Disentangled Contrastive Learning on Graphs

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recently, self-supervised learning for graph neural networks (GNNs) has attracted considerable attention because of their notable successes in learning the representation of graph-structure data. However, the formation of a real-world graph typically arises from the highly complex interaction of many latent factors. The existing self-supervised learning methods for GNNs are inherently holistic and neglect the entanglement of the latent factors, resulting in the learned representations suboptimal for downstream tasks and difficult to be interpreted. Learning disentangled graph representations with self-supervised learning poses great challenges and remains largely ignored by the existing literature. In this paper, we introduce the Disentangled Graph Contrastive Learning (**DGCL**) method, which is able to learn disentangled graph-level representations with self-supervision. In particular, we first identify the latent factors of the input graph and derive its factorized representations. Each of the factorized representations describes a latent and disentangled aspect pertinent to a specific latent factor of the graph. Then we propose a novel factor-wise discrimination objective in a contrastive learning manner, which can force the factorized representations to independently reflect the expressive information from different latent factors. Extensive experiments on both synthetic and real-world datasets demonstrate the superiority of our method against several state-of-the-art baselines, which achieves a new SOTA in self-supervised graph representation learning on nine graph classification datasets and one new Open Graph Benchmark.

1 Introduction

Graph structured data is ubiquitous in the real world, e.g., social networks, biology networks, traffic networks, etc. Recently, graph neural networks (GNNs) have become increasingly prevalent in learning graph representations in a supervised manner, demonstrating their strength in a wide variety of research fields [22, 18, 40, 45]. GNNs require task-dependent annotated labels to learn effective representations, which are extremely scarce, or even unavailable in practice, thus motivating the advent of self-supervised graph representation learning.

Contrastive learning, as a discriminative approach pulling similar samples close and pushing dissimilar samples far away, has become a dominant strategy in self-supervised graph representation learning [37, 32, 41, 29, 30, 9, 31]. Despite their notable successes, the existing graph contrastive learning methods generally adopt a holistic scheme, i.e., the learned representations characterize graphs as a perceptual whole, ignoring the nuances between different aspects of the graph. In fact, the formation of a graph typically follows a relational process in the real world, driven by many complex latent factors. For example, in social networks, a social group may have several communities originated from different relations (e.g., friends, colleagues, etc.) or interests (e.g., sports, games, etc.) [47]. And a molecular graph may consist of various groups of atoms and bonds representing different functional units [48]. The complex relations among the multiple latent factors bring an urge for disentangling these factors

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

in contrastive graph representation learning, which remains unexplored by the existing holistic works. As a result, the graph representations learned by the existing methods contain a mixture of entangled factors, harming interpretability and leading to suboptimal performance for predictive tasks involving whole graph representations.

In this paper, we propose to learn disentangled contrastive graph representation, for the first time. Although disentangled representation learning, which aims to characterize the various underlying explanatory factors behind the observed data in different parts of the factorized representations [2, 11], has been demonstrated to be more explainable [47] and generalizable [2], disentangled graph contrastive learning faces the following two challenges. (1) Tailored graph encoder for disentangled contrastive learning. The graph encoder should be carefully designed so that it can be sufficiently expressive to infer the disentangled latent factors in the graph. (2) Tailored discrimination tasks designed for disentangled graph contrastive learning. Since task-dependent labels are not available in the self-supervised setting, disentangled graph contrastive learning can only utilize the limited amount of self-supervision information. This implies that the discrimination tasks should be well-designed for disentangled contrastive representation learning on graphs.

To tackle these challenges, we propose a novel disentangled graph contrastive learning (**DGCL**) model capable of disentangled contrastive learning on graphs. In particular, we first design a disentangled graph encoder whose key ingredient is a multi-channel message-passing layer. Each channel is tailored to aggregate features only from one disentangled latent factor. Then a separate readout operation in each channel summarizes the specific aspect of the graph according to the corresponding latent factor, so as to produce the disentangled graph representation. Next, we conduct contrastive learning in each representation subspace characterized by each factor independently instead of in the whole representation space. This novel factor-wise contrastive approach can ensure that each disentangled factor of the vectorized representations is sufficiently discriminative only under one specific aspect of the whole graph. Thus the representations are encouraged to be disentangled and best characterize the aspect pertinent to a latent factor of the graph. In addition, we employ a variant of the Expectation-Maximization (EM) algorithm to optimize the proposed disentangled graph encoder. We iteratively perform E-step as inferring the latent factors and M-step as learning the parameters of our disentangled graph encoder via the proposed factor-wise contrastive learning. Compared with the existing methods, our proposed **DGCL** model encodes a graph with multiple disentangled representations, making it possible to explore the meaning of each channel, which benefits in more explainability for producing graph representations.

