Fast Graph Similarity Computation via Adversarial Knowledge Distillation

Zejian Li
Department of Computer Science and Technology
University of Cambridge
zl525@cam.ac.uk

Abstract

Graph Similarity Computing (GSC) has always been an important task in graph computing. Traditional methods to solve GSC tasks have a high computational complexity, so they are difficult to deploy in real-world industrial scenarios which often have graphs with large sizes and require a strict inference time. Based on an already proposed GNN-based model (EGSC-S[1]) for GSC tasks, this paper is the first attempt to introduce Adversarial Knowledge Distillation (AKD) into the student model (EGSC-S). Experimental results show that our proposed method outperforms existing student models in most scenarios and has a matching inference time. Therefore, this work provides a good direction for the compression and deployment of GSC computational models in industrial scenarios. Our implementation code is available at: https://github.com/leeinscky/EGSC-AKD

1 Introduction

A crucial stage in many machine learning problems, such as classification [2], clustering [3], ranking [4], etc., is learning a function to quantify the distance or similarity between objects [5]. In addition, graph data is widely used in a variety of fields today, including chemistry, bioinformatics, social systems, recommend systems, and more [6]. Therefore, computing graph similarity has been an important task.

Graph similarity computing (GSC) has been studied for many real applications, such as social group identification [7, 8], molecular graph classification in chemoinformatics [9, 10], protein-protein interaction network analysis for disease prediction [11], binary function similarity search in computer security [12], multi-subject brain network similarity learning for neurological disorder analysis [13], etc [5]. To measure the similarity between pair-wise graphs, Graph Edit Distance (GED) [14] has been a major metric due to its generality, and many other graph similarity measures have been shown to be its special cases [15]. As shown in Fig. 1, the edit distance between the source graph $\mathcal{G}_{source}$ and the target graph $\mathcal{G}_{target}$ is GED($\mathcal{G}_{source}$, $\mathcal{G}_{target}$). The value of GED represents the number of edit operations in the optimal alignments that transform $\mathcal{G}_{source}$ into $\mathcal{G}_{target}$, where an edit operation on a graph $\mathcal{G}$ is an insertion or deletion of a vertex/edge or relabelling of a vertex [6]. Intuitively, if the source graph and target graph are identical (isomorphic), GED($\mathcal{G}_{source}$, $\mathcal{G}_{target}$) = 0.

Unfortunately, computing the exact GED is an NP-hard problem [16] in general [17], which requires exponential time complexity and largely limits the application to real-world tasks [18]. To further improve the performance and efficiency of GSL models, several data-driven approximate approaches such as [6, 1, 17, 19, 20, 12] based on graph neural networks (GNNs) have recently been proposed [18].

However, due to the increasing size of the graph data, increasing the accuracy of GSC is computationally expensive and can affect the inference speed. Considering the application of GSC models in

Preprint. Under review.

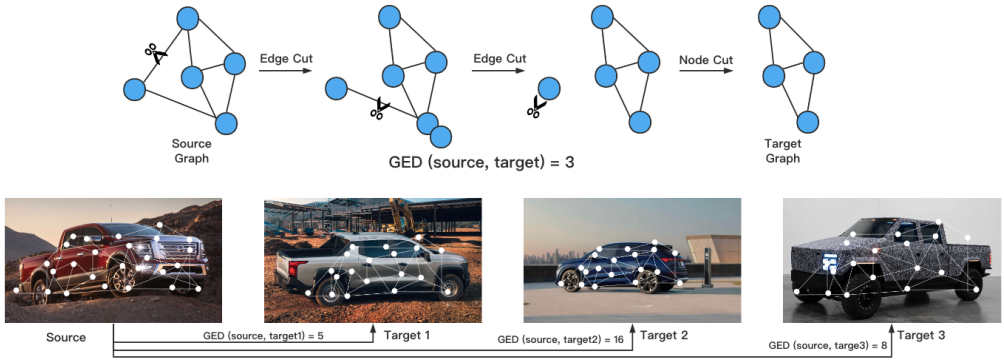

Figure 1: ***Top***: an example of an edit path between the source graph and the target graph. ***Bottom***: GED examples of a source graph with three target graphs.

real industrial scenarios, we want the inference speed of the GSC models to be as fast as possible. To the best of our knowledge, Qin et al. [1] first attempt to apply model compression techniques to GSC, which propose a novel Knowledge Distillation method for GSC where the joint embeddings of input graphs are decomposed to distill for efficient inference and off-line embedding collection. At the same time, He et al. [21] propose the first adversarial knowledge distillation framework for GNNs named GraphAKD. Inspired by GraphAKD, we attempt to incorporate adversarial training into GSC knowledge distillation to further improve the compression and inference accuracy of GSC. In summary, our main contributions are highlighted as follows:

- To the best of our knowledge, we are the first to adopt adversarial training during the knowledge distillation for the GSC model.
- Several comparative experiments have shown that the use of Adversarial Knowledge Distillation can improve the inference accuracy of the GSC model.

