Efficient Graph Similarity Computation with Alignment Regularization

Anonymous Author(s)
Affiliation
Address
email

Abstract

We consider the graph similarity computation (GSC) task based on graph edit distance (GED) estimation. State-of-the-art methods treat GSC as a learning-based prediction task using Graph Neural Networks (GNNs). To capture fine-grained interactions between pair-wise graphs, these methods mostly contain a node-level matching module in the end-to-end learning pipeline, which causes high computational costs in both the training and inference stages. We show that the expensive node-to-node matching module is not necessary for GSC, and high-quality learning can be attained with a simple yet powerful regularization technique, which we call the *Alignment Regularization* (AReg). In the training stage, the AReg term imposes a node-graph correspondence constraint on the GNN encoder. In the inference stage, the graph-level representations learned by the GNN encoder are directly used to compute the similarity score without using AReg again to speed up inference. We further propose a multi-scale GED discriminator to enhance the expressive ability of the learned representations. Extensive experiments on real-world datasets demonstrate the effectiveness, efficiency and transferability of our approach.

1 Introduction

Graph similarity computation (GSC) is a fundamental task in graph databases and plays a critical role in many real-world applications, including drug design [14, 28], program analysis [16], and social group identification [21, 27]. For example, one can search a drug database for a query chemical compound, identifying drugs with high similarity in structures or attributes and thus similar curative effects as desired [23]. To measure the similarity between pair-wise graphs, evaluation metrics such as Graph Edit Distance (GED) [4] has by far been the most well-studied due to its generality, and many other graph similarity measures have been proven to be special cases of GED [17]. However, computing exact GED is an NP-Hard problem in general [12].

With the provably expressive power in distinguishing graph structures [33, 19, 7], Graph Neural Networks (GNNs) have been adopted for GED approximation and achieve superior performance on accuracy. Most state-of-the-art GNN-based GSC models [1, 2, 8, 18, 16, 32] contain two sequential submodules in the end-to-end learnable pipeline (left of Fig. 1): (1) GNN encoder, which is shared across two graphs to embed nodes into representation vectors to capture intra-graph structure and feature information. (2) Matching model, which computes cross-graph node-level similarity, i.e., how a node in one graph relates to all the nodes in the other graph. The model outputs a summarized vector that fuses the node-level similarities between two graphs. Then, the similarity score is predicted based on the summarized vector via a regression head. The computational cost of such a sequential framework mainly comes from the matching model, which requires computational and memory cost quadratic in the number of nodes and sometimes involves additional parameters such as attention

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

weights [18, 16], leading to heavy time consumption in both the training and inference stages. Especially in the inference stage, we need to query every graph in the testing set from all graphs in the database. The recent approach EGSC [22] speeds up similarity learning by dropping the matching model from SimGNN [1]. However, since cross-graph interactions are ignored, EGSC cannot capture finer-grained similarity information, incurring suboptimal prediction performance. To overcome the intrinsic tension between predictive accuracy and speed, we propose a separated neural structure (right of Fig. 1) that detaches the matching model from the sequential pipeline, where the matching model only acts as a regularization term in the training stage to help the GNN encoder capture fine-grained similarity information, while in the inference (testing) stage, since the latent cross-graph interactions have been learned by the GNN encoder, the final similarity score can be directly computed based on the output representations of the GNN encoder without invoking the matching model again.

We show that explicitly learning the cross-graph node-to-node similarity is not necessary, as the correlation information contained in the features and graph topology, when properly exploited, is sufficient to reflect such cross-graph interactions (See Section 3). Specifically, the problem of GED computation is equivalent to finding an optimal permutation such that the adjacency matrices of the two graphs can be best aligned. Hence, by analyzing the the necessary conditions under the optimal permutation, we find that the best matching between two graphs can be inferred by minimizing the difference between the *intra-graph node-graph similarity* and *cross-graph node-graph similarity*. It motivates us to propose a task-agnostic matching model based on input data itself, leading to a regularization design, called the Alignment Regularization (AReg). AReg obviates the need of a matching model in the inferenece stage, thus making the model more efficient. AReg is also model-agnostic and can be applied to other GNN-based GSC models.

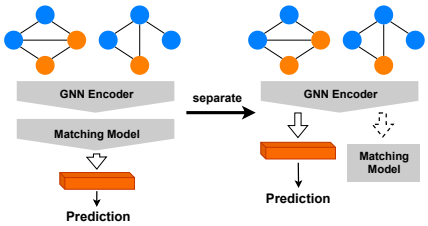


Figure 1: Illustration of separating the matching model from the end-to-end GSC framework to achieve a fast model (right side). In the fast model, the dotted arrow means the matching model does not participate in the similarity computation in the inference stage.

On the other hand, GNN-based GSC models usually use a single GED discriminator followed by a regression head to fuse graph representations from pair-wise input graphs and output a final similarity prediction score. We show that a single GED discriminator can not fully capture the dissimilarity between two graphs, while diverse discriminators may provide complementary information to reflect GED more accurately. Thus, in our paper we propose multi-scale GED discriminator to improve the discriminability of the learned representations. We call the overall framework ERIC: Efficient gRaph sImilarity Computation, and conduct extensive experiments to verify the efficacy of our design. Results on several real-world datasets demonstrate that ERIC achieves state-of-the-art performance by significantly outperforming the baselines.

2 Preliminary

Problem Formulation of GSC. Given a graph database $\mathcal{D}$ and a set of query graphs $\mathcal{Q}$, all the graphs in $\mathcal{D}$ and $\mathcal{Q}$ are unweighted and undirected. The problem of graph similarity computation is to produce a similarity score $\boldsymbol{y}$ between $\forall G_i \in \mathcal{Q}$ and $\forall G_j \in \mathcal{D}$, i.e., $\boldsymbol{y} = s(G_i, G_j)$ where $s : \mathcal{D} \times \mathcal{Q} \rightarrow \mathbb{R}^{(0,1]}$ is a similarity estimator. For a graph $G \in \mathcal{D} \cup \mathcal{Q}$, it is defined as $G = (V, E)$, where $V = \{v_k\}_{k=1}^{N}$ is a node set and E is an edge set. In our setting, all the accessible graphs are unweighted. V and E jointly formulate an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$. If nodes are accompanied by features (e.g., labels or attribute vectors), they are represented as $\boldsymbol{X} \in \mathbb{R}^{N \times d}$ with d dimensions.

