mathcal{B}}))$ after several graph convolution layers and then constructs the co-attention vector $\vec{\alpha}$ to calculate node scores. Then, the subgraphs are extracted by indexing each graph with the node scores. Below, we cover details of how CAGPool works.

1) OBTAINING CO-ATTENTION VECTOR

The co-attention vector $\vec{\alpha}$ is obtained from the two graph-level representations of the input graph pair $(x_{\mathcal{A}}, x_{\mathcal{B}})$, where each graph-level representation $x \in \mathbb{R}^{nF'}$ is obtained by global graph pooling. In this paper, we chose global mean pooling

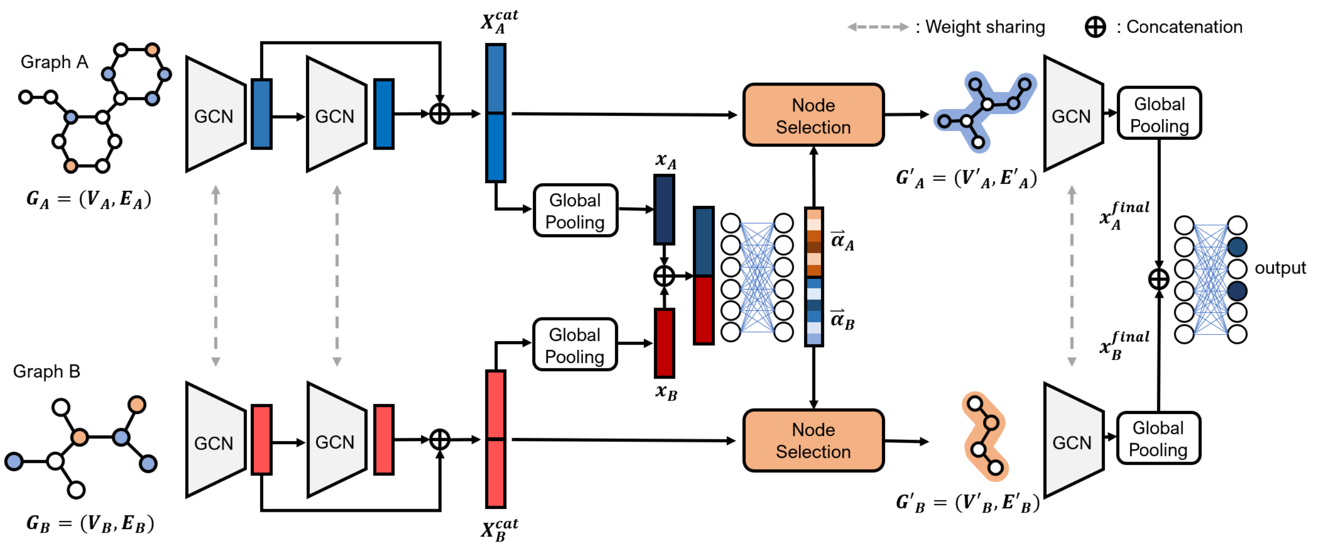


FIGURE 2. **The overall architecture of GNN with Co-Attention Graph Pooling.**

as

$$x = \frac{1}{N}\sum_{r=1}^{N} X_r^{cat}, \tag{3}$$

where $X_r^{cat} \in \mathbb{R}^{nF'}$ is a r-th row vector of node representation matrix $X^{cat} \in \mathbb{R}^{N\times nF'}$. Other global graph pooling methods can also be applied.

The graph-level representations are then concatenated and fed into a linear transformation layer to extract the co-attention vector as

$$\vec{\alpha} = W_\alpha[x_\mathcal{A} \| x_\mathcal{B}] + b_\alpha, \tag{4}$$

where W_α and b_α are both trainable parameters with dimension of $W_\alpha \in \mathbb{R}^{2nF'\times 2nF'}$ and $b_\alpha \in \mathbb{R}^{2nF'}$. The extracted co-attention vector is then indexed as $\vec{\alpha}_\mathcal{A} = \vec{\alpha}_{0:nF'} \in \mathbb{R}^{nF'}$ and $\vec{\alpha}_\mathcal{B} = \vec{\alpha}_{nF':2nF'} \in \mathbb{R}^{nF'}$. The multilayer perceptron (MLP) can also be used instead of a single linear transformation.

2) NODE SELECTION USING CO-ATTENTION VECTOR

The score of r-th node $Z_r \in \mathbb{R}$ is calculated by dot product of $X_r^{cat} \in \mathbb{R}^{nF'}$ and $\vec{\alpha} \in \mathbb{R}^{nF'}$. All node scores $Z \in \mathbb{R}^N$ can be calculated as

$$Z = \frac{X^{cat}\cdot\vec{\alpha}}{\|\vec{\alpha}\|}, \quad \text{idx} = \text{TopK}(Z, \lceil kN \rceil) \tag{5}$$

where $\cdot$ denotes dot product, TopK($\cdot$) function returns the indices of top $\lceil kN \rceil$ nodes according to Z, and $k \in (0, 1]$ is a pooling ratio. We hold $\text{idx}_\mathcal{A}$ from $Z_\mathcal{A}$ and $\text{idx}_\mathcal{B}$ from $Z_\mathcal{B}$ for graphs $G_\mathcal{A}$ and $G_\mathcal{B}$, respectively.

Then, following the procedure of *pooling by node selection* methods, we treat each node score as the significance of the corresponding node and select the top $\lceil kN \rceil$ nodes by indexing with idx as

$$X' = X_{\text{idx},:}^{cat} \odot Z_{\text{idx}}, \quad A' = A_{\text{idx},\text{idx}}, \tag{6}$$

where $(\cdot)_{\text{idx}}$ denotes the indexing operation and $\odot$ is the broadcasted elementwise product. $X' \in \mathbb{R}^{\lceil kN \rceil\times nF'}$ and $A' \in \mathbb{R}^{\lceil kN \rceil\times\lceil kN \rceil}$ are the node feature matrix and the adjacency matrix of a pooled graph $G' = (V', E')$, respectively. Since $\vec{\alpha}$ is obtained from both $G_\mathcal{A}$ and $G_\mathcal{B}$, sub-graphs are extracted according to the interaction representation earlier denoted as $I_g(G_\mathcal{A}, G_\mathcal{B})$. Now we have $(X'_\mathcal{A}, A'_\mathcal{A})$ for sub-graph $G'_\mathcal{A}$ and $(X'_\mathcal{B}, A'_\mathcal{B})$ for sub-graph $G'_\mathcal{B}$.