2 Related Works

2.1 Graph Similarity Computation (GSC)

The goal of Graph Similarity Computation (GSC) is to quantify the similarity between graphs under a specific similarity measure [17]. Various similarity measures have been well studied in prior work, such as the graph edit distance (GED) [14, 22, 23] and the maximum common subgraph (MCS) [24, 25]. Among these, GED is the most popular one, and many other similarity measures can be proven to be its special cases [15]. However, the computation of these metrics is generally an NP-complete problem [26]. Although some pruning strategies and heuristic methods have been proposed to approximate the values and speed up the computation, it is difficult to examine the computational complexities of the above heuristic algorithms and the sub-optimal solutions provided by them are also unbounded [26, 5]. Thus, it is difficult to apply these methods in real-world tasks since these approaches are only feasible for graphs of relatively small size [5].

However, the recent success in machine learning on non-euclidean data (i.e. graphs) via GNNs [27–29] has encouraged researchers to design approximators for graph similarity measurements such as GED [20]. The main idea is to learn graph representations for candidate graphs with carefully designed graph matching networks [12]. The crucial stage is to introduce the node-node matching scores while predicting the similarity scores [18]. The node-node matching scores are usually introduced via extracted histogram features [6], Convolutional Neural Networks (CNNs) [19], and inter-graph message passing techniques [12, 30–32], serving as vital inductive biases for accurate graph similarity measuring [18]. However, among these methods, only Qin et al. [1] introduce the approaches of model compression (Knowledge Distillation) to speed up inference. Our proposed method also uses a similar knowledge distillation method to compress the GSC model, but our model introduces adversarial training to enhance the KD process.

2.2 Knowledge Distillation for Graph Models

Hinton et al. [33] first propose Knowledge Distillation (KD) where the goal is distilling the knowledge from a teacher model that is often large into a smaller model so that the student model can maintain a similar performance as the teacher. Many distillation strategies, such as [34–36] are designed to deal with the deep convolutional network with grid data as input. In terms of knowledge distillation for graph models, Yang et al. [37] propose the first approach to distilling knowledge from a pre-trained GCN model. A KD model named Reliable Data Distillation [38] is proposed as a reliable data-driven semi-supervised GCN training method that can make better use of high-quality data and improve the graph representation learning by defining node reliability and edge reliability in a graph. However, Yang et al. [39] point out that previous methods cannot fully leverage the important feature-based prior knowledge and they are single models rather than frameworks. Thus, they propose an effective knowledge distillation framework to inject the knowledge of an arbitrarily learned GNN (teacher model) into a well-designed student model. Compared to previous work, He et al. [21] introduce GraphAKD, which adversarially trains a discriminator and a generator to adaptively detect and decrease the discrepancy between teacher and student to compress deep graph neural networks. Inspired by GraphAKD [21], our proposed method first applies adversarial knowledge distillation in the field of graph similarity computation to achieve performance improvement.

3 Approach

3.1 Problem Formulation

We define a graph $\mathcal{G}$ as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with node set $\mathcal{V}$ and edge set $\mathcal{E}$. And a pair of nodes connected by an edge can be represented as $(u, v) \in \mathcal{E}$, where $u \in \mathcal{V}$ and $v \in \mathcal{V}$. In the GSC task, we aim to measure the similarity of a pair of graphs $\mathcal{G}_i$ and $\mathcal{G}_j \in \mathcal{D}$, where $\mathcal{D} = \{\mathcal{G}_0, \mathcal{G}_1, \mathcal{G}_2, ...\}$ is the graph set [1]. The similarity score of two graphs $\mathcal{G}_i$ and $\mathcal{G}_j$ can be represented as the Graph Edit Distance (GED) or Maximum Common Subgraph (MCS), where GED is defined as the number of edit operations in the optimal trajectory to transform the source graph $\mathcal{G}_i$ to the target $\mathcal{G}_j$ and MCS is the maximum common subgraph for both two graphs [1].

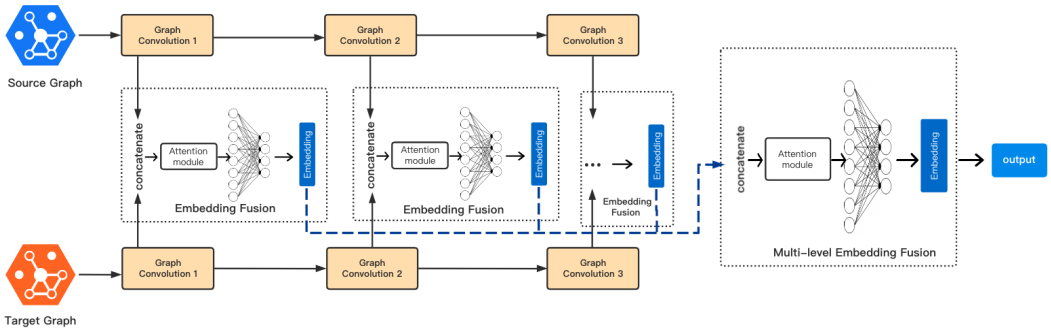

Figure 2: The overview of the GNN-based Feature-fusion Framework (Teacher Model)

3.2 GNN-based Feature-fusion Framework (Teacher Model) for GSC

Our implementation of the GNN-based framework to compute graph similarity is based on the early-feature fusion model in [1]. As shown in Fig. 2, the model has a strong co-attention network and takes the Graph Isomorphism Network (GIN) as the backbone. Overall, the Graph Isomorphism Network (GIN) [40] is used for abstract feature extraction. Then, the features of different levels are encoded within convolution layers. Qin et al. [1] also use an attention layer to enrich the representation ability of the node and graph embeddings and take an MLP for further feature learning.