Graph Edit Distance (GED). GED as a graph similarity measure has been popularly adopted on graph search queries, due to its capacity to capture the structural and feature differences between graphs. As shown in Fig. 2, GED is defined as the number of edit operations in the optimal path that transform G_i into G_j, where the possible edit operations we consider in this paper including edge

deletion/insertion, node deletion[1]/insertion, and node relabeling. To well fit the end-to-end regression task, instead of directly estimate the GED between two graphs, we convert the GED to the ground truth similarity score that the learning model aims to approximate. Following [1], the normalized GED is defined as $\mathrm{nGED}(G_i, G_j) = \frac{\mathrm{GED}(G_i,G_j)}{(|V_i|+|V_j|)/2}$, and the ground truth similarity score between G_i and G_j is defined as the normalized exponential of GED, resulting in a value ranging $(0, 1]$, i.e., $\mathbf{S}_{ij} = \exp(-\mathrm{nGED}(G_i, G_j))$.

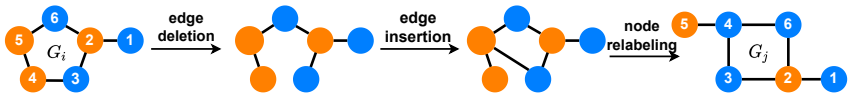


Figure 2: The optimal edit path with 3 edit operations to transform G_i to G_j. As a result, $\mathrm{GED}(G_i, G_j) = 3$.

3 Motivation: Analyzing GED in Embedding Space

Given two graphs $G_i = (\boldsymbol{A}_i, \boldsymbol{X}_i)$ and $G_j = (\boldsymbol{A}_j, \boldsymbol{X}_j)$ with the same number of nodes N (If the number of nodes is different, padding with 0 for $\boldsymbol{A}$ and $\boldsymbol{X}$ in the graph with less nodes), $\pi(\cdot)$ is a node index permutation that preserves the adjacency matrix. $\pi(\boldsymbol{A})$ denotes the adjacency matrix after node index permuting. We divide the computation of GED between G_i and G_j into two steps: (i) finding a permutation for G_j, such that

$$c = \min_{\pi} \sum_{k,l} \left| \left(\boldsymbol{A}_i - \pi(\boldsymbol{A}_j)\right)[k, l] \right|, \tag{1}$$

where c is the sum of the absolute values of all elements in matrix $\boldsymbol{A}_i - \pi(\boldsymbol{A}_j)$. We denote the optimal permutation satisfying Eq. (1) as $\pi^\star$. (ii) counting the number of cross-graph node pairs with the same indices yet different features, denoted as m. Then $\mathrm{GED}(G_i, G_j) = \frac{c}{2} + m$. Obviously, if two graphs are topologically isomorphic, there exists $\pi = \pi^\star$ such that $\boldsymbol{A}_i = \pi^\star(\boldsymbol{A}_j)$, i.e., $c = 0$. Hence the GED is only determined by distinct features. For another example, when the node permutation π assigns indices to G_j as shown in Fig. 2, the structure of G_i and G_j can be best aligned, i.e., $c = 4$. Then, only one pair of nodes with the same index across graphs have different features (e.g., node 4). Thus, $\mathrm{GED}(G_i, G_j) = 3$. The core of this two-step method is finding the optimal permutation to best align two graphs, then m is determined under such alignment. Traversing all possible permutations to find $\pi^\star$ is also NP-hard, so we make some heuristic rules from Eq. (1) to guide the model design. Under the optimal permutation $\pi^\star$, the row similarity between $\boldsymbol{A}_i$ and $\pi^\star(\boldsymbol{A}_j)$ is maximized, so given an injective function $f_\theta(\cdot) : \mathbb{R}^N \to \mathbb{R}^d$ parameterized by θ to guarantee that different node connectivity can be distinguished, a necessary condition is that the distance between $f_\theta(\boldsymbol{A}_i[k])$ and $f_\theta(\pi^\star(\boldsymbol{A}_j)[k])$ is minimized for all $k \in \{1, \cdots, N\}$. From global view, the matrix similarity between $\boldsymbol{A}_i$ and $\pi^\star(\boldsymbol{A}_j)$ is also maximized. Given an injective function $g_\phi(\cdot) : \mathbb{R}^{N\times N} \to \mathbb{R}^d$ parameterized by ϕ, another necessary condition under the optimal permutation is that the distance between $g_\phi(\boldsymbol{A}_i)$ and $g_\phi(\pi^\star(\boldsymbol{A}_j))$ is minimized. Thus, the optimal permutation $\pi^\star$ in Eq. (1) also satisfies the following function,

$$\pi^\star = \arg\min_{\pi} \mathrm{DIST}\left(f_\theta(\boldsymbol{A}_i[k]), f_\theta(\pi(\boldsymbol{A}_j)[k])\right) + \mathrm{DIST}\left(g_\phi(\boldsymbol{A}_i), g_\phi(\pi(\boldsymbol{A}_j))\right) \quad \forall k \in \{1, \cdots, N\}, \tag{2}$$

where $\mathrm{DIST}(\cdot, \cdot)$ is a distance metric. The two terms in Eq. (2) can reflect GED at node-level and global-level respectively. Further, we regard $\{f_\theta(\boldsymbol{A}_i[k])\}_{k=1}^N \cup \{f_\theta(\pi^\star(\boldsymbol{A}_j)[k])\}_{k=1}^N$ as $2N$ anchors and assume $N \gg d$. When the second term of Eq. (2) reaches minimum, it indicates that $g_\phi(\boldsymbol{A}_i)$ and $g_\phi(\pi^\star(\boldsymbol{A}_j))$ have similar distances to all anchors, i.e., the following γ_i and γ_j take the minimum value when $\pi = \pi^\star$,

$$\gamma_i = \sum_{k}^{N} \left\| \mathrm{DIST}\left(f_\theta(\boldsymbol{A}_i[k]), g_\phi(\boldsymbol{A}_i)\right) - \mathrm{DIST}\left(f_\theta(\boldsymbol{A}_i[k]), g_\phi(\pi(\boldsymbol{A}_j))\right) \right\|_2 \tag{3}$$

[1]For node deletion, all edges connected to the deleted node are also deleted. Although editing of multiple edges is involved, it is a single-time effort. Thus, treat node deletion as as a one-time graph editing operation.