3) PREDICTION USING SUB-GRAPHS

The sub-graphs obtained by Equation (6) are embedded again with graph convolution as

$$X'^{(l+1)} = \sigma(\tilde{D}'^{-\frac{1}{2}}\tilde{A}'\tilde{D}'^{-\frac{1}{2}}X'^{(l)}\Theta'^{(l)}). \tag{7}$$

To feed to the final MLP layer, we convert the arbitrary sized representations from Equation (7) into fixed-size vectors $x_\mathcal{A}^{final}, x_\mathcal{B}^{final}$ using the global graph pooling described in Equation (3). Then we concatenate the two graph-level representations and feed to the MLP layers as

$$\text{output} = \text{MLP}([x_\mathcal{A}^{final} \| x_\mathcal{B}^{final}]). \tag{8}$$

Depending on the task, appropriate functions (e.g. sigmoid, softmax) are applied to the final output.

IV. EXPERIMENTS

We evaluate our method under two tasks: pairwise graph classification and pairwise graph regression. In this section, we first outline the experimental settings and describe the datasets used to evaluate each task. Following that, we quantitatively compare our proposed CAGPool with the baseline methods to validate its effectiveness. During the training

TABLE 1. AUROC results on the Decagon test set. The ✓ under the feature+ column indicates that the according baseline leverages external features other than the drug itself such as protein-protein interaction or the one-hot encoding of the drug-drug interaction (the target class).

Methods	feature+	AUROC
Concatenated features	✓	0.793
Decagon	✓	0.872
Late-Outer	✓	0.724
CADDI	✓	0.778
MHCADDI	✓	0.882
MPNN-Concat	-	0.661
Drug-Fingerprints	-	0.744
RESCAL	-	0.693
DEDICOM	-	0.705
DeepWalk	-	0.761
MHCADDI-ML	-	0.819
Ours	-	**0.891**

TABLE 2. Evaluation results under AUPRC and AP@50 on the Decagon test set for a more precise comparison with baselines that provide their performance under these metrics.

Methods	AUROC	AUPRC	AP@50
RESCAL	0.693	0.613	0.476
DEDICOM	0.705	0.637	0.567
DeepWalk	0.761	0.737	0.658
Concatenated features	0.793	0.764	0.712
Decagon	0.872	0.832	0.803
Ours	**0.891**	**0.867**	**0.812**

process, we use the Adam Optimizer with a learning rate of 1e-3 for 20 epochs and a batch size of 32. We fix the hidden dimension F' via grid search across 32, 64, 128, and 256. All models, including the baselines, were implemented using PyTorch [25] and PyTorch Geometric [26]. We carried out all experiments on a single NVIDIA Titan Xp GPU. The reported performance represents an average of the results from five repetitions.

A. TASKS AND DATASETS

1) DRUG-DRUG INTERACTION PREDICTION

Polypharmacy refers to the simultaneous use of multiple drugs by a single patient to treat one or more conditions. One of the key challenges with polypharmacy is that patients may experience unexpected side-effects when they take several drugs concurrently. These side-effects can lead to potentially devastating clinical and financial outcomes, including post-marketing withdrawal of drugs from the market [27]. Consequently, it is of paramount importance to accurately predict potential DDIs of drug candidates during the drug discovery process [28].

We utilized DDI dataset collected and used by Decagon [29]. Decagon adopts a pathway-based approach, integrating Protein-Protein, Drug-Protein, and Drug-Drug interaction networks into a singular graph structure. In this structure, each node represents a drug or protein, while the links indicate the interactions between these entities. Following the experimental setup of [14], we exclusively used the Drug-Drug Interaction subset of this dataset. In our representation, each drug is depicted as an individual graph with atoms serving as nodes and chemical bonds as edges. The structural information for each drug compound is obtained using Rdkit.[2] The ultimate goal of our experiment is to predict all types of DDI (if any) that might occur, using solely the structural information of two given drug compounds.

We adopt the exact filtering steps employed by the baselines [14], [29] on the Decagon dataset, including negative sampling. A detailed explanation of the negative sampling process is provided in our supplementary material. We use the 964 types of polypharmacy side effects that occur more than 500 times. The complete dataset consists of 4,576,785 positive examples. We allocate 80% of the interactions to the training set, 10% to the validation set, and the remaining 10% to the test set. The evaluation results are reported on the test set, for the model that achieved the best performance on the validation set. During the testing phase, we calculate the AUROC, AUPRC, and AP@50 across the 964 drug-drug interaction classes, exclusively for valid samples (either positive samples or those obtained via negative mining).

2) GRAPH SIMILARITY PREDICTION

Graph retrieval is a fundamental problem which involves calculating the distance or similarity between two graphs. The Graph Edit Distance (GED) is the most widely used distance metric for graph retrieval [13], [16], [30], [31], [32], [33]. The edit distance is defined as the minimum number of operations needed to transform $G_{\mathcal{A}}$ into $G_{\mathcal{B}}$. However, computing the GED is known to be an NP-complete problem [34], making it infeasible to calculate the exact GED within a reasonable time frame for graphs that have more than 16 nodes [35].

Recently, there are attempts to approximate GED by using GNNs [16]. The authors also provide GED datasets containing AIDS, LINUX, and IMDB.[3] *AIDS*[4] is commonly used in graph similarity search [13], [31], [32], [33]. AIDS dataset contains chemical compound structure graphs with labeled nodes. Bai et al. [16] selected 700 graphs of equal or less than 10 nodes each. *LINUX* [36] dataset is about program dependence graphs generated from the Linux kernel. In the program function graphs of the LINUX dataset, a node represents a statement and an edge is a dependency between two statements. Bai et al. [16] selected 1000 graphs, each of which has equal or less than 10 nodes. For both AIDS and LINUX datasets, the ground truth GEDs are calculated by using the A^* algorithm. *IMDB* [37] dataset contains 1500 graphs of which the nodes represent actors or actresses. Edges denote that two people appear in the same movie. As in SimGNN, we use all the graphs in IMDB dataset for testing the scalability. For the IMDB dataset, the approximation

[2]http://www.rdkit.org

[3]https://github.com/yunshengb/SimGNN
[4]https://wiki.nci.nih.gov/display/NCIDTPdata

TABLE 3. The experimental results of graph similarity regression task. The evaluation metrics are Mean Square Error (MSE, 10^{-3}), Spearman's rank correlation coefficient ρ, and Kendall's rank correlation coefficient τ.