We conduct extensive experiments on both synthetic graph dataset and empirical well-known graph benchmarks. The results show that the representations learned from **DGCL** can achieve substantial performance gains on the downstream graph classification task compared with various state-of-the-art baselines. Specifically, our proposed **DGCL** outperforms all the baselines on nine widely used graph classification datasets and one new Open Graph Benchmark dataset.

The contributions of this paper are summarized as follows:

- We propose a disentangled graph contrastive learning model (**DGCL**), which is able to learn disentangled graph representation via factor-wise contrastive learning. To the best of our knowledge, we are the first to study disentangled self-supervised graph representation learning.
- We propose a disentangled graph encoder to capture multiple aspects of graphs through learning disentangled latent factors on graphs. We further present the factor-wise contrastive learning approach on tailored discrimination tasks in terms of each latent factor independently.
- We employ a variant of the Expectation-Maximization (EM) algorithm to optimize the parameters of our proposed disentangled graph encoder and conduct extensive experiments to verify the efficacy of our proposed model for the graph classification task. Results on nine graph classification datasets and one new dataset from Open Graph Benchmark demonstrate that **DGCL** achieves state-of-the-art performance by significantly outperforming all the existing baselines.

2 Problem Formulation and Preliminaries

Problem Formulation. Let $\mathbf{G} = \{G_i\}_{i=1}^N$ be a graph dataset with N graphs. The key of most self-supervised graph representation learning methods, including ours, is to derive a graph encoder $f(\cdot)$, which outputs a d-dimensional representation $\mathbf{z}_i = f(G_i) \in \mathbb{R}^d$ for each input graph, such that

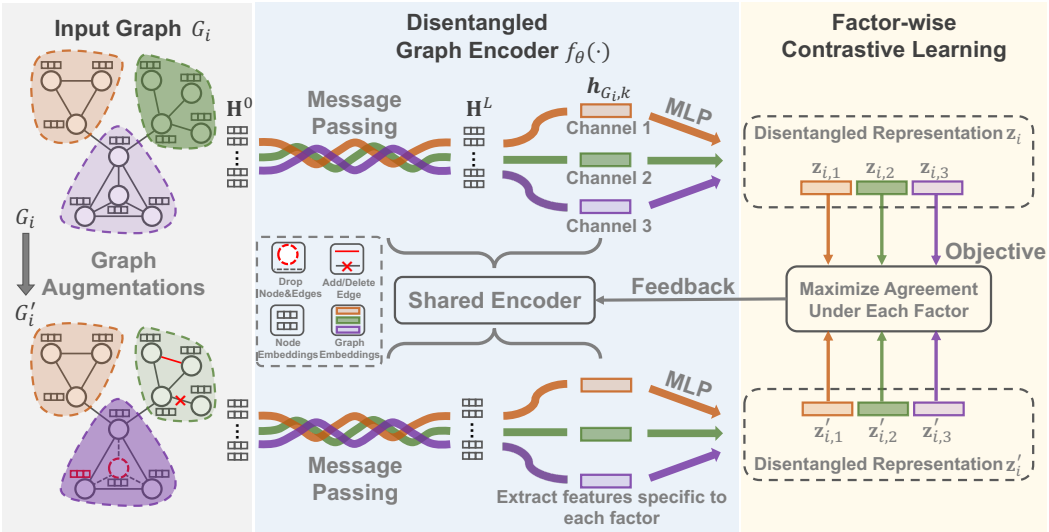


Figure 1: The framework of **DGCL** Model. (1) The input graph G_i undergoes graph augmentations to produce G_i', and both of them are fed into the shared disentangled graph encoder $f_\theta(\cdot)$. (2) In the encoder $f_\theta(\cdot)$, the node features $\mathbf{H}^0$ are first aggregated by L layers message passing and then taken as the input of a multichannel message passing layer. (3) Based on the disentangled graph representation $\mathbf{z}_i$, the factor-wise contrastive learning aims to maximize the agreement under each latent factor and provide feedback for the encoder to improve the disentanglement. This example assumes that there are three latent factors, hence the three channels.