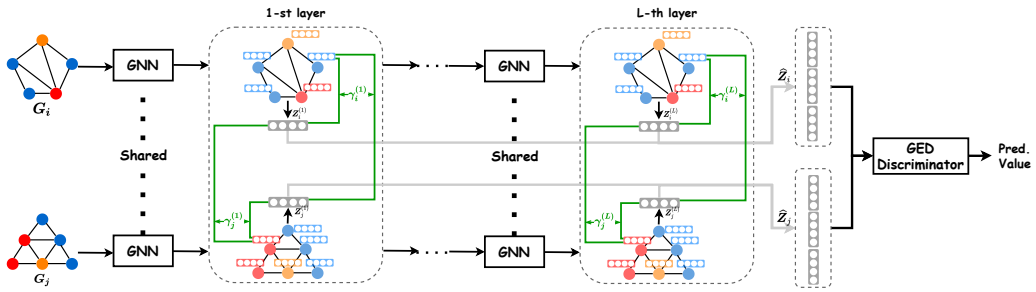


Figure 3: Overview of ERIC. The GNN encoder is shared across two graphs. The green lines denote AReg. The summarized graph representations $\widehat{\boldsymbol{Z}}_i$ and $\widehat{\boldsymbol{Z}}_j$ are combination of graph representations learned in each layer, and they are fed into the GED discriminator followed by a regression function to obtain the prediction value. In the inference stage, the AReg submodule is removed.

$$\gamma_j = \sum_{k}^{N} \left\| \mathrm{DIST}\left(f_\theta(\pi(\boldsymbol{A}_j)[k]), g_\phi(\boldsymbol{A}_i)\right) - \mathrm{DIST}\left(f_\theta(\pi(\boldsymbol{A}_j)[k]), g_\phi(\pi(\boldsymbol{A}_j))\right) \right\|_2 . \tag{4}$$

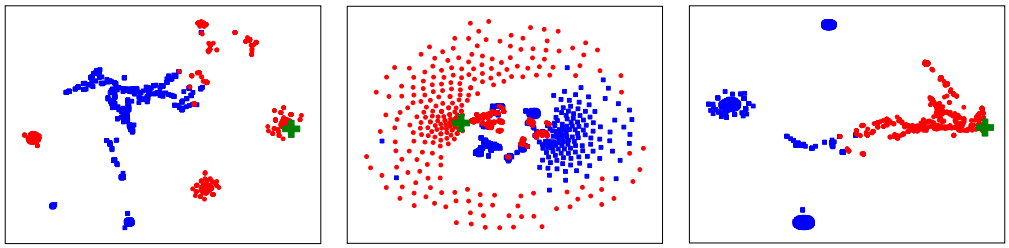
On the other hand, the first term of Eq. (2) is a finer grained alignment between two graphs, and $\pi^\star$ causes the first term of Eq. (2) to reach a minimum. Since the only difference between γ_i and γ_j is that γ_j replaces $f_\theta(\boldsymbol{A}_i[k])$ in γ_i with $f_\theta(\pi(\boldsymbol{A}_j)[k])$, therefore the distance between $f_\theta(\boldsymbol{A}_i[k])$ and $f_\theta(\pi(\boldsymbol{A}_j)[k])$ can be reflected by the difference between γ_i and γ_j, i.e., $\|\gamma_i - \gamma_j\|_2$. Hence, under the optimal permutation $\pi = \pi^\star$, G_i and G_j are best aligned such that Eq. (2) is established, which is equivalent to γ_i, γ_j, and $\|\gamma_i - \gamma_j\|_2$ all taking the minimum values. These terms reflect similarities between pair-wise graphs at both node- and graph- levels, which can be inferred from the graph structure itself without consulting the ground-truth GEDs. It motivates us to separate the matching model from the end-to-end pipeline. Assuming g_ϕ is a permutation-invariant function, then all permutation operator in Eq. (3) and Eq. (4) can be directly removed. Moreover, instead of computing cross-graph node-to-node similarity to compute a 'soft' alignment as in most related works, γ_i and γ_j formally only compute node-to-graph similarity, which is more efficient.

4 Proposed Model: ERIC

In this section, we detail the proposed ERIC framework, which consists of two submodules: 1) Alignment Regularization (AReg) and 2) Multi-Scale GED Discriminator. The framework of ERIC is shown in Fig. 3. AReg aims to train the shared GNN encoder such that GNN is capable of capturing underlying alignment information between pair-wise graphs. The multi-scale GED discriminator trains the same GNN encoder such that the learned representations of two graphs can directly reflect the exact GED value. Such paradigm follows the intuition of GED definition that when two graphs are best aligned, the difference between them is the GED.