		AIDS			LINUX			IMDB-M		
Methods	**Complexity**	MSE (↓)	ρ (↑)	τ (↑)	MSE (↓)	ρ (↑)	τ (↑)	MSE (↓)	ρ (↑)	τ (↑)
SimGNN	$O(\max(\lvert V_A\rvert, \lvert V_B\rvert)^2)$	**1.189**	0.843	0.690	1.509	0.939	**0.830**	1.264	0.878	0.770
GraphSim	$O(\lvert V_A\rvert\lvert V_B\rvert)$	1.919	0.849	-	0.471	0.976	-	0.743	0.926	-
GMN	$O(\lvert V_A\rvert\lvert V_B\rvert)$	1.886	0.751	-	1.027	0.933	-	4.422	0.725	-
MPNGMN	$O(\lvert V_A\rvert\lvert V_B\rvert)$	1.191	0.904	**0.749**	1.561	0.945	0.814	1.331	0.889	-
GENN	$O(\lvert V_A\rvert\lvert V_B\rvert)$	1.618	0.901	-	0.438	0.955	-	0.883	0.880	-
GED-CDA	$O(\lvert V_A\rvert\lvert V_B\rvert + \lvert V_A\rvert^2 + \lvert V_B\rvert^2)$	1.367	**0.914**	-	**0.125**	**0.985**	-	**0.711**	**0.919**	-
S-Mean	$O(\lvert E_A\rvert + \lvert E_B\rvert)$	3.115	0.633	0.480	16.950	0.020	0.016	3.749	0.774	0.644
H-Mean	$O(\lvert E_A\rvert + \lvert E_B\rvert)$	3.046	0.681	0.629	6.431	0.430	0.525	5.019	0.456	0.378
H-Max	$O(\lvert E_A\rvert + \lvert E_B\rvert)$	3.396	0.655	0.505	6.575	0.879	0.740	6.993	0.455	0.354
Att.Deg.	$O(\lvert E_A\rvert + \lvert E_B\rvert)$	3.338	0.628	0.478	8.064	0.742	0.609	2.144	0.828	0.695
Att.GC	$O(\lvert E_A\rvert + \lvert E_B\rvert)$	1.472	0.813	0.653	3.125	0.904	0.781	3.555	0.684	0.553
Att.LGC	$O(\lvert E_A\rvert + \lvert E_B\rvert)$	**1.340**	0.825	0.667	2.055	0.916	0.804	1.455	0.835	0.700
SGNN	$O(\lvert E_A\rvert + \lvert E_B\rvert)$	2.822	0.765	0.588	11.832	0.566	0.404	1.430	0.870	-
CAGPool (Ours)	$O(\lvert E_A\rvert + \lvert E_B\rvert)$	2.959	**0.880**	**0.717**	**0.658**	**0.988**	**0.922**	**0.884**	**0.975**	**0.874**

algorithms, Beam [38], Hungarian [39], and VJ [40], were used for the ground truth because the IMDB dataset contains graphs with more than 16 nodes. GED is converted to similarity score S with normalization ($nGED$) as follow: $nGED(G_{\mathcal{A}}, G_{\mathcal{B}}) = \frac{GED(G_{\mathcal{A}},G_{\mathcal{B}})}{(|G_{\mathcal{A}}|+|G_{\mathcal{B}}|)/2}$, $S(GED(G_{\mathcal{A}}, G_{\mathcal{B}})) = \exp^{-nGED(GED(G_{\mathcal{A}},G_{\mathcal{B}}))}$,where $|G_i|$ denotes the number of nodes of graph G_i and the similarity score S is in the range of (0,1].

We exactly follow their training/testing data split and randomly select validation set within a training set with the same proportion. All the graphs are split into 60%, 20%, and 20% as a training set, a validation set, and a testing set.

B. EVALUATION AND BASELINES

1) DRUG-DRUG INTERACTION PREDICTION

In the DDI prediction task, we assess the effectiveness of CAGPool by comparing it with existing methods using the Decagon dataset. We primarily employ three evaluation metrics used in Decagon [29]: AUROC, AUPRC, and AP@50.

Table 1 presents a comparison of our network with existing baseline networks on the Decagon dataset. The ✓in the **feature+** column indicates that the method uses additional information beyond the structural information of drugs. The **Concatenated features** method [29] employs a PCA representation of the drug-target protein interaction matrix and individual drug side-effects. The **Decagon** method [29] further includes protein-protein interactions, drug-protein target interactions, and single drug side-effect information. Methods such as **MPNN-Concat**, **Late-Outer**, **CADDI**, **MHCADDI** [14] use one-hot encoding of interactions to predict their presence or absence. Our model outperforms all these methods, whether they use additional information or not, by only leveraging the structural features of the graph representation for drugs.

Table 2 presents a performance comparison across AUROC, AUPRC, and AP@50 metrics on the Decagon dataset. Not only does our method outperform the baseline methods proposed in [29], but it also demonstrates consistently superior performance across all evaluation metrics. Both **RESCAL** and **DEDICOM** are tensor decomposition approaches applied to the drug-drug matrix, while **DeepWalk** employs neural embedding based on a random walk procedure.

2) GRAPH SIMILARITY PREDICTION

Table 3 presents the regression performance on GED datasets. The baselines encompass GNN approaches reported in the GED-CDA paper [41] and the SimGNN paper [16]. **SimGNN** [16], **GraphSim** [42], **GMN** [15], **MPNGMN** [43], **GENN** [44], and **GED-CDA** [41] are characterized by their focus on node-wise interactions. This results in a substantial computational burden due to their high complexity, which is at least $O(|V_A||V_B|)$. For instance, **SimGNN** combines *AttLearnableGC* and *Pairwise Node Comparison*. To account for graph-graph interactions, *SimGNN* utilizes the histogram information of the dot product of all node pairs between two graphs, an approach known as *Pairwise Node Comparison*. **SimpleMean (S-Mean)** generates a graph-level embedding by averaging the node representations. Both **HierarchicalMean (H-Mean)** and **HierarchicalMax (H-Max)** employ a graph coarsening algorithm for hierarchical graph representations [45], and then apply global mean and max pooling, respectively. In **AttDegree (Att.Deg.)**, the attention weight of nodes is calculated using the natural log. Both **AttGlobalContext (Att.GC)** and **AttLearnableGC (Att.LGC)** compute the attention weights using the graph-level representations. However, the latter also incorporates a learnable non-linear transformation, unlike the former. SGNN [43], a Siamese architecture with GCN, has its performance reported in the GED-CDA paper. For a fair comparison, we maintain the model architecture of SimGNN and only replace its *Pairwise Node Comparison* module with CAGPool.