Specifically, Graph Isomorphism Network (GIN) is beneficial to GSC tasks because graphs isomorphism ($\mathcal{G}_i \simeq \mathcal{G}_j$) also means that the GED is 0: $\mathcal{G}_i \simeq \mathcal{G}_j \leftrightarrow GED(\mathcal{G}_i, \mathcal{G}_j)$. According to the universal approximation theory [41, 42], MLP can be suitable in iterative updating of node features

as:

$$h_v^{(k)} = \mathbf{MLP}^{(k)}\left(\left(1+\epsilon^{(k)}\right)\cdot h_v^{(k-1)} + \sum_{u\in\mathcal{N}(v)} h_u^{(k-1)}\right), \tag{1}$$

where $\mathcal{N}(v)$ is the set of neighbouring nodes of node v, $h_v^{(k)}$ is the node embeddings (features) in the k-th layer, $\epsilon^{(k)}$ can be either learnable or fix parameter [1]. Besides, the 'sum' operation has been verified as the most powerful mapping function in GIN, so we can get the graph's global embedding as:

$$h_{\mathcal{G}} = \text{CONCAT}\left(\text{sum}\left(\left\{h_v^{(k)} \mid v\in\mathcal{G}\right\}\right) \mid k=0,\ldots,K\right), \tag{2}$$

which indicates that the features from layer 0 to layer K are concatenated as the global feature [1].

Moreover, it is necessary to represent the joint and individual embeddings of input graphs to have better graph representation learning. Given graphs A and B as input graphs, the joint embedding of a graph A and graph B is denoted as $h_{AB}^* = E\left(\mathcal{G}_A, \mathcal{G}_B\right)$. And if we have duplicate graphs as input, then we have $h_{AA}^* = E\left(\mathcal{G}_A, \mathcal{G}_A\right)$ and $h_{BB}^* = E\left(\mathcal{G}_B, \mathcal{G}_B\right)$. The authors of [1] assume that the joint embedding might be represented as the linear combination of individual embeddings in the high-dimensional feature space, so the pseudo-individual graph embedding will be computed as $h_{aB}^* = h_{AB}^* - h_{AA}^*$ where h_{aB}^* is supposed to cover all the knowledge of graph B and parts of graph A. And the pseudo-individual graph embedding of graph A is collected in the same way: $h_{Ab}^* = h_{AB}^* - h_{BB}^*$.

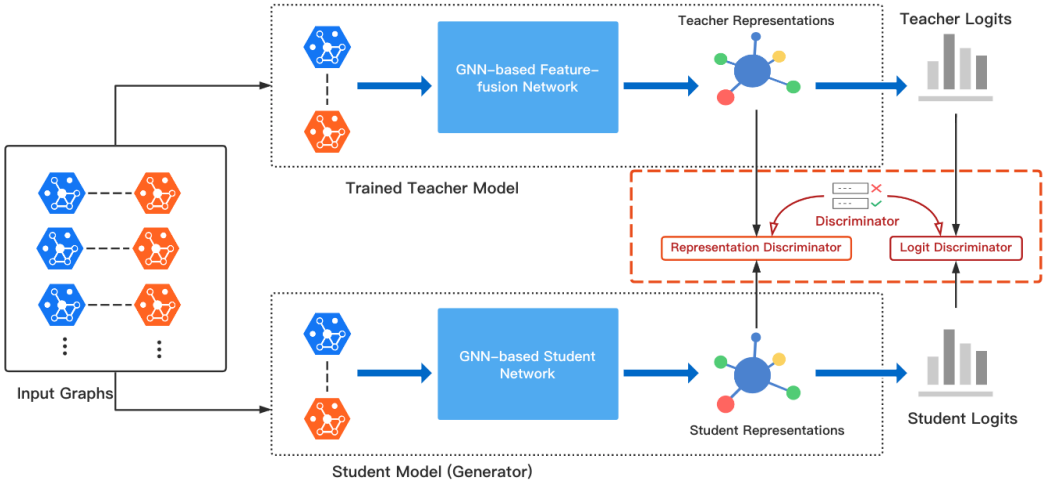

Figure 3: Illustration of the proposed Adversarial Knowledge Distillation (AKD) method

3.3 Adversarial Knowledge Distillation (AKD)

The novelty of our work is that we first adopt the Adversarial Knowledge Distillation (AKD) in the feature-fusion framework for GSC. As a comparison, we start by explaining the original implementation of knowledge distillation in [1], and then we describe how AKD differs from it.

In knowledge distillation, we want the student model to inherit the knowledge from the teacher, so the discrepancy of the pseudo-individual features should be minimised. Thus, the loss of knowledge distillation is:

$$\mathcal{L}_{KD}\left(\mathcal{G}_A, \mathcal{G}_B\right) = \frac{\alpha}{2}\left(\left\|h_{Ab}^T - h_{Ab}^S\right\|_1 + \left\|h_{aB}^T - h_{aB}^S\right\|_1\right) + (1-\alpha) l_\delta\left(\psi_D\left(h_{Ab}^T, h_{aB}^T\right), \psi_D\left(h_{Ab}^S, h_{aB}^S\right)\right), \tag{3}$$

where h_{aB}^T and h_{Ab}^T are the pseudo-individual embeddings of the teacher model, h_{aB}^S and h_{Ab}^S are the pseudo individual embeddings of the student model [1]. $\psi_D\left(h_i, h_j\right) = \left\|h_i - h_j\right\|_1$ is the distance-wise potential function measuring the first order distance in the same domain, and l_δ is the Huber loss [43, 1].