4.1 Alignment Regularization

Learning the optimal alignment between graphs is crucial for GED estimation, however, most existing GSC models rely on intricate cross-graph node-to-node similarity computation to learn a 'soft' alignment. Furthermore, learning node-to-node similarity inevitably yields a dense similarity matrix, which could introduce more noise. To address this issue, we introduce Alignment Regularization (AReg) as a part of our framework. AReg is a model- and task-agnostic regularization, it can trivially be combined with any existing embedding-based GSC model in a plug-and-play manner. The design of AReg is motivated by our analysis of GED in Section 3. Recall that GED is the difference between two graphs when they are best aligned under the optimal permutation $\pi^\star$, whose necessary conditions are γ_i, γ_j and $\|\gamma_i - \gamma_j\|_2$ taking the minimum values. Thus, we expect that the learning-based model can couple with the nature of GED as much as possible, i.e., the GNN encoder can preserve the best alignment, in which case the difference between the learned graph representations can reflect the GED. Therefore, we design the model based on the necessary conditions under $\pi^\star$. Specifically, based on Eq. (3) and Eq. (4), we instantiate $f_\theta(\cdot)$ as the L-layer Graph Isomorphism Network (GIN) [33],



(a) NTN (b) ℓ_2 distance (c) NTN + ℓ_2 distance

Figure 4: t-SNE [29] visiualization of *IMDB* dataset. Each point is a graph encoded by the GNN encoder of ERIC. The green cross means a randomly sampled query graph; red points mean the top 50% of graph datasets that are most similar to the query graph based on ground-truth GEDs; blue points means the remaining 50% graphs in the dataset. (a): Using NTN as the discriminator, many similar graphs do not cluster together around the query graph even if each cluster is tight. (b): Using ℓ_2 distance as the discriminator, different clusters are separated clearly but the query graph is closed to the cluster boundary and each cluster is dispersive. (c): By adaptively combining NTN and ℓ_2 distance as the discriminator, the query is far from the boundary and similar graphs are located around the query.

then at the ℓ-th layer $f_\theta^{(\ell)}(\boldsymbol{A}_i[k])$ can be represented as:

$$\boldsymbol{H}_i^{(\ell)}[k] = f_\theta^{(\ell)}(\boldsymbol{A}_i[k]) = \mathrm{MLP}_\theta^{(\ell)}\left((1+\xi^{(\ell)})\boldsymbol{H}_i^{(\ell-1)}[k] + \boldsymbol{A}_i[k]\boldsymbol{H}_i^{(\ell-1)}\right), \tag{5}$$

where $\xi^{(\ell)}$ is a learnable parameter, $\boldsymbol{H}_i^{(\ell)} \in \mathbb{R}^{N\times d^{(\ell)}}$ is the feature matrix of G_i at the ℓ-th layer where $d^{(\ell)}$ denotes the feature dimension, $\boldsymbol{H}_i^{(0)} = \boldsymbol{X}_i$. On the other hand, AReg implements the readout function g_ϕ with a one-layer *permutation-invariant* DeepSets [34] to guarantee injectiveness, taking the form:

$$\boldsymbol{Z}_i^{(\ell)} = g_\phi^{(\ell)}(\boldsymbol{A}_i) = \mathrm{MLP}_\phi^{(\ell)}\left(\sum_k^N f_\theta^{(\ell)}(\boldsymbol{A}_i[k])\right), \tag{6}$$

where $\mathrm{MLP}_\phi^{(\ell)}$ and $\mathrm{MLP}_\theta^{(\ell)}$ have the same output dimension $d_{\mathrm{MLP}}^{(\ell)}$. Since $g_\phi(\boldsymbol{A}_j) = g_\phi(\pi^\star(\boldsymbol{A}_j))$, and γ_i and γ_j are sum over all N nodes, hence they are permutation-invariant and we can remove all operation π in Eq. (3) and Eq. (4). $\mathrm{DIST}(\cdot,\cdot)$ can be defined as any distance metrics such as cosine similarity. Let $\gamma_i^{(\ell)}$ and $\gamma_j^{(\ell)}$ be the value of Eq. (3) and Eq. (4) at the ℓ-th layer, by considering multi-scale cross-graph interactions, AReg is represented as $\mathcal{L}_{\mathrm{AReg}}$ where:

$$\mathcal{L}_{\mathrm{AReg}} = \frac{1}{L}\sum_\ell^L \left(\gamma_i^{(\ell)} + \gamma_j^{(\ell)} + \left\|\gamma_i^{(\ell)} - \gamma_j^{(\ell)}\right\|_2\right). \tag{7}$$

Note that AReg as a regularization term helps the GNN encoder capture the underlying cross-graph interactions in the training stage, and is no longer needed in the inference stage. Since γ_i and γ_j induced from Eq. (2) integrate graph-level alignment and finer grained node-level alignment, $\mathcal{L}_{\mathrm{AReg}}$ therefore preserves underlying cross-graph interactions without computing complicated node-to-node similarity, also making the training stage more efficient.

4.2 Multi-Scale GED Discriminator

Now we have L graph-level representations for G_i and G_j respectively, denoted as $\{\boldsymbol{Z}_i^{(1)}, \cdots, \boldsymbol{Z}_i^{(L)}\}$ and $\{\boldsymbol{Z}_j^{(1)}, \cdots, \boldsymbol{Z}_j^{(L)}\}$. We concatenate the layer-wise graph representations for G as: $\widehat{\boldsymbol{Z}} = \bigoplus\left\{\boldsymbol{Z}^{(\ell)}\right\}_{\ell=1}^L$, where $\bigoplus$ denotes the concatenation operator along the last dimension to combine the graph representation in each layer, i.e., $\widehat{\boldsymbol{Z}}_i, \widehat{\boldsymbol{Z}}_j \in \mathbb{R}^{d_{\mathrm{ms}}}$, and $d_{\mathrm{ms}} = \sum_\ell d_{\mathrm{MLP}}^{(\ell)}$. Then they are fed into a GED discriminator that generates score vectors as GED similarity embedding for the graph pair. Neural Tensor Network (NTN) [26] has demonstrated strong power to model the relation between the graph-level embeddings of two graphs [1, 22, 30] thanks to its capacity of exploring the element-wise