V. DISCUSSION

A. EFFECTIVENESS OF CONSIDERING THE INTERACTION REPRESENTATION

The Graph convolution Siamese network [31] provides a basic framework to deal with pairwise graph inputs by

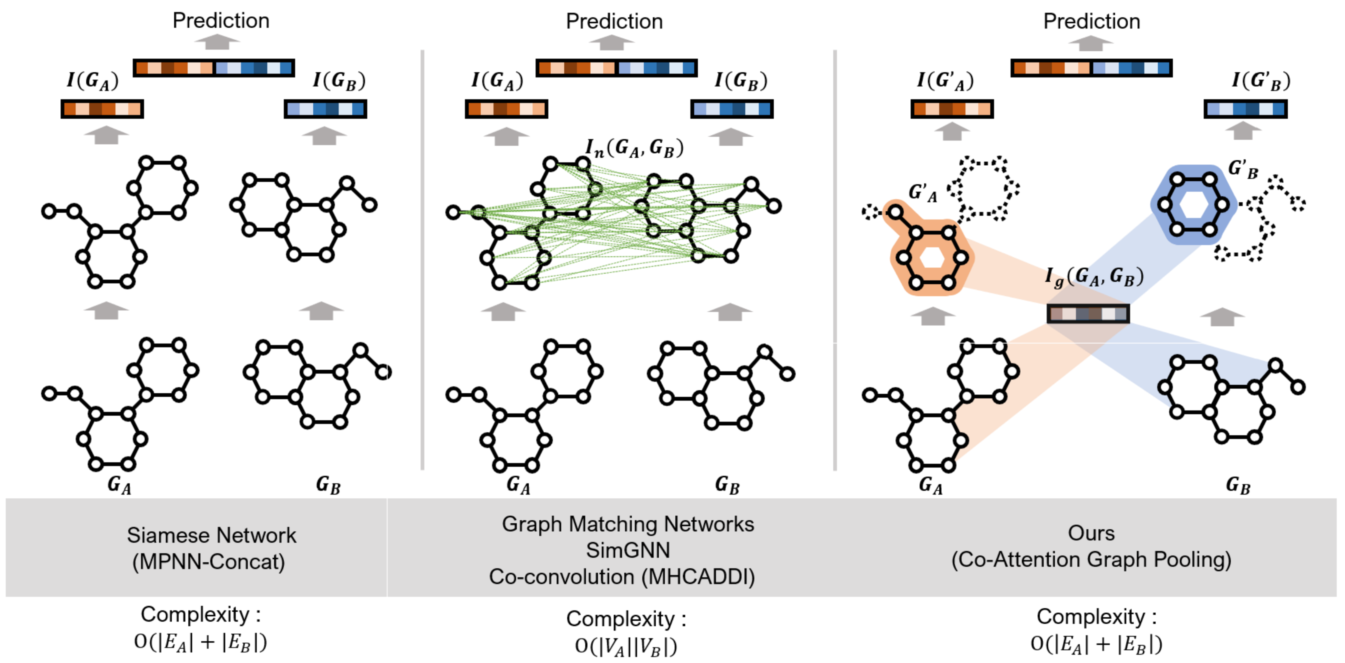


FIGURE 3. Comparison of the complexity between GNNs for pairwise graph inputs. Given a pair of graphs $G_{\mathcal{A}}$ and $G_{\mathcal{B}}$, the graph Siamese network (left) has complexity $O(|E_{\mathcal{A}}| + |E_{\mathcal{B}}|)$, but cannot embed the graph-graph interaction. Although SimGNN and MHCADDI (middle) can consider the interaction $I_n(G_{\mathcal{A}}, G_{\mathcal{B}})$ in the node-level, the complexity increases to $O(|V_{\mathcal{A}}||V_{\mathcal{B}}|)$. CAGPool (right) is able to consider the interaction representation $I_g(G_{\mathcal{A}}, G_{\mathcal{B}})$ in the graph-level while maintaining the complexity upper bound of the graph Siamese network. In our notations, $I(G'_{\mathcal{A}})$, $I(G'_{\mathcal{B}})$, and $I_g(G_{\mathcal{A}}, G_{\mathcal{B}})$ are aligned with vectors $x_{\mathcal{A}}^{final}$, $x_{\mathcal{B}}^{final}$, and $\tilde{\alpha}$, respectively.

predicting the interaction from individually embedded inputs (i.e., $I(G_{\mathcal{A}})$ and $I(G_{\mathcal{B}})$). However, as the interaction is only minimally reflected in the final prediction layer, previous works have extended the scope to leverage interaction representations (i.e., $I_n(G_{\mathcal{A}}, G_{\mathcal{B}})$) during the embedding stage. *MHCADDI* exchanges node representations between two graphs in the message-passing step, while *SimGNN* compares all node pairs of two graphs after the node embedding layers. Since *MHCADDI* and *SimGNN* show a substantial gain in predictive power compared to simple Siamese networks (see Table 1 and Table 3), it can be concluded that utilizing $I(G_{\mathcal{A}}, G_{\mathcal{A}})$ in the embedding stage is crucial when dealing with pairwise graphs. However, as these methods utilize all node pairs, unnecessary complexity and redundant information are produced. This results in suboptimal performance and efficiency, thereby compromising the final prediction. Our CAGPool focuses on subgraphs that are extracted by the interaction representation, mitigating the issue of noisy representations. The experimental results in Table 1 and Table 3 show that CAGPool successfully utilizes the interaction representations at the graph-level $I_g(G_{\mathcal{A}}, G_{\mathcal{A}})$, achieving state-of-the-art performance on both tasks.

B. THE EFFICIENCY OF CO-ATTENTION GRAPH POOLING

Figure 3 illustrates the complexity of each method that treats pairwise graph input. The Graph convolution Siamese network, being the most basic form, has an overall complexity bounded by the complexity of embedding each graph with GNNs: $O(|E_{\mathcal{A}}| + |E_{\mathcal{B}}|)$. Recent methods, such as *MHCADDI* and *SimGNN*, construct interaction representations at the node-level and require pairwise calculations for every node pair between $V_{\mathcal{A}}$ and $V_{\mathcal{B}}$. Therefore, the computational complexity is bounded by $O(|V_{\mathcal{A}}||V_{\mathcal{B}}|)$. Strictly speaking, *SimGNN* has a complexity of $O(\max(|V_{\mathcal{A}}|, |V_{\mathcal{B}}|)^2)$. On the other hand, our approach requires additional computation with a complexity of $O(|V_{\mathcal{A}}| + |V_{\mathcal{B}}|)$ for constructing the interaction representation at the graph-level and selecting the nodes. Therefore, CAGPool maintains the $O(|E_{\mathcal{A}}| + |E_{\mathcal{B}}|)$ complexity of the simple graph convolution Siamese network without increasing the complexity upper bound. Additionally, we only need $W_\alpha \in \mathbb{R}^{2nF' \times 2nF'}$ and $b_\alpha \in \mathbb{R}^{2nF'}$ (see Equation (4)) as additional trainable parameters, where n is the number of GCN layers and F' is the hidden dimension. Given inputs X_A and X_B, our module produces X'_A and X'_B, demonstrating a $31.2 \sim 64.7\%$ faster running time than the node-level interaction module when we set the number of nodes from 50 to 200. We describe the details in the supplementary material.