$\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^N$ best describes $\mathbf{G}$. In this work, we aim to learn a multichannel graph encoder $f_\theta(\cdot)$ with parameters θ, so that the output $\mathbf{z}_i$ can be a disentangled representation, i.e. $f_\theta(\cdot) = \{f_\theta^{(k)}(\cdot)\}_{k=1}^K$, where K is the number of channels. To be specific, $\mathbf{z}_i$ is expected to be composed of K independent components, i.e., $\mathbf{z}_i = [\mathbf{z}_{i,1}, \mathbf{z}_{i,2}, \ldots, \mathbf{z}_{i,K}]$, where $\mathbf{z}_{i,k} = f_\theta^{(k)}(G_i) \in \mathbb{R}^{\Delta d}$, $k \in [1, K]$, $\Delta d = d/K$, assuming that there are K latent factors behind the graph instances to be disentangled. The kth component $\mathbf{z}_{i,k}$ is for characterizing the aspect of G_i that is pertinent to factor k accurately. We also assume that the value of $\mathbf{z}_{i,k}$ will be merely a white noise vector if the input graph G_i does not contain any information of factor k. Note that we focus primarily on undirected graphs in our method, although it can be straightforwardly extended to directed graphs.

Preliminaries on Contrastive Learning. Unlike generative models, contrastive learning is an instance-wise *discriminative* approach that aims at making similar instances closer and dissimilar instances far from each other in representation space [15, 20]. It treats each instance in the dataset as a distinct class of its own and trains a classifier to distinguish between individual instance classes [43]. Given a dataset $\mathbf{X} = \{x_i\}_{i=1}^N$, each instance x_i is assigned with a unique surrogate label y_i, since no ground-truth labels are given. y_i is often regarded as the index of the instance in the dataset [43], i.e., $y_i = i$. So the probability classifier is defined as:

$$p_\theta(y_i|x_i) = \frac{\exp \phi(\mathrm{v}_i, \mathrm{v}'_{y_i})}{\sum_{j=1}^N \exp \phi(\mathrm{v}_i, \mathrm{v}'_{y_j})}, \tag{1}$$

where θ denotes the parameters of the encoder. Both v_i and v'_{y_i} are the embeddings from x_i, which are generated from two different encoders [38], or a shared encoder [5]. Before being passed into the encoder, the input x_i could undergo data augmentations [5], which play a critical role in defining effective predictive tasks for learning the encoder. ϕ is the similarity function, often adopting cosine similarity with temperature τ [42], i.e., $\phi(\mathrm{v}_i, \mathrm{v}'_{y_i}) = \mathrm{v}'^{\top}_{y_i}\mathrm{v}_i/\tau$, assuming the embeddings are ℓ^2-normalized. Then the learning objective is to maximize the joint probability $\prod_{i=1}^N p(y_i|x_i)$ over the dataset, namely minimize the log-likelihood function $\sum_{i=1}^N \ell_i$, if let $\ell_i = -\log p(y_i|x_i)$. Note that loss ℓ_i could be NCE loss [43], InfoNCE loss [28], or NT-Xent loss [5]. The encoder will be encouraged to learn a representation space where samples (e.g., augmented data) from the same instance (e.g., an image, a graph) are pulled closer and samples from different instances are pushed apart [15]. For convenience, we follow the settings above in this work.

3 Disentangled Graph Contrastive Learning

In this section, we present the proposed **DGCL** model. The framework of **DGCL** is shown in Fig. 1. In Section 3.1, we introduce the disentangled graph encoder to identify the complex latent factors and capture multiple aspects of graphs. Then in Section 3.2, we propose a factor-wise contrastive learning approach to conduct instance discrimination under each latent factor independently. The objective of our method is derived in Section 3.3, and in Section 3.4, we develop a variant of the Expectation-Maximization (EM) algorithm to solve the nontrivial inference and learning problems caused by the complex latent factors behind the graphs.