dependence among the features. However, directly using NTN as the discriminator may be expensive when the dimension of $\widehat{\boldsymbol{Z}}_i$ and $\widehat{\boldsymbol{Z}}_j$ is high. To solve this issue, we decompose the weight matrix $\mathbf{W}^t \in \mathbb{R}^{d_{\text{ms}} \times d_{\text{ms}}}$ into two matrices $\mathbf{W}_1^t \in \mathbb{R}^{d_{\text{ms}} \times d'}$ and $\mathbf{W}_2^t \in \mathbb{R}^{d' \times d_{\text{ms}}}$ where $d' \ll d_{\text{ms}}$ to reduce the number of parameters. Then NTN with decomposed weight matrices is used to measure the similarity between $\widehat{\boldsymbol{Z}}_i$ and $\widehat{\boldsymbol{Z}}_j$:

$$s_{\text{NTN}}(G_i, G_j) = \delta_{\text{NTN}}\left(\left[(\widehat{\boldsymbol{Z}}_i \mathbf{W}_1^t)(\mathbf{W}_2^t \widehat{\boldsymbol{Z}}_j^\top) : t \in \{1, \cdots, T\}\right]^\top + \mathbf{W}_3 \bigoplus \left\{\widehat{\boldsymbol{Z}}_i, \widehat{\boldsymbol{Z}}_j\right\} + \mathbf{b}\right) \tag{8}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_{\text{ms}} \times d' \times T}$, $\mathbf{W}_2 \in \mathbb{R}^{d' \times d_{\text{ms}} \times T}$, $\mathbf{W}_3 \in \mathbb{R}^{T \times 2d_{\text{ms}}}$, and $\mathbf{b} \in \mathbb{R}^T$ are learnable; T is a hyper-parameter controlling the output dimension which is assigned as 16 for all datasets. $[\cdot]$ in Eq. (8) means computing $(\widehat{\boldsymbol{Z}}_i \mathbf{W}_1^t)(\mathbf{W}_2^t \widehat{\boldsymbol{Z}}_j^\top)$ for all $t \in \{1, \cdots, T\}$ and stacking them as a T-dimensional tensor. $\delta_{\text{NTN}} : \mathbb{R}^T \to \mathbb{R}^{(0,1]}$ is a fully-connected neural network with sigmoid activation as a regression function to project the similarity score vector to the final predicted similarity value. NTN is an expressive and general similarity discriminator because it can approximate many similarity measures. The first term of Eq. (8) can be regarded as a *multi-head* weighted cosine similarity function. It can also approximate kernel similarity between $\widehat{\boldsymbol{Z}}_i$ and $\widehat{\boldsymbol{Z}}_j$ according to the universal approximation theorem [13]. The second term can capture the residual knowledge. However, it is difficult for NTN to approximate high-order Minkowski distance between $\widehat{\boldsymbol{Z}}_i$ and $\widehat{\boldsymbol{Z}}_j$, while diverse similarity discriminators may provide complementary information to reflect GED more accurately as shown in Fig. 4. Hence, we consider an additional similarity discriminator based on exponential p-order Minkowski distance:

$$s_p(G_i, G_j) = \delta_p\left(\exp\left(-\left\|\widehat{\boldsymbol{Z}}_i - \widehat{\boldsymbol{Z}}_j\right\|_p\right)\right) \tag{9}$$

where $\delta_p : \mathbb{R}^{d_{\text{ms}}} \to \mathbb{R}^{(0,1]}$ is also a fully-connected neural network with sigmoid activation. For simplicity, we uniformly set $p = 2$ (i.e., ℓ_2 distance) for all datasets, and analyze the sensitivity of the hyper-parameter p in Appendix B.3. After two similarity scores $s_{\text{NTN}}(G_i, G_j)$ and $s_p(G_i, G_j)$ are obtained, the final estimated similarity score is given by:

$$s(G_i, G_j) = \alpha s_{\text{NTN}}(G_i, G_j) + \beta s_p(G_i, G_j) \tag{10}$$

where α and β are trainable scalars denoting the weights of two similarity discriminators respectively. Given a graph database $\mathcal{D}$, the GED discriminator is trained on a set of n training pairs $(G_i, G_j) \in \mathcal{D} \times \mathcal{D}$. The predicted similarity is compared against the ground-truth similarity $\mathbf{S}_{ij}$ based on GEDs with Mean Squared Error (MSE) loss function as:

$$\mathcal{L}_{\text{GED}} = \frac{1}{n} \sum_{(G_i, G_j) \in \mathcal{D} \times \mathcal{D}} (s(G_i, G_j), \mathbf{S}_{ij}) \tag{11}$$

Model training: Combining AReg and GED discriminator, the training stage aims to minimize the following overall objective function $\mathcal{L} = \mathcal{L}_{\text{GED}} + \lambda \mathcal{L}_{\text{AReg}}$, where λ is a trainable parameter controlling the strength of the regularization term.

Model inference: Given a set of query graphs $\mathcal{Q}$ and a graph database $\mathcal{D}$, in the testing stage, all pairs of graphs $(G_i, G_j) \in \mathcal{Q} \times \mathcal{D}$ are fed into ERIC, and directly computing a similarity score for each pair based on Eq. (10) without any node-level interactions.

Complexity: The time complexity of GIN in AReg is $O(m)$ [31] where m is the number of edges. The cross-graph node-graph interactions have complexity $O(\max(N_i, N_j))$. The time complexity of GED discriminator is $O(d_{\text{ms}} d' T)$. Thus the complexity of ERIC is $O(m + \max(N_i, N_j) + d_{\text{ms}} d' T)$ in the training stage, while in the inference stage the complexity is $O(m + d_{\text{ms}} d' T)$.

5 Experiments

In this section, we empirically evaluate our proposed ERIC[2] on graph similarity computation task.

[2]Code is available at https://anonymous.4open.science/r/ERIC-8765

Table 1: Evaluation on benchmarks. **Bold** : best.