C. ABLATION STUDY

We conduct an ablation study to validate (1) whether co-attention serves as an effective component when treating pairwise graphs, and (2) whether CAGPool demonstrates a meaningful improvement compared to individual pooling without considering interaction representation.

For (1), it can be concluded that leveraging co-attention is effective by comparing *MHCADDI-ML* and CAGPool with

TABLE 4. Ablation studies and comparison with other pooling methods on the Decagon test set. Note that the scope of this ablation study only includes methods that do not use additional information such as protein features or one-hot encoding of the interaction.

Methods	GNN	Co-att	Pooling	AUROC
MPNN-Concat	✓	-	-	0.661
MHCADDI-ML	✓	✓	-	0.819
TopKPool	✓	-	✓	0.847
SAGPool	✓	-	✓	0.862
Ours	✓	✓	✓	**0.891**

MPNN-Concat (the vanilla Siamese Network architecture in Figure 3). Moreover, CAGPool shows a significant improvement in performance even when compared to *MHCADDI-ML*, implying that CAGPool is a more effective way to utilize the co-attention mechanism.

For (2), we only consider hierarchical pooling methods with node selection to match our experimental settings. For a fair comparison with other hierarchical node selection-based pooling methods, we implemented TopKPool [20] and SAGPool [7] and kept the pooling ratio at 50%. Although all pooling methods show a gain in performance, our co-attention-based approach serves as the most effective pooling method for pairwise graph prediction.

D. FUTURE WORKS

1) STUDIES ON THE EXTRACTED SUBGRAPHS

The main focus of our work is the proposal of a novel pooling method, CAGPool, for extracting subgraphs from graph pairs, which we refer to as $G'_{\mathcal{A}} = (V'_{\mathcal{A}}, E'_{\mathcal{A}})$ and $G'_{\mathcal{B}} = (V'_{\mathcal{B}}, E'_{\mathcal{B}})$, that can help predict labels such as drug-drug interactions. Even if some nodes, $v \in V'$, are isolated, they contain important information due to the use of GCN layers. In the context of predicting drug-drug interactions, it can be extremely difficult to disambiguate the functional groups (i.e., subgraphs) related to a specific side effect, even for experts in the biomedical field. This is because these subgraphs do not directly interact with each other, but rather affect each other through complex biological pathways within the human body. While our research did not specifically investigate this analysis, we expect that our method can facilitate future research in this area by identifying subgraphs that are likely to be related to the functional groups responsible for the side effects between drugs.

VI. CONCLUSION

In this paper, we describe the Co-Attention Graph Pooling (CAGPool) method for processing pairs of graph-structured data in an efficient and effective way. CAGPool combines the co-attention mechanism and pooling by node selection to enable a graph neural network to identify important subgraphs for prediction tasks while maintaining the computational simplicity of Siamese networks. We demonstrate the effectiveness of our approach on two real-world benchmark datasets, showing that it outperforms baselines even without using additional information commonly utilized by other methods. We believe that CAGPool has the potential to be beneficial in a wide range of applications, including the urgent development of COVID-19 treatments, where the ability to identify key sub-graphs in paired data can help prevent unintended side effects.

APPENDIX A DATASETS

In this paper, we validate CAGPool on two tasks: a *classification task* for Drug-Drug Interaction (DDI) prediction and a *regression task* on the Graph Edit Distance (GED) dataset. Here, we provide additional information on the preprocessing details and analysis for each dataset.

A. DRUG-DRUG INTERACTION DATASET

As described in the main text, we utilized the DDI dataset assembled by Decagon [29]. The original Decagon dataset contains Protein-Protein, Drug-Protein, and Drug-Drug pairs integrated into a graph structure. In this structure, each node represents either drugs or proteins, and the links indicate interactions between the two corresponding vertices. For our research, we focused solely on the drug-drug interaction subset. Each drug is represented as a single graph with atoms as nodes and chemical bonds as edges. The node attributes include the type of atoms, polarity, number of hydrogen atoms, and aromaticity. The original literature filtered for 500 or more drug pairs, but after our preprocessing, we found that to align with the 964 interaction types, we had to filter for 498 or more pairs.

Negative Sampling To compensate for TWOSIDES, which only contains positive samples, we adopt the *negative sampling* approach used in previous works [14], [29]. The detailed process involving a drug d and side-effect se is described below:

- During training, tuples $(\tilde{d_x}, \tilde{d_y}, se_z)$, where $\tilde{d_x}$ and se_z are chosen from the dataset and $\tilde{d_y}$ is chosen at random from the set of drugs difference from d_y in the true samples $(\tilde{d_x}, d_y, se_z)$. The negative sample is selected randomly according to sampling distribution P_r, where for each node d_i has a probability of $P(d_i) = \frac{f(d_i)^{3/4}}{\sum_{j=0}^{n} f(d_j)^{3/4}}$ of appearing [46].
- During validation and testing, we randomly sample two distinct drugs which do not appear in the positive dataset.

B. GRAPH EDIT DISTANCE DATASET

We utilized the GED dataset from SimGNN[5] [16]. The GED dataset comprises three sub-datasets: AIDS, LINUX, and IMDB. These datasets were designed and collected to evaluate the performance of the *graph retrieval* task, a task designed to find similar/dissimilar graphs from the database when given a query graph.

[5]https://github.com/yunshengb/SimGNN

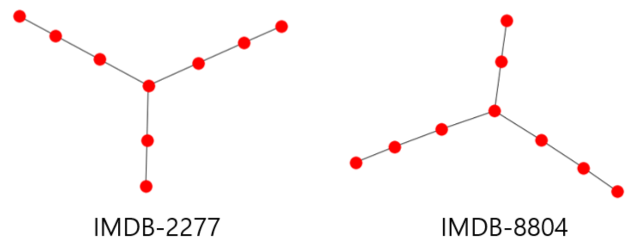


FIGURE 4. An example of isomorphic graphs. IMDB-2277 and IMDB-8804 are from different instance but have a same graph structure and node features. The graph edit distance between them is zero.

We found that some graphs in GED datasets were in an equivalence relation (i.e. isomorphism) (see Figure 4). According to SimGNN authors, they treat the query graph as an unseen graph even if there exists an isomorphic graph in the database. Because checking for isomorphism is expensive with traditional graph algorithms, an algorithm that can efficiently capture isomorphic graphs is important in this dataset. Since our model is permutation invariant, it can benefit this task by capturing isomorphism efficiently.