3.1 Disentangled Graph Encoder

The key of the disentangled graph encoder is to produce the factorized graph representation $\mathbf{z}_i = [\mathbf{z}_{i,1}, \mathbf{z}_{i,2}, \ldots, \mathbf{z}_{i,K}]$ for each input graph $G_i \in \mathbf{G}$. Based on the factorized representation, we can infer the related latent factors behind the graph.

Generally, GNNs use the graph structure and node features to learn the representation vector $\boldsymbol{h}_v$ of each node v with a message-passing mechanism, i.e., iteratively updating the representation of a node by aggregating representations of its neighbors. The propagation of the lth layer is formulated as:

$$\boldsymbol{h}_v^l = \text{COMBINE}^l(\boldsymbol{h}_v^{l-1}, \text{AGGREGATE}^l(\{\boldsymbol{h}_u^{l-1} : u \in \mathcal{N}(v)\})), \tag{2}$$

where $\boldsymbol{h}_v^l$ is the representation of node v at the lth layer and $\boldsymbol{h}_v^0$ is initialized with node features. $\mathcal{N}(v)$ is the neighborhood to node v. We use the term GNN to indicate the message-passing layer in Eq. (2).

Let $\mathbf{H}^l = \{\boldsymbol{h}_v^l | v \in V\}$ be the node embeddings after the lth GNN, where V denotes the node set of the graph. After applying L traditional message-passing layers, we propose a graph-disentanglement layer to learn the disentangled representations. The goal is to extract features specific to each latent factor with a separate channel. Specifically, we adopt K separate channels to identify the complex heterogeneous latent factors and capture multiple aspects of the input graph. For each channel, we first utilize a GNN_k to propagate information with its own parameters: $\mathbf{H}_k^{L+1} = \text{GNN}_k(\mathbf{H}^L, A)$, where A is the adjacency matrix of the input graph. $\mathbf{H}_k^{L+1}$ is the node embeddings which is only pertinent to the kth latent factor. Then the READOUT function of each channel, which is realized with mean pooling in our method, is used to summarize all the obtained node representations into a fixed-length graph-level representation: $\boldsymbol{h}_{G_i,k} = \text{READOUT}_k(\{\mathbf{H}_k^{L+1}\})$. Finally, each channel outputs the factorized graph representation with a separate MLP: $\mathbf{z}_{i,k} = \text{MLP}_k(\boldsymbol{h}_{G_i,k})$.

Compared with the existing graph encoders that are inherently holistic, our disentangled graph encoder consists of K channels, rending the possibility to identify the complex heterogeneous latent factors and capture multiple aspects of graphs.

3.2 Disentangled Factor-wise Contrastive Learning

Unlike the existing contrastive learning methods, **DGCL** designs a novel factor-wise instance discriminative task and learns to solve this task under each latent factor independently. This design not only makes similar samples closer and dissimilar samples far from each other in the representation space, but also encourages the learned representation to incorporate factor-level information for disentanglement.

Specifically, we assume that the formation of real-world graphs is usually driven by multiple latent heterogeneous factors. So the instance discriminative task should be represented as the expectation of several subtasks under the latent factors: $p_\theta(y_i|G_i) = \mathbb{E}_{p_\theta(k|G_i)}\left[p_\theta(y_i|G_i, k)\right]$. Here $p_\theta(k|G_i)$ is the probability distribution over latent factors for the input graph G_i. $p_\theta(y_i|G_i, k)$ denotes the instance discrimination subtask under the kth latent factor.

Firstly, given the representation $\mathbf{z}_i$ of G_i derived from the disentangled graph encoder $f_\theta(\cdot)$, we present a prototype-based method to obtain $p_\theta(k|G_i)$. We introduce K latent factor prototypes $\{\mathbf{c}_k\}_{k=1}^K$, and the probability of the kth latent factor reflected in G_i is parameterized as:

$$p_\theta(k|G_i) = \frac{\exp \phi(\mathbf{z}_{i,k}, \mathbf{c}_k)}{\sum_{k=1}^K \exp \phi(\mathbf{z}_{i,k}, \mathbf{c}_k)}, \tag{3}$$

where ϕ is the cosine similarity with temperature τ, i.e., $\phi(\mathbf{a}, \mathbf{b}) = \text{COSINE}(\mathbf{a}, \mathbf{b})/\tau$ and $\text{COSINE}(\mathbf{a}, \mathbf{b}) = \mathbf{a}^\top \mathbf{b}/(\|\mathbf{a}\|_2 \|\mathbf{b}\|_2)$.