Inspired by [21], we develop an Adversarial Knowledge Distillation (AKD) method as shown in Fig. 3. In specific, it includes a topology-aware discriminator that stimulates student networks to mimic teachers and produces similar global affinity of patch-summary pairs. The student model serves as a generator and produces embeddings and logits that are similar to teacher output. During the training process, we adversarially train the student model against the discriminator in a "two-player minimax game" [21], which can ensure that the student network perfectly models the probability distribution of teacher knowledge at equilibrium via adversarial losses [44, 45].

In Section 4.2, we can get the joint graph embedding h_{AB} of graphs A and B by using the fusion module in the Feature-fusion framework, and we use mean-pooling to get the corresponding summary vectors s_{AB}. Then, to compute the loss of AKD, the discriminator has to predict a binary value "Real/Fake" that implies whether the pair $\{h_{AB}, s_{AB}\}$ or $\{h_{AB}, s_{AB}\}$ is real or fake. Fake embedding-summary pair implies that the joint graph embedding and summary representation are produced by the network of different roles, i.e., teacher and student. The mapping function of embedding discriminator D_e can be formulated as:

$$\begin{aligned} D_e\left(\mathbf{h}^{T/S}, \mathbf{s}_{\mathcal{G}}^{T}\right) &= \left\langle \mathbf{h}^{T/S}, \mathbf{W}\mathbf{s}_{\mathcal{G}}^{T} \right\rangle \in [0,1], \\ D_e\left(\mathbf{h}^{T/S}, \mathbf{s}_{\mathcal{G}}^{S}\right) &= \left\langle \mathbf{h}^{T/S}, \mathbf{W}\mathbf{s}_{\mathcal{G}}^{S} \right\rangle \in [0,1], \end{aligned} \tag{4}$$

where W is a learnable diagonal matrix. Thus, under the guidance of the discriminator D_e, the generator (student model) strives to yield indistinguishable graph embeddings. And the adversarial training process can be expressed as:

$$\min_{G^S} \max_{D_e} \mathcal{J}, \tag{5}$$

where $\mathcal{J}$ is:

$$\begin{aligned} \sum_{\mathcal{G}\in\mathcal{T}} \big(&\log \mathrm{P}\left(\text{Real} \mid D_e^g\left(\mathbf{h}_v^T, \mathbf{s}_{\mathcal{G}}^T\right)\right) + \log \mathrm{P}\left(\text{ Fake } \mid D_e^g\left(\mathbf{h}_v^S, \mathbf{s}_{\mathcal{G}}^T\right)\right) + \\ &\log \mathrm{P}\left(\text{Real} \mid D_e^g\left(\mathbf{h}_v^S, \mathbf{s}_{\mathcal{G}}^S\right)\right) + \log \mathrm{P}\left(\text{ Fake } \mid D_e^g\left(\mathbf{h}_v^T, \mathbf{s}_{\mathcal{G}}^S\right)\right)\big), \end{aligned} \tag{6}$$

where $\mathcal{T}$ is the set of input pair graphs. Therefore, by alternatively maximising and minimising the objective function, we will get an expressive student model when it converges [21].

Apart from the embedding discriminator D_e, the logit discriminator D_l is also used to transfer the knowledge hidden in the teacher logits to the student. Let $z_{\mathcal{G}}$ be the logit of teacher and student model and $y_{\mathcal{G}}$ be the label, the target of D_l is to maximize the following value:

$$\begin{aligned} \max_{D_\ell} \sum_{\mathcal{G}\in\mathcal{T}} \big(&\log \mathrm{P}\left(\text{Real} \mid D_\ell\left(\mathbf{z}_{\mathcal{G}}^T\right)\right) + \log \mathrm{P}\left(\text{ Fake } \mid D_\ell\left(\mathbf{z}_{\mathcal{G}}^S\right)\right) \\ &+ \log \mathrm{P}\left(y_{\mathcal{G}} \mid D_\ell\left(\mathbf{z}_{\mathcal{G}}^T\right)\right) + \log \mathrm{P}\left(y_G \mid D_\ell\left(\mathbf{z}_{\mathcal{G}}^S\right)\right)\big), \end{aligned} \tag{7}$$

Similar to the loss computation in [46, 47], we also introduce instance-level alignment between teacher and student logits besides the category-level alignment [21]. And we try to minimize the following value for the generator G^S (student model):

$$\begin{aligned} \min_{G^S} \sum_{\mathcal{G}\in\mathcal{T}} \big(&\log \mathrm{P}\left(\text{Real} \mid D_\ell\left(\mathbf{z}_{\mathcal{G}}^T\right)\right) + \log \mathrm{P}\left(\text{ Fake } \mid D_\ell\left(\mathbf{z}_{\mathcal{G}}^S\right)\right) \\ &- \left[\log \mathrm{P}\left(y_{\mathcal{G}} \mid D_\ell\left(\mathbf{z}_{\mathcal{G}}^T\right)\right) + \log \mathrm{P}\left(y_{\mathcal{G}} \mid D_\ell\left(\mathbf{z}_{\mathcal{G}}^S\right)\right)\right] \\ &+ \left\|\mathbf{z}_{\mathcal{G}}^S - \mathbf{z}_{\mathcal{G}}^T\right\|_1\big) \end{aligned} \tag{8}$$

4 Experiments

4.1 Datasets

We use four popular graph datasets that are commonly used for GSC tasks. Our experiments load the datasets by using the dataloader provided in Pytorch Geometric [48].