APPENDIX B COMPREHENSIVE OVERVIEW OF GRAPH POOLING METHODS

In this section, we 1) explain the categorization of graph pooling methods and 2) compare them in terms of complexity. Hierarchical graph pooling methods can be classified into *pooling by graph transformation* and *pooling by node selection*. The overview is illustrated in Figure 5. There are various pooling methods based on graph algorithms such as spectral clustering [45], but we only consider graph pooling methods that can be trained end-to-end. Our approach is based on the *pooling by node selection* method.

Given the input graph $G = (V, E)$ with the set of vertices V and the set of edges E, the hierarchical graph pooling can be represented as the function $g : G \mapsto G'$, where $G' = (V', E')$ is the small-size graph. Graph G has the node feature matrix $X \in \mathbb{R}^{N\times F}$ and the adjacency matrix $A \in \mathbb{R}^{N\times N}$, where N is the number of nodes and F is the feature dimension. The final goal of the graph pooling method is to obtain the graph $G' = (V', E')$ that has the node features $X' \in \mathbb{R}^{N'\times F}$ and the adjacency matrix $A' \in \mathbb{R}^{N'\times N'}$. We categorize the graph pooling methods according to how X' and A' are calculated.

C. POOLING BY GRAPH TRANSFORMATION

Pooling by graph transformation (also called as pooling by clustering) downsamples graph by learning the transformation matrix. Following the work of Ying et al. [6], pooling methods in this category calculate the output node features X' and the adjacency matrix A' by using the transformation matrix $S \in \mathbb{R}^{N\times N'}$, which is called as the assignment matrix [6]. Here, the matrix S transforms not only nodes, but also edges. Therefore, how we define S is the main key in *pooling by graph transformation*. In DiffPool, the transformation matrix S is obtained from the node embedding of Graph Neural Networks (GNNs). In StructPool, the transformation matrix S is trained via conditional random fields [8]. Briefly, *pooling by graph transformation* methods have a common framework as

$$X' = S^\top X, \quad A' = S^\top AS, \tag{9}$$

where $X' \in \mathbb{R}^{N'\times F}$ is the feature matrix of N' cluster nodes and $A' \in \mathbb{R}^{N'\times N'}$ is the adjacency matrix of them. Note that $G' = (V', E')$ is not a sub-graph of G (i.e. $V' \not\subset V, E' \not\subset E$).

1) COMPLEXITY ISSUE OF GRAPH TRANSFORMATION

Despite the improvement in their performance, pooling by graph transformation suffers from heavy computational cost when obtaining the new adjacency matrix A'. DiffPool suffers from heavy computational complexity because the S and the output graph is represented as a dense matrix. Although S of StructPool can be either dense or sparse according to their setting, it still suffers from heavy complexity because of the iterative method for S and the calculation of A' described in Equation (9).

D. POOLING BY NODE SELECTION

In *Pooling by node selection*, X' and its adjacency matrix A' are obtained by selecting the nodes according to the node score $Z \in \mathbb{R}^N$, leaving nodes with high scores and discarding the rest. Because both X' and A' are simply calculated by indexing, these methods do not increase the upper bound of computational complexity in GNNs. Pooling by node selection therefore eventually boils down to "how we define the scoring function for each node? (i.e., how we define Z?)" TopKPool calculate node scores Z from the dot product of node features and a trainable projection vector [20]. SAGPool exploit both node features and the graph topology to calculate node scores Z [7]. After the calculation of Z, both perform the indexing operation as

$$X' = X_{\text{idx},:}, \quad A' = A_{\text{idx},\text{idx}}, \tag{10}$$

where $(\cdot)_{\text{idx}}$ denotes the indexing operation and idx is the top-k indices of node scores Z. Unlike *pooling by graph transformation*, G' is a sub-graph of G (i.e. $V' \subset V, E' \subset E$). Because the nodes are explicitly selected, it would be helpful to interpret which local structures are important to increase the predictive power of GNNs.

1) WHY IS CAGPool BASED ON NODE-SELECTION?

As referred in Section VI-C and Section VI-D, pooling by node selection alleviates the computational complexity issue of pooling by graph transformation such as DiffPool. Since the main goal of our model is to design a low-compleixty model that can effectively and efficiently encode the interaction representation between the pair of input graphs, we follow the architectural design of *pooling by node selection*. For instance, if we adapt the graph DiffPool-like CAGPool which is described the below section, the complexity is quadratic

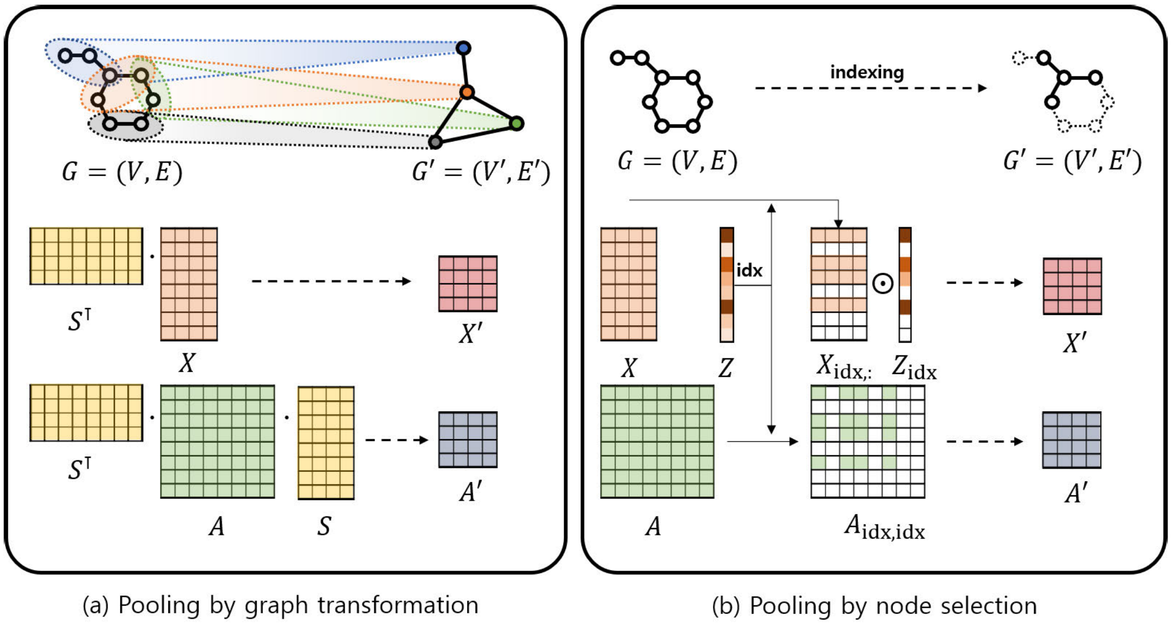


FIGURE 5. The illustration of pooling by graph transformation and *pooling by node selection*. · denotes the matrix multiplication and ⊙ denotes the broadcasted multiplication. For both methods, the input graph $G = (V, E)$ has the node feature matrix $X \in \mathbb{R}^{N \times F}$ and the adjacency matrix $A \in \mathbb{R}^{N \times N}$, where N, F denote the number of nodes and the feature dimension, respectively. In (a) pooling by graph transformation, the output graph $G' = (V', E')$ is generated according to the transformation matrix $S \in \mathbb{R}^{N \times N'}$, where V' is the set of cluster nodes and N' denotes the number of cluster nodes. In (b) pooling by node selection, it is important to define the node score $Z \in \mathbb{R}^{N}$ well. According to the node scores Z, the top-k nodes are selected as elements of $V' \subset V$.