Then, we define the instance discrimination subtask under the kth latent factor as:

$$p_\theta(y_i|G_i, k) = \frac{\exp \phi(\mathbf{z}_{i,k}, \mathbf{z}'_{y_i,k})}{\sum_{j=1}^{N} \exp \phi(\mathbf{z}_{i,k}, \mathbf{z}'_{y_j,k})}, \tag{4}$$

where $\mathbf{z}_{i,k}$ and $\mathbf{z}'_{y_i,k}$ are the disentangled representations produced by the shared graph encoder, and y_i is the unique surrogate label (see Section 2) of the graph G_i. In our method, we follow [43] to implement y_i as the ID of the graph in the dataset, i.e., $y_i = i$. For notation convenience, we do not distinguish y_i and i hereafter when there is no risk of confusion.

Next, we describe the process to get $\mathbf{z}'_{i,k}$, i.e. $\mathbf{z}'_{y_i,k}$ in Eq. (4). First, the input graph G_i undergoes graph data augmentations to obtain its correlated views G'_i, and they form a positive pair. Data augmentation is expected to create novel and realistically rational data by applying certain transformations that do not affect the label, and plays a critical role in defining effective predictive tasks [5, 49]. We follow [49] to adopt four types of graph augmentation strategies, including node dropping, edge perturbation, attribute masking, and subgraph sampling. More details of graph augmentations can be found in supplementary material. Then, the augmented graph G'_i is also fed into the shared disentangled graph encoder $f_\theta(\cdot)$ to produce $\mathbf{z}'_{i,k}$, i.e. $\mathbf{z}'_{y_i,k}$. Given the disentangled representations $\mathbf{z}_{i,k}$ and $\mathbf{z}'_{i,k}$ of the G_i and G'_i respectively, we conduct factor-wise contrastive learning for each latent factor independently as Eq. (4).

3.3 Evidence Lower Bound (ELBO)

Next, we present the objective of our method. Following the existing methods, we aim to maximize the joint probability $\prod_{i=1}^{N} p(y_i|G_i)$ over the graph dataset $\mathbf{G} = \{G_i\}_{i=1}^{N}$. We learn the model parameters θ by maximizing the log-likelihood:

$$\theta^* = \arg\max_\theta \sum_{i=1}^{N} \log p_\theta(y_i|G_i) = \arg\max_\theta \sum_{i=1}^{N} \log \mathbb{E}_{p_\theta(k|G_i)} \left[p_\theta(y_i|G_i, k)\right]. \tag{5}$$

However, directly maximizing the log-likelihood function is difficult because of the latent factors. Therefore, we instead optimize the evidence lower bound (ELBO) of the log-likelihood function given by Theorem 3.1. See the supplementary material for the proof.

Theorem 3.1. *The log likelihood function of each graph* $\log p_\theta(y_i|G_i)$ *is lower bounded by the ELBO:* $\mathcal{L}(\theta, i) = \mathbb{E}_{q_\theta(k|G_i,y_i)}[\log p_\theta(y_i|G_i, k)] - D_{KL}(q_\theta(k|G_i, y_i) \parallel p_\theta(k|G_i))$.

To make the equality in Eq. (9) hold, we require $q_\theta(k|G_i, y_i)$ equals to $p_\theta(k|G_i, y_i)$, whose detailed implementations are provided in the next subsection (see Eq. (6) and Eq. (8)). In the ELBO $\mathcal{L}(\theta, i)$, $p_\theta(y_i|G_i, k)$ and $p_\theta(k|G_i)$ have been introduced in Eq. (4) and Eq. (3), respectively, and $q_\theta(k|G_i, y_i)$ is a variational distribution to infer the posterior distribution of the latent factors after observing both G_i and its correlated view G'_{y_i}.