- **AIDS** (AIDS700nef) is a collection of antivirus screen chemical compounds which contains 42,687 chemical compound structures with Hydrogen atoms omitted [6]. We use the AIDS700nef selected by Bai et al. [6]. It contains 700 graphs, each of which has 10 or less than 10 nodes. Each node is labelled with one of 29 types.

- **LINUX** was first proposed by Wang et al. [49], which consists of 48,747 Program Dependence Graphs (PDG) generated from the Linux kernel. Each graph represents a function, where a node represents one statement and an edge represents the dependency between the two statements [6]. The nodes are unlabeled.
- **IMDB** dataset (named "IMDB-MULTI") [50] consists of 1500 ego-networks of movie actors/actresses, where there is an edge if the two people appear in the same movie [6]. The nodes are unlabeled.
- **ALKANE** dataset [51] is a purely structural dataset containing 120 chemical compound graphs. All the graphs are acyclic (i.e., trees) without node labels [6].

4.2 Experimental Settings

We set batch size=128, learning rate=0.001, and epochs=6000 for all four datasets. All experiments are run on University HPC (High Performance Computing) with NVIDIA A100-SXM4-80GB GPUs.

4.2.1 Baselines

The baseline algorithms we use follow the methods in [1] and the baseline results are obtained from their published papers. The introduction of baselines is given below.

- **Beam** [22] is a GED estimator based on Beam Search and is a variant of the A* algorithm [52] in sub-exponential time [1, 17].
- **Hungarian** [53] is the cubic-time algorithm based on the Hungarian Algorithm for bipartite graph matching [1].
- **VJ** [54] algorithm is a variant of the Hungarian method [1] and uses the Volgenant and Jonker algorithm for GED approximation [17].
- **SimGNN** [6] relies on a shared GNN encoder, a neural tensor network, and a pairwise node comparison module to compute the similarity between two graphs [1].
- **Extended-SimGNN** is an improved version of SimGNN that uses GIN as the backbone [1].
- **GraphSim** [19] extends SimGNN by using convolutional neural networks to capture multi-scale node-level interactions [17].
- **GMN** [12] is another GNN-based method and introduces a cross-graph attention layer that allows the nodes in the two graphs to interact with each other and predicts graph similarity using the representation vectors that fuse cross-graph information [17].
- **GENN-A*** [20] applies the GNN to accelerate hard GED solvers such as A*. Beam, Hungarian, VJ. GENN-A* are the GED solvers that require output edit path, which, however, are hard to generalize to other GSC metrics [1].

4.2.2 Evaluation Metric

- **Mean Squared Error (MSE)** is the most popular matrix that measures the average squared error between the predicted scores with the ground-truth similarities [1].
- **Spearman's Rank Correlation Coefficient (ρ) and Kendall's Rank Correlation Coefficient (τ)** evaluate the correlation of ranking-wise computed results and ground-truth results [1].
- **Precision at k (p@k)** (k = 10, 20) computes the interactions of the predicted and ground-truth top-k results divided by k [17].

  In general, the smaller the MSE, the better performance of models; for ρ, τ, and p@k, the larger the better [17].

4.3 Main Results

The evaluation results are shown in Table 1. Based on the results, it can be seen that EGSC-T (Teacher Model) and EGSC-S (Student Model) outperform baselines in most datasets. Although beaten by the GENN-A* method in some cases, EGSC-T and EGSC-S have advantages in extensibility and scalability because they do not require step-by-step output of the edit path [1].

Table 1: Evaluation results of baselines and EGSC-AKD-S (our method) over four datasets: AIDS700, LINUX, IMDB, ALKANE.

	AIDS700					LINUX				
	MSE ↓ ($x10^{-3}$)	$\rho\uparrow$	$\tau\uparrow$	p@10↑	p@20↑	MSE ↓ ($x10^{-3}$)	$\rho\uparrow$	$\tau\uparrow$	p@10↑	p@20↑
Beam	12.09	0.609	0.463	0.481	0.493	9.268	0.827	0.714	0.973	0.924
Hungarian	25.30	0.510	0.378	0.360	0.392	29.81	0.638	0.517	0.913	0.836
VJ	29.16	0.517	0.383	0.310	0.345	63.86	0.581	0.450	0.287	0.251
GEMN-A*	0.635	0.959	-	0.871	-	0.324	0.991	-	0.962	-
SimGNN	1.189	0.843	0.690	0.421	0.514	1.509	0.939	0.830	0.942	0.933
E-SimGNN	2.096	0.869	0.699	0.534	0.641	0.469	0.982	0.892	0.971	0.968
GMN	1.886	0.751	-	0.401	-	1.027	0.933	-	0.833	-
GraphSim	0.787	0.874	-	0.534	-	0.058	0.981	-	0.992	-
EGSC-T	1.601	0.901	0.739	0.658	0.729	0.163	0.988	0.908	0.994	0.998
EGSC-S	1.621	0.896	0.7338	0.6229	0.7193	0.2473	0.984	0.896	0.979	0.9838
EGSC-AKD-S (Our method)	**1.514**	**0.8977**	**0.7358**	**0.6471**	**0.7314**	**0.2123**	**0.9862**	**0.9024**	**0.9865**	**0.988**