to the number of nodes for each graph, $\mathcal{O}(|V_A|^2 + |V_B|^2)$. Also we can identify the important substructure in the original topology that is difficult in *Pooling by graph transformation*. During the experiments in our main paper, we followed 50% ratio pooling for fair comparison with other *pooling by node selection* methods.

2) EXTENSION OF CAGPool TO A GRAPH-TRANSFORM VERSION

While pooling by node selection can be beneficial in terms of computational complexity and interpretability, it is also true that some nodes are not selected during the pooling process and are therefore discarded. On the other hand, pooling by graph transformation includes all nodes in the final aggregated clusters, without loss of information. CAGPool can be extended to a graph transformation version by setting the assignment matrix as follows:

$$S = h(X_A, X_B) \tag{11}$$

This means that the clustering is dynamically performed based on the representation of the pair of graphs.

APPENDIX C
COMPARISON OF THE RUNNING TIME

To demonstrate the efficiency of our method, we compare the running time of node-level and graph-level interaction modules. Each module accepts X_A and X_B as an input pair and outputs X'_A and X'_B, which represent the pooled node feature matrix. We set the number of nodes from 50 to 200 and repeat the process 10k times to obtain consistent results. The graph-level interaction module produces X'_A and X'_B 31.2 - 64.7 % faster than the node-level interaction module.

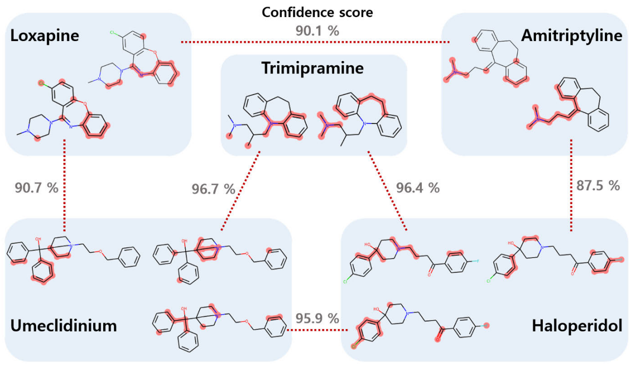


FIGURE 6. Visualization of the top attention score nodes on true-positive samples from the test set. The highlighted areas represent attention patterns based on the pairs. The percentages indicate our prediction values.

APPENDIX D
VISUALIZATION OF THE ATTENTION PATTERNS

Figure 6 showcases a relationship diagram that includes visualization of attention areas for an example of many-to-many interactions. We examined the true-positive cases of our model and highlighted the nodes with the highest attention scores. The percentage above the edge between each structure represents the confidence score of our prediction model. As shown in the figure, each drug is pooled and projected differently based on the drug it interacts with.

ACKNOWLEDGMENT

(Junhyun Lee and Bumsoo Kim contributed equally to this work.)

REFERENCES

[1] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–14.

[2] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lió, and Y. Bengio, "Graph attention networks," *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–12.

[3] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *Proc. 37th Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 1725–1735.

[4] G. Corso, L. Cavalleri, D. Beaini, P. Liò, and P. Veličković, "Principal neighbourhood aggregation for graph nets," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*. Red Hook, NY, USA: Curran Associates, 2020, pp. 13260–13271.

[5] G. Wang, R. Ying, J. Huang, and J. Leskovec, "Multi-hop attention graph neural networks," in *Proc. 13th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 3089–3096.

[6] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 4800–4810.

[7] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *Proc. 36th Int. Conf. Mach. Learn.*, Jun. 2019, pp. 9–15.

[8] H. Yuan and S. Ji, "STRUCTPOOL: Structured graph pooling via conditional random fields," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–12.

[9] J. Baek, M. Kang, and S. J. Hwang, "Accurate learning of graph representations with graph multiset pooling," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–22.

[10] J. Du, S. Wang, H. Miao, and J. Zhang, "Multi-channel pooling graph neural networks," in *Proc. 13th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 1442–1448.

[11] J. Wu, X. Chen, K. Xu, and S. Li, "Structural entropy guided graph hierarchical pooling," in *Proc. 39th Int. Conf. Mach. Learn. (ICML)*, 2022, pp. 24017–24030.

[12] S. M. Ko, S. Cho, D.-W. Jeong, S. Han, M. Lee, and H. Lee, "Grouping-matrix based graph pooling with adaptive number of clusters," in *Proc. 27th AAAI Conf. Artif. Intell. (AAAI)*, 2022, pp. 1–9.

[13] S. I. Ktena, S. Parisot, E. Ferrante, M. Rajchl, M. Lee, B. Glocker, and D. Rueckert, "Distance metric learning using graph convolutional networks: Application to functional brain networks," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* Cham, Switzerland: Springer, 2017, pp. 469–477.

[14] A. Deac, Y.-H. Huang, P. Velickovic, P. Lio, and J. Tang, "Empowering graph representation learning with paired training and graph co-attention," in *Proc. ICLR*, 2020, pp. 1–14.

[15] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli, "Graph matching networks for learning the similarity of graph structured objects," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3835–3845.

[16] Y. Bai, H. Ding, S. Bian, T. Chen, Y. Sun, and W. Wang, "SimGNN: A neural network approach to fast graph similarity computation," in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, Jan. 2019, pp. 384–392.

[17] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5425–5434.

[18] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.

[19] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proc. AAAI Conf. Artif. Inteligence*, 2018, pp. 1–9.

[20] H. Gao and S. Ji, "Graph U-Nets," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2083–2092.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[22] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," in *Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–7.

[23] Y. Pathak, S. Laghuvarapu, S. Mehta, and U. D. Priyakumar, "Chemically interpretable graph interaction network for prediction of pharmacokinetic properties of drug-like molecules," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 873–880.