	AIDS700					LINUX					IMDB					PubChem2000				
	mse ($\times 10^{-3}$)↓	ρ↑	τ↑	$p@10$↑	$p@20$↑	mse ($\times 10^{-3}$)↓	ρ↑	τ↑	$p@10$↑	$p@20$↑	mse ($\times 10^{-3}$)↓	ρ↑	τ↑	$p@10$↑	$p@20$↑	mse ($\times 10^{-3}$)↓	ρ↑	τ↑	$p@10$↑	$p@20$↑
Beam	12.090	0.609	0.463	0.481	0.493	9.268	0.827	0.714	0.973	0.924	-	-	-	-	-	-	-	-	-	-
VJ	29.157	0.517	0.383	0.310	0.345	63.86	0.581	0.450	0.287	0.251	-	-	-	-	-	-	-	-	-	-
Hungarian	25.296	0.510	0.378	0.360	0.392	29.81	0.638	0.517	0.913	0.836	-	-	-	-	-	-	-	-	-	-
SimGNN	1.573	0.835	0.678	0.417	0.489	2.479	0.912	0.791	0.635	0.650	1.437	0.871	0.752	0.710	0.769	3.381	0.296	0.397	0.274	0.298
GraphSim	2.014	0.839	0.662	0.401	0.499	0.762	0.953	0.882	0.956	0.951	1.924	0.825	0.821	0.813	0.825	2.930	0.557	0.497	0.568	0.617
GMN	4.610	0.672	0.497	0.200	0.263	2.571	0.906	0.763	0.888	0.856	4.320	0.665	0.601	0.588	0.593	4.394	0.207	0.298	0.226	0.203
EGSC	1.676	0.888	0.723	0.604	0.708	0.214	0.984	0.897	0.987	0.989	0.573	**0.939**	**0.829**	0.872	0.883	1.852	0.795	0.692	0.657	0.642
CoSimGNN	2.573	0.668	0.674	0.529	0.507	3.784	0.902	0.841	0.792	0.751	1.964	0.866	0.707	0.662	0.678	3.389	0.551	0.573	0.482	0.500
MGMN	2.297	0.904	0.736	0.456	0.534	2.040	0.965	0.858	0.956	0.920	0.496	0.881	0.803	0.874	0.861	1.979	0.826	0.663	0.629	0.656
ERIC	**1.374**	**0.906**	**0.741**	**0.685**	**0.758**	**0.107**	**0.988**	**0.908**	**0.994**	**0.999**	**0.385**	0.890	0.791	**0.882**	**0.891**	**0.837**	**0.896**	**0.725**	**0.664**	**0.680**

5.1 Datasets

We conduct experiments on four widely used GSC datasets including *AIDS700*, *LINUX*, *IMDB* [1], and *PubChem* [6]. For PubChem, we extract 2000 graphs as datasets. Following the same splits as [1, 2], i.e., 60%, 20%, and 20% of all graphs as training set, validation set, and query set, respectively. Training set together with validation set is called *database*. More details of the datasets are presented in Appendix A.1.

5.2 Experimental Settings

Baselines. We implement two groups of baselines for comparison: (1) Combinatorial Search-based Methods: Beam [20], Hungarian [15], and VJ [9]. (2) GNN-based Methods: SimGNN [1], GMN [16], GraphSim [2], CoSimGNN [32], MGMN [18], and EGSC [22]. We re-implemented all baselines with the same hyper-parameters as the original literature provides, or carefully tuned the parameters to get the optimal results when they are not provided. We give more implementation details of ERIC and baselines in Appendix A.2.

Evaluation Metric. Similarly to [1, 22], the following five metrics are adopted to evaluate results for fair comparisons: **Mean Squared Error** (MSE) which measures the average squared differences between the predicted and the ground-truth similarity. **Spearman's Rank Correlation Coefficient** (ρ) and **Kendall's Rank Correlation Coefficient** (τ) which evaluates the ranking correlations between the predicted and the true ranking results. **Precision@k** where $k = 10, 20$, which computes the interactions of the predicted and ground-truth top-k results divided by k. The smaller the MSE, the better performance of models. But for ρ, τ, and $p@k$, the larger the better.

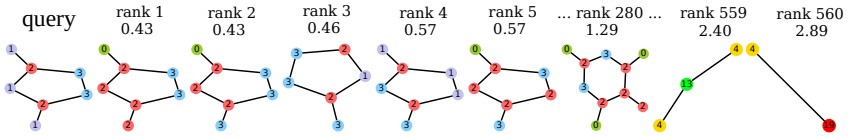
5.3 Main Results

The results of ERIC and the baselines on our benchmarks are reported in Table 1. For datasets with relatively small graphs whose ground-truth GEDs is exactly computed by $\mathrm{A}^\star$ algorithm, ERIC consistently achieve state-of-the-arts performance across all evaluation metrics as shown in Table 1. For datasets with large graphs whose ground-truth GEDs are taken from approximation, ERIC still achieves the best results on PubChem2000, and comparable results on IMDB. The suboptimal performance of the methods based on node-to-node similarity (SimGNN, GraphSim, GMN, MGMN and CoSimGNN) demonstrate that overly dense and fine-grained similarity computation may not always bring a beneficial boost. EGSC also uses GIN as then backbone and totally ignores cross-graph node-level interactions, and it achieves the best results on metrics ρ and τ on the IMDB dataset. The reason is that EGSC adopts intra-graph attention pooling and NTN in all layers to enhance the expression ability of representation vectors. However, the attention mechanism and full NTN need additional parameters which increase the burden of learning. Although EGSC proposed a student model based on knowledge distillation to speed up the inference process, it would sacrifice the prediction accuracy in most cases. The performance of ERIC over baselines illustrates the importance of the fine-grained interaction and the proper design of such interaction.

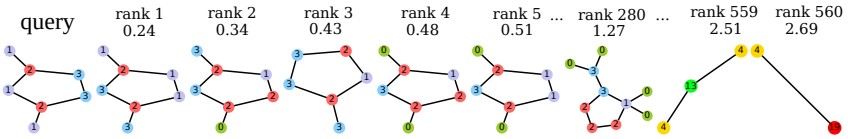
Our ERIC framework contains two key components: (i) AReg, (ii) Multi-Scale GED Discriminator. To glean a deeper insight into how different components help ERIC to achieve highly competitive results, we conduct ablation experiments in Appendix B.1 by removing individual component separately. Further, since AReg is a model-agnostic regularization, we are interested in the transferability of AReg, so we evaluate the performance of applying AReg on other GSC models in Appendix B.2.