	IMDB					ALKANE				
	MSE ↓ ($x10^{-3}$)	$\rho\uparrow$	$\tau\uparrow$	p@10↑	p@20↑	MSE ↓ ($x10^{-3}$)	$\rho\uparrow$	$\tau\uparrow$	p@10↑	p@20↑
SimGNN	1.264	0.878	0.770	0.759	0.777	2.446	0.859	0.686	0.87	0.782
E-SimGNN	1.148	0.864	0.75	0.806	0.807	1.622	0.886	0.722	0.982	0.955
GMN	4.422	0.725	-	0.604	-	-	-	-	-	-
GraphSim	0.743	0.926	-	0.828	-	-	-	-	-	-
EGSC-T	0.553	0.938	0.829	0.872	0.878	0.533	0.930	0.787	0.998	0.991
EGSC-S	0.5602	**0.938**	**0.8319**	0.857	0.866	0.9713	0.8899	0.7261	0.9967	0.9842
EGSC-AKD-S (Our method)	**0.5368**	0.9376	0.8298	**0.8623**	**0.8677**	**0.9044**	**0.9002**	**0.7406**	**0.9983**	**0.9875**

Bold Explanation: We mainly compare EGSC-AKD-S (our method) and EGSC-S (method in [1]), the bolded data represent the better results in EGSC-AKD-S and EGSC-S.

Table 2: Comparison of model parameters and computational complexity (FLOPs) between the teacher model and the student model. FLOPs are calculated from a data batch.

	EGSC-T (Teacher Model)	EGSC-AKD-S (Student Model)
Model Parameters	122038	20995
FLOPs (Floating Point Operations)	25749248	8765440

Since the improvements made by our proposed method (EGSC-AKD-S) are mainly based on **EGSC-S**, we compare **EGSC-AKD-S** and **EGSC-S**. Specifically, for MSE, our method outperforms EGSC-S on all four datasets, with MSE values less than **EGSC-S** by 0.107, 0.035, 0.0234, and 0.0669, respectively. For ρ and τ, although our method is slightly lower than **EGSC-S** on IMDB, it outperforms **EGSC-S** on the other three datasets. In terms of p@10, our method outperforms **EGSC-S** by 2.42%, 0.75%, 0.53% and 0.16% respectively. For p@20, the results of **EGSC-AKD-S** are 1.21%, 0.42%, 0.17%, 0.33% higher than **EGSC-S**. In summary, our proposed method outperforms **EGSC-S** in most scenarios.

4.4 Model Parameters and Computational Complexity

Considering the deployment of GSC models in industrial scenarios, we need to examine the impact of AKD on the model parameters and the amount of model computation. Table 3 demonstrates the comparison of model parameters and computational complexity (FLOPs) between the teacher model and the student model. Specifically, the number of model parameters for the student model is 17.2% of that in the teacher model and the value of FLOPs is 34.04% of that in the teacher model. Thus, through knowledge distillation, the student model is more lightweight compared to the teacher model.

Table 3: Inference time on AIDS700. **EGSC-S-R**: EGSC Student model with raw input graphs. **EGSC-AKD-S-R**: Our proposed method with raw input graphs. **EGSC-S-F**: EGSC Student model and the embeddings are stored offline.

Model	GENN-A*	SimGNN	E-SimGNN	E-SimGNN-F	EGSC-T	EGSC-S-R	EGSC-AKD-S-R	EGSC-S-F
Inference Time	290.1h	11.139s	9.672s	3.464s	7.176s	6.240s	6.325s	0.148s

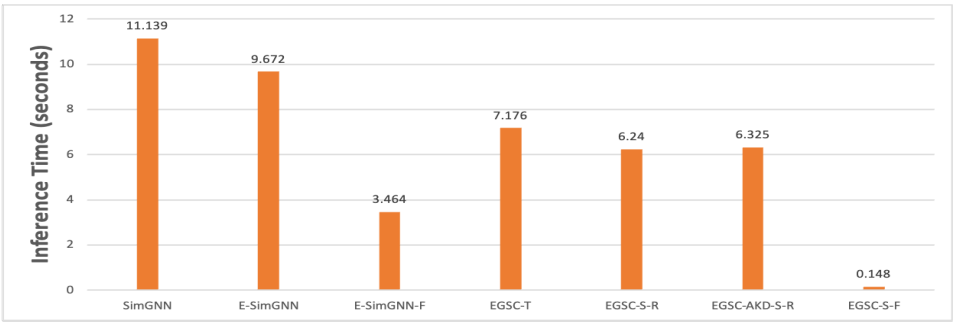

Figure 4: Inference time on AIDS700. The inference time (290.1h) for GENN-A* is not shown in the figure. **EGSC-AKD-S-R**: Our method with raw input graphs. **EGSC-S-F**: EGSC Student model and the embeddings are stored offline.