[24] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *Proc. ICML Deep Learn. Workshop*, Lille, France, vol. 2, 2015, pp. 1–8.

[25] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, and L. Antiga, "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.

[26] M. Fey and J. Eric Lenssen, "Fast graph representation learning with PyTorch geometric," 2019, *arXiv:1903.02428*.

[27] I. J. Onakpoya, C. J. Heneghan, and J. K. Aronson, "Post-marketing withdrawal of 462 medicinal products because of adverse drug reactions: A systematic review of the world literature," *BMC Med.*, vol. 14, no. 1, p. 10, 2016.

[28] K. Han, E. E. Jeng, G. T. Hess, D. W. Morgens, A. Li, and M. C. Bassik, "Synergistic drug combinations for cancer identified in a CRISPR screen for pairwise genetic interactions," *Nature Biotechnol.*, vol. 35, no. 5, pp. 463–474, May 2017.

[29] M. Zitnik, M. Agrawal, and J. Leskovec, "Modeling polypharmacy side effects with graph convolutional networks," *Bioinformatics*, vol. 34, no. 13, pp. i457–i466, Jul. 2018.

[30] H. Bunke, "What is the distance between graphs," *Bull. EATCS*, vol. 20, pp. 35–39, Jun. 1983.

[31] Y. Liang and P. Zhao, "Similarity search in graph databases: A multi-layered indexing approach," in *Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE)*, Apr. 2017, pp. 783–794.

[32] X. Zhao, C. Xiao, X. Lin, Q. Liu, and W. Zhang, "A partition-based approach to structure similarity search," *Proc. VLDB Endowment*, vol. 7, no. 3, pp. 169–180, 2013.

[33] W. Zheng, L. Zou, X. Lian, D. Wang, and D. Zhao, "Graph similarity search with edit distance constraint in large graph databases," in *Proc. 22nd ACM Int. Conf. Conf. Inf. Knowl. Manage.*, 2013, pp. 1595–1600.

[34] H. Bunke and K. Shearer, "A graph distance metric based on the maximal common subgraph," *Pattern Recognit. Lett.*, vol. 19, nos. 3–4, pp. 255–259, Mar. 1998.

[35] D. B. Blumenthal and J. Gamper, "On the exact computation of the graph edit distance," *Pattern Recognit. Lett.*, vol. 134, pp. 46–57, Jun. 2020.

[36] X. Wang, X. Ding, A. K. H. Tung, S. Ying, and H. Jin, "An efficient graph indexing method," in *Proc. IEEE 28th Int. Conf. Data Eng.*, Apr. 2012, pp. 210–221.

[37] P. Yanardag and S. V. N. Vishwanathan, "Deep graph kernels," in *Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2015, pp. 1365–1374.

[38] M. Neuhaus, K. Riesen, and H. Bunke, "Fast suboptimal algorithms for the computation of graph edit distance," in *Proc. Joint IAPR Int. Workshops Stat. Techn. Pattern Recognit. (SPR) Structural Syntactic Pattern Recognit. (SSPR)*. Cham, Switzerland: Springer, 2006, pp. 163–172.

[39] H. W. Kuhn, "The Hungarian method for the assignment problem," *Nav. Res. Logistics Quart.*, vol. 2, nos. 1–2, pp. 83–97, Mar. 1955.

[40] S. Fankhauser, K. Riesen, and H. Bunke, "Speeding up graph edit distance computation through fast bipartite matching," in *Proc. Int. Workshop Graph-Based Represent. Pattern Recognit.* Cham, Switzerland: Springer, 2011, pp. 102–111.

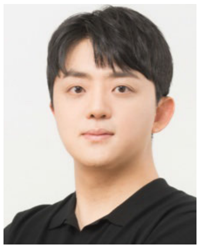
[41] R. Jia, X. Feng, X. Lyu, and Z. Tang, "Graph-graph context dependency attention for graph edit distance," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2023, pp. 1–5.

[42] Y. Bai, H. Ding, K. Gu, Y. Sun, and W. Wang, "Learning-based efficient graph similarity computation via multi-scale convolutional set matching," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 3219–3226.

[43] X. Ling, L. Wu, S. Wang, T. Ma, F. Xu, C. Wu, and S. Ji, "Hierarchical graph matching networks for deep graph similarity learning," in *Proc. ICLR*, 2020, pp. 1–18.

[44] R. Wang, T. Zhang, T. Yu, J. Yan, and X. Yang, "Combinatorial learning of graph edit distance via dynamic embedding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 5237–5246.
[45] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.
[46] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

JUNHYUN LEE received the B.S. degree in biomedical engineering from Korea University, in 2017, where he is currently pursuing the Ph.D. degree in computer science, under the supervision of Prof. Jaewoo Kang. His research interests include deep learning, geometric machine learning, and biomedical applications.



BUMSOO KIM received the B.S. and Ph.D. degrees in computer science from Korea University, Seoul, South Korea, in 2016 and 2022, respectively. During the Ph.D., he was a Research Scientist with Kakao Brain, South Korea, from 2020 to 2022. After the Ph.D., he was a Research Scientist with LG AI Research, South Korea, where his research interests include deep learning, scene understanding, efficient transformers, and large-scale vision-language pretraining

MINJI JEON received the B.S. degree in computer science, the M.S. degree in bioinformatics, as part of the Interdisciplinary Graduate Program, and the Ph.D. degree in computer science from Korea University, Seoul, South Korea, in 2012, 2014, and 2018, respectively. Upon completion of the Ph.D., she held the position of a Research Professor with Korea University, from 2018 to 2019. She was a Postdoctoral Fellow with the Icahn School of Medicine at Mount Sinai, New York, NY, USA, from 2020 to 2022. Since 2022, she has been an Assistant Professor with the Department of Medicine, Korea University College of Medicine, Seoul.

JAEWOO KANG received the B.S. degree in computer science from Korea University, Seoul, South Korea, in 1994, the M.S. degree in computer science from the University of Colorado Boulder, CO, USA, in 1996, and the Ph.D. degree in computer science from the University of Wisconsin–Madison, WI, USA, in 2003. From 1996 to 1997, he was a Technical Staff Member with AT&T Labs Research, Florham Park, NJ, USA. From 1997 to 1998, he was a Technical Staff Member with Savera Systems Inc., Murray Hill, NJ, USA. From 2000 to 2001, he was the CTO and a Co-Founder of WISEngine Inc., Santa Clara, CA, USA, and Seoul. From 2003 to 2006, he was an Assistant Professor with the Department of Computer Science, North Carolina State University, Raleigh, NC, USA. Since 2006, he has been a Professor with the Department of Computer Science, Korea University, where he is the Department Head of the Bioinformatics for Interdisciplinary Graduate Program.