Table 2: Inference time (sec) comparison.

Dataset	SimGNN	GraphSim	GMN	MGMN	EGSC	ERIC
AIDS700	10.773	14.043	23.975	11.337	8.763	**5.662**
LINUX	19.347	31.238	82.489	22.574	21.573	**15.969**
IMDB	225.682	379.480	1253.551	357.933	133.437	**48.750**
PubChem2000	369.227	431.520	2539.693	832.774	331.568	**146.917**



(a) The exact similarity ranking based on $A^\star$



(b) The predicted similarity ranking based on ERIC

Figure 5: Visualization of graph search case study on AIDS700 dataset. Nodes with different labels are assigned by different colors. (a) and (b) are similarity ranking based on the normalized GEDs computed by $A^\star$ (exact) and ERIC (estimation) respectively.

5.4 Inference Time

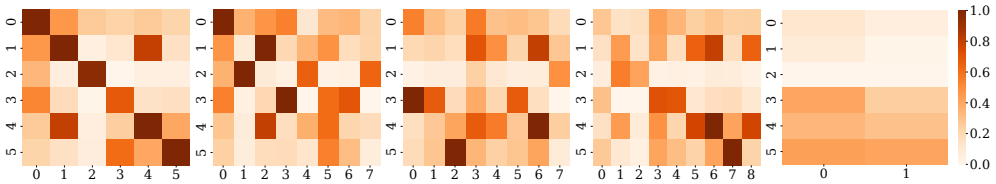
ERIC is a separated neural structure in which the cross-graph alignment only acts as a regularization term in the training stage but is ignored in the inference stage. To evaluate the efficiency, we compare the performance of ERIC with baselines in terms of inference time in Table 2. In Table 3 of Appendix A.1 we list the number of graph pairs in the inference stage (#Testing pairs), and all experiments are implemented with a single machine with 1 NVIDIA Quadro RTX 8000 GPU. As can be observed, node-to-node similarity computation is more time-consuming on all datasets. Our proposed node-graph similarity computation does not incur substantial additional running time in practice. To summarize, ERIC is faster than all baseline models, while still achieving significantly higher accuracy on all datasets.

5.5 Visualization of Graph Search Case

The goal of graph search task is to find k graphs from the dataset that are most similar to the given query graph, which is a routine task in drug discovery [24]. In Fig. 5 we show a case based on the AIDS700 dataset, where each graph represents a functional group. Comparing the exact similarity ranking computed by $A^\star$ and the estimated one computed by ERIC based on the same query graph, we can find that the ranking of ERIC has a high consistency with the exact ranking, which proves that ERIC is able to extract graphs that contain the similar substructure from around 700 graphs with varying size and structure and the top-ranked graph has a high degree of isomorphism with the query. It also demonstrates that ERIC can capture structural patterns shared across graphs. More results of visualization are included in Appendix C.

5.6 Analysis of Node-to-Node Similarity

In Fig. 6, we show the node-to-node similarity between a query graph and the graphs at different ranking positions in terms of normalized GEDs. It can be found that for graph pairs with small GED, the node representations learned by ERIC show a clear correspondence between nodes as shown in Fig. 6 (a). As the GED increases, the correspondence between nodes across the graph gradually weakens, i.e., cross-graph node-to-node similarity reduces. Thus, the similarity matrices in Fig. 6 can guide us to find a better alignment between pair-wise graphs.



(a) 0.167 (1st) (b) 0.571 (10th) (c) 0.857 (50th) (d) 1.067 (100th) (e) 2.750 (last)

Figure 6: Heatmaps of cross-graph node-to-node cosine similarity based on the node representations learned by the GNN encoder of ERIC. Y-axis means the node in a randomly sampled query graph. (a)~(e) are ranked by the exact normalized GEDs between the query graph and graphs of particular ranks in the database. For example, 0.571 (10th) means the nGED of the graph that ranks 10th in similarity to the query graph is 0.571. The color depth in the heatmap represents the similarity of the node pair, the deeper the color, the higher the similarity.

6 Related Work

The goal of graph similarity computation (GSC) is to quantify the similarity between graphs under a specific similarity measure. Various similarity measures have been well studied in prior works, such as graph edit distance (GED) [4, 25, 20] and maximum common subgraph (MCS) [5, 10], where GED as the most popular one that many other similarity measures can be proven to be its special cases [17]. However, computing the exact GED between two graphs is known to be NP-Complete and still challenging in practice when the number of nodes is more than 16 [3] using exact GED solvers such as $\mathrm{A}^\star$ algorithm [25]. Thus, approximation algorithms have been proposed for GED-based GSC. These approximate methods can be broadly divided into two groups: (1) *Combinatorial search-based methods*, which aim to exploit combinatorial structures or GED lower-bounds to approximate GED. Beam [20] is a GED estimator based on Beam Search; Hungarian [15] is proposed based on the famous Hungarian algorithm for weighted graph matching; Hausdorff approximation [11] provides a lower bound to the GED approximation; VJ [9] using the Volgenant and Jonker algorithm for GED approximation. However, these methods are highly heuristic-driven and run with either sub-exponential time or cubic time complexity, limiting the scalability as the growth of graph size. Besides, these methods totally ignore the node feature information, such that the underlying semantic similarity can not be captured. (2) *Learning-based methods*, which is a kind of data-driven solution aiming to learn graph similarity from the data itself with relatively high accuracy and far lower time complexity compared with search-based methods. GMN [16] introduces a cross-graph attention layer to allow the nodes in the two graphs to interact with each other and predicts graph similarity using the representation vectors that fuses cross-graph information. SimGNN [1] relies on a shared GNN encoder, a neural tensor network, and a pairwise node comparison module to compute similarity between two graphs. GraphSim [2] extends SimGNN by using convolutional neural networks to capture the multi-scale node-level interactions. EGSC [22] simplifies SimGNN by ignoring node-level interactions and uses knowledge distillation to accelerate the inference stage. MGMN [18] employs a node-graph matching network to capture cross-level features between nodes of a graph and the other whole graph, where the cross-graph aggregation weights are computed by node-to-node attention coefficients. Our proposed ERIC is also a learning-based method. Unlike the above approach either discards node-level interactions entirely, or computes all node interactions, our method proposes a novel node-graph interaction paradigm based on the insight into GED. It allows avoiding dense similarity computations while preserving fine-grained interactions.