4.5 Inference Time

Section 4.3 discusses the accuracy improvement of **EGSC-AKD-S** compared to **EGSC-S**. In this section, we will discuss the effect of our proposed method on inference time.

Through experiments, we obtained the inference time of the model on the test dataset, as shown in Table 2 and Figure 4. Specifically, **EGSC-AKD-S** needs 6.325s to compute a total of 78400 GED values (78400 pairs of graphs) when processing the entire test set, and thus the average time taken by **EGSC-AKD-S** to compute each pair of the graph is 0.08 ms. Also, despite the addition of the adversarial training process, the inference time of **EGSC-AKD-S** does not change much compared with that of **EGSC-S** and still ensures a high inference efficiency.

5 Discussion

As described in section 4.3, our method performs better than EGSC-S in most situations. However, we need to note that the improvement in some metrics is not particularly large compared to EGSC-S, which gives us directions for future optimisation. First, we can analyse the impact of the loss calculation methods of the discriminator and generator on the accuracy of prediction and find a better loss function that can help us improve the performance. Second, although the current student model uses fewer parameters and computational effort (e.g. floating point operations) than the teacher model, we can still try to optimise the structure of the student model to make it lighter and still maintain good performance, which may require a better balance between accuracy and speed of inference. Besides, as we have only used knowledge distillation to compress the GSC model so far, in future work we can use more model compression techniques such as pruning, quantization, low-rank factorization, etc. to help achieve better performance and easier industrial deployment of the GSC model. Finally, the AKD approach we use is mainly to understand graph embeddings within the graph, but there are many more feature representations, so we can explore more representation discriminators in the future.

6 Conclusion

Graph similarity computing (GSC) has always been an important task in graph computing. This paper first introduces adversarial knowledge distillation (AKD) into the training process of the student model (EGSC-S) to better compress the current model (EGSC) and improve the prediction accuracy. The results on four popular datasets have shown that the proposed method can outperform the EGSC-S model in most scenarios. In addition, the proposed method has a similar inference time as EGSC-S, which suggests a good fit for deployment in industrial scenarios.

References

[1] Can Qin, Handong Zhao, Lichen Wang, Huan Wang, Yulun Zhang, and Yun Fu. Slow learning and fast inference: Efficient graph similarity computation via knowledge distillation. *Advances*

in Neural Information Processing Systems, 34:14110–14121, 2021.

[2] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[3] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[4] Yan-Bo Zhou, Ting Lei, and Tao Zhou. A robust ranking algorithm to spamming. *EPL (Europhysics Letters)*, 94(4):48002, 2011.

[5] Guixiang Ma, Nesreen K Ahmed, Theodore L Willke, and Philip S Yu. Deep graph similarity learning: A survey. *Data Mining and Knowledge Discovery*, 35(3):688–725, 2021.

[6] Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 384–392, 2019.

[7] Kirk Ogaard, Heather Roy, Sue Kase, Rakesh Nagi, Kedar Sambhoos, and Moises Sudit. Discovering patterns in social networks with graph matching algorithms. In *International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction*, pages 341–349. Springer, 2013.

[8] Karsten Steinhaeuser and Nitesh V Chawla. Community detection in a large real-world social network. In *Social computing, behavioral modeling, and prediction*, pages 168–175. Springer, 2008.

[9] Tamás Horváth, Thomas Gärtner, and Stefan Wrobel. Cyclic pattern kernels for predictive graph mining. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 158–167, 2004.

[10] Holger Fröhlich, Jörg K Wegner, Florian Sieker, and Andreas Zell. Kernel functions for attributed molecular graphs–a new similarity-based approach to adme prediction in classification and regression. *QSAR & Combinatorial Science*, 25(4):317–326, 2006.

[11] Karsten M Borgwardt, Hans-Peter Kriegel, SVN Vishwanathan, and Nicol N Schraudolph. Graph kernels for disease outcome prediction from protein-protein interaction networks. In *Biocomputing 2007*, pages 4–15. World Scientific, 2007.

[12] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *International conference on machine learning*, pages 3835–3845. PMLR, 2019.

[13] Sofia Ira Ktena, Sarah Parisot, Enzo Ferrante, Martin Rajchl, Matthew Lee, Ben Glocker, and Daniel Rueckert. Metric learning with spectral graph convolutions on brain connectivity networks. *NeuroImage*, 169:431–442, 2018.

[14] Horst Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern recognition letters*, 18(8):689–694, 1997.

[15] Yongjiang Liang and Peixiang Zhao. Similarity search in graph databases: A multi-layered indexing approach. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 783–794. IEEE, 2017.

[16] Juris Hartmanis. Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson). *Siam Review*, 24(1):90, 1982.

[17] Wei Zhuo and Guang Tan. Efficient graph similarity computation with alignment regularization. In *Advances in Neural Information Processing Systems*.

[18] Yupeng Hou, Wayne Xin Zhao, Yaliang Li, and Ji-Rong Wen. Privacy-preserved neural graph similarity learning. *arXiv preprint arXiv:2210.11730*, 2022.