7 Conclusion

We propose ERIC, a simple yet powerful GNN-based framework for the graph similarity computation task. Specifically, we fist give a deep insight into the graph edit distance, and propose Alignment Regularization (AReg) which is a separated structure independent of the end-to-end learning pipeline. AReg can free the model from complicated node-to-node similarity computation. Further, we propose a multi-scale GED discriminator to improve the discriminative of the learned representations. We showed the effectiveness of our model through a comprehensive set of experiments and analyses.

References

[1] Bai, Y., Ding, H., Bian, S., Chen, T., Sun, Y., and Wang, W. Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 384–392, 2019.

[2] Bai, Y., Ding, H., Gu, K., Sun, Y., and Wang, W. Learning-based efficient graph similarity computation via multi-scale convolutional set matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3219–3226, 2020.

[3] Blumenthal, D. B. and Gamper, J. On the exact computation of the graph edit distance. *Pattern Recognition Letters*, 134:46–57, 2020.

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

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

[6] Chang, L., Feng, X., Lin, X., Qin, L., Zhang, W., and Ouyang, D. Speeding up ged verification for graph similarity search. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pp. 793–804. IEEE, 2020.

[7] Chen, Z., Villar, S., Chen, L., and Bruna, J. On the equivalence between graph isomorphism testing and function approximation with gnns. *Advances in neural information processing systems*, 32, 2019.

[8] Doan, K. D., Manchanda, S., Mahapatra, S., and Reddy, C. K. Interpretable graph similarity computation via differentiable optimal alignment of node embeddings. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 665–674, 2021.

[9] Fankhauser, S., Riesen, K., and Bunke, H. Speeding up graph edit distance computation through fast bipartite matching. In *International Workshop on Graph-Based Representations in Pattern Recognition*, pp. 102–111. Springer, 2011.

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

[11] Fischer, A., Plamondon, R., Savaria, Y., Riesen, K., and Bunke, H. A hausdorff heuristic for efficient computation of graph edit distance. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pp. 83–92. Springer, 2014.

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

[13] Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[14] Kriegel, H.-P., Pfeifle, M., and Schönauer, S. Similarity search in biological and engineering databases. *IEEE Data Eng. Bull.*, 27(4):37–44, 2004.

[15] Kuhn, H. W. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[16] Li, Y., Gu, C., Dullien, T., Vinyals, O., and Kohli, P. Graph matching networks for learning the similarity of graph structured objects. In *International conference on machine learning*, pp. 3835–3845. PMLR, 2019.

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

[18] Ling, X., Wu, L., Wang, S., Ma, T., Xu, F., Liu, A. X., Wu, C., and Ji, S. Multilevel graph matching networks for deep graph similarity learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[19] Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 4602–4609, 2019.

[20] Neuhaus, M., Riesen, K., and Bunke, H. 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)*, pp. 163–172. Springer, 2006.

[21] Ogaard, K., Roy, H., Kase, S., Nagi, R., Sambhoos, K., and Sudit, M. Discovering patterns in social networks with graph matching algorithms. In *International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction*, pp. 341–349. Springer, 2013.

[22] Qin, C., Zhao, H., Wang, L., Wang, H., Zhang, Y., and Fu, Y. Slow learning and fast inference: Efficient graph similarity computation via knowledge distillation. *Advances in Neural Information Processing Systems*, 34, 2021.

[23] Qin, Z., Bai, Y., and Sun, Y. Ghashing: semantic graph hashing for approximate similarity search in graph databases. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2062–2072, 2020.

[24] Ranu, S., Calhoun, B. T., Singh, A. K., and Swamidass, S. J. Probabilistic substructure mining from small-molecule screens. *Molecular Informatics*, 30(9):809–815, 2011.

[25] Riesen, K., Emmenegger, S., and Bunke, H. A novel software toolkit for graph edit distance computation. In *International Workshop on Graph-Based Representations in Pattern Recognition*, pp. 142–151. Springer, 2013.

[26] Socher, R., Chen, D., Manning, C. D., and Ng, A. Reasoning with neural tensor networks for knowledge base completion. *Advances in neural information processing systems*, 26, 2013.

[27] Steinhaeuser, K. and Chawla, N. V. Community detection in a large real-world social network. In *Social computing, behavioral modeling, and prediction*, pp. 168–175. Springer, 2008.

[28] Tian, Y., Mceachin, R. C., Santos, C., States, D. J., and Patel, J. M. Saga: a subgraph matching tool for biological graphs. *Bioinformatics*, 23(2):232–239, 2007.

[29] Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[30] Wang, R., Zhang, T., Yu, T., Yan, J., and Yang, X. Combinatorial learning of graph edit distance via dynamic embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5241–5250, 2021.

[31] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1): 4–24, 2020.

[32] Xu, H., Chen, R., Bai, Y., Duan, Z., Feng, J., Sun, Y., and Wang, W. Cosimgnn: Towards large-scale graph similarity computation. *arXiv preprint arXiv:2005.07115*, 2020.

[33] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ryGs6iA5Km`.

[34] Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [N/A]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3
   (b) Did you include complete proofs of all theoretical results? [Yes] See Section 3
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Supplemental Material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix A
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Appendix B.3
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 5.4
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]