[19] Yunsheng Bai, Hao Ding, Ken Gu, Yizhou Sun, and Wei Wang. Learning-based efficient graph similarity computation via multi-scale convolutional set matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3219–3226, 2020.

[20] Runzhong Wang, Tianqi Zhang, Tianshu Yu, Junchi Yan, and Xiaokang Yang. Combinatorial learning of graph edit distance via dynamic embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5241–5250, 2021.

[21] Huarui He, Jie Wang, Zhanqiu Zhang, and Feng Wu. Compressing deep graph neural networks via adversarial knowledge distillation. *arXiv preprint arXiv:2205.11678*, 2022.

[22] Michel Neuhaus, Kaspar Riesen, and Horst Bunke. Fast suboptimal algorithms for the computation of graph edit distance. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 163–172. Springer, 2006.

[23] Kaspar Riesen, Sandro Emmenegger, and Horst Bunke. A novel software toolkit for graph edit distance computation. In *International Workshop on Graph-Based Representations in Pattern Recognition*, pages 142–151. Springer, 2013.

[24] Horst Bunke and Kim Shearer. A graph distance metric based on the maximal common subgraph. *Pattern recognition letters*, 19(3-4):255–259, 1998.

[25] Mirtha-Lina Fernández and Gabriel Valiente. A graph distance metric combining maximum common subgraph and minimum common supergraph. *Pattern Recognition Letters*, 22(6-7): 753–758, 2001.

[26] Zhiping Zeng, Anthony KH Tung, Jianyong Wang, Jianhua Feng, and Lizhu Zhou. Comparing stars: On approximating graph edit distance. *Proceedings of the VLDB Endowment*, 2(1):25–36, 2009.

[27] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 869–877, 2018.

[28] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[29] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[30] Xiang Ling, Lingfei Wu, Saizhuo Wang, Tengfei Ma, Fangli Xu, Alex X Liu, Chunming Wu, and Shouling Ji. Multilevel graph matching networks for deep graph similarity learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[31] Zhen Zhang, Jiajun Bu, Martin Ester, Zhao Li, Chengwei Yao, Zhi Yu, and Can Wang. H2mn: Graph similarity learning with hierarchical hypergraph matching networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2274–2284, 2021.

[32] Yupeng Hou, Binbin Hu, Wayne Xin Zhao, Zhiqiang Zhang, Jun Zhou, and Ji-Rong Wen. Neural graph matching for pre-training graph neural networks. In *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*, pages 172–180. SIAM, 2022.

[33] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[34] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[35] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? *Advances in neural information processing systems*, 27, 2014.

[36] Zhenyang Wang, Zhidong Deng, and Shiyao Wang. Accelerating convolutional neural networks with dominant convolutional kernel and knowledge pre-regression. In *European Conference on Computer Vision*, pages 533–548. Springer, 2016.

[37] Yiding Yang, Jiayan Qiu, Mingli Song, Dacheng Tao, and Xinchao Wang. Distilling knowledge from graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7074–7083, 2020.

[38] Wentao Zhang, Xupeng Miao, Yingxia Shao, Jiawei Jiang, Lei Chen, Olivier Ruas, and Bin Cui. Reliable data distillation on graph convolutional network. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1399–1414, 2020.

[39] Cheng Yang, Jiawei Liu, and Chuan Shi. Extract the knowledge of graph neural networks and go beyond it: An effective knowledge distillation framework. In *Proceedings of the Web Conference 2021*, pages 1227–1237, 2021.

[40] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[41] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[42] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[43] Antoine Miech, Jean-Baptiste Alayrac, Ivan Laptev, Josef Sivic, and Andrew Zisserman. Thinking fast and slow: Efficient text-to-visual retrieval with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9826–9836, 2021.

[44] Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. Kdgan: Knowledge distillation with generative adversarial networks. *Advances in neural information processing systems*, 31, 2018.

[45] Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. Adversarial distillation for learning with privileged provisions. *IEEE transactions on pattern analysis and machine intelligence*, 43(3): 786–797, 2019.

[46] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[47] Zheng Xu, Yen-Chang Hsu, and Jiawei Huang. Training shallow and thin networks for acceleration via knowledge distillation with conditional adversarial networks. *arXiv preprint arXiv:1709.00513*, 2017.

[48] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[49] Xiaoli Wang, Xiaofeng Ding, Anthony KH Tung, Shanshan Ying, and Hai Jin. An efficient graph indexing method. In *2012 IEEE 28th International Conference on Data Engineering*, pages 210–221. IEEE, 2012.

[50] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1365–1374, 2015.

[51] Sébastien Bougleux, Luc Brun, Vincenzo Carletti, Pasquale Foggia, Benoit Gaüzere, and Mario Vento. A quadratic assignment formulation of the graph edit distance. *arXiv preprint arXiv:1512.07494*, 2015.

[52] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[53] Kaspar Riesen and Horst Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision computing*, 27(7):950–959, 2009.

[54] Stefan Fankhauser, Kaspar Riesen, and Horst Bunke. Speeding up graph edit distance computation through fast bipartite matching. In *International Workshop on Graph-Based Representations in Pattern Recognition*, pages 102–111. Springer, 2011.