3.4 Expectation-Maximization

We introduce a variant of the Expectation-Maximization (EM) algorithm [27] to deal with the nontrival inference and learning of maximizing ELBO.

Inference (E-step). $q_\theta(k|G_i, y_i)$ is a variational distribution to infer the posterior probability $p_\theta(k|G_i, y_i)$ that is defined with Bayes' theorem as follows:

$$p_\theta(k|G_i, y_i) = \frac{p_\theta(k|G_i) p_\theta(y_i|G_i, k)}{\sum_{k=1}^{K} p_\theta(k|G_i) p_\theta(y_i|G_i, k)}. \tag{6}$$

$p_\theta(k|G_i, y_i)$ is the probability of the kth latent factor pertinent to both G_i and the augmented G'_i simultaneously. Compared with the prior distribution $p_\theta(k|G_i)$ in Eq. (3), $p_\theta(k|G_i, y_i)$ incorporates more useful information (i.e., factor-wise similarity) from $p_\theta(y_i|G_i, k)$. Although both $p_\theta(k|G_i)$ and $p_\theta(k|G_i, y_i)$ are designed to infer the latent factor distribution, $p_\theta(k|G_i)$ is calculated only given the

Table 1: Graph classification accuracy (%) of **DGCL** and baselines. In each column, the boldfaced score denotes the best result and the underlined score represents the second-best result. "–" indicates the result is not reported in the paper.

	MUTAG	**PTC-MR**	**PROTEINS**	**NCI1**	**IMDB-B**	**IMDB-M**	**RDT-B**	**RDT-M5K**	**COLLAB**
SP	85.2±2.4	58.2±2.4	75.1±0.5	73.0±0.2	55.6±0.2	38.0±0.3	64.1±0.1	39.6±0.2	–
GK	81.7±2.1	57.3±1.4	71.7±0.6	62.3±0.3	65.9±1.0	43.9±0.4	77.3±0.2	41.0±0.2	72.8±0.3
WL	80.7±3.0	58.0±0.5	72.9±0.6	80.0±0.5	72.3±3.4	47.0±0.5	68.8±0.4	46.1±0.2	–
DGK	87.4±2.7	60.1±2.6	73.3±0.8	80.3±0.5	67.0±0.6	44.6±0.5	78.0±0.4	41.3±0.2	73.1±0.3
MLG	87.9±1.6	63.3±1.5	76.1±2.0	80.8±1.3	66.6±0.3	41.2±0.0	–	–	–
node2vec	72.6±10.2	58.6±8.0	57.5±3.6	54.9±1.6	–	–	–	–	–
sub2vec	61.1±15.8	60.0±6.4	53.0±5.6	52.8±1.5	55.3±1.5	36.7±0.8	71.5±0.4	36.7±0.4	–
graph2vec	83.2±9.3	60.2±6.9	73.3±2.1	73.2±1.8	71.1±0.5	50.4±0.9	75.8±1.0	47.9±0.3	–
GVAE	87.7±0.7	61.2±1.8	–	–	70.7±0.7	49.3±0.4	87.1±0.1	52.8±0.2	–
InfoGraph	89.0±1.1	61.7±1.4	74.4±0.3	76.2±1.1	73.0±0.9	49.7±0.5	82.5±1.4	53.5±1.0	70.7±1.1
GCC	–	–	–	–	72.0	49.4	89.8	53.7	78.9
MVGRL	89.7±1.1	62.5±1.7	–	–	74.2±0.7	51.2±0.5	84.5±0.6	–	–
GraphCL	86.8±1.3	63.6±1.8	74.4±0.5	77.9±0.4	71.1±0.4	50.7±0.4	89.5±0.8	56.0±0.3	71.4±1.2
DGCL	**92.1±0.2**	**65.8±1.5**	**76.4±0.5**	**81.9±0.2**	**75.9±0.7**	**51.9±0.4**	**92.7±0.2**	**56.1±0.2**	**81.2±0.3**

graph G_i, but $p_\theta(k|G_i, y_i)$ is calculated after observing G_i, the augmented version G'_{y_i}, and their similarities under the specific latent factor.

However, we cannot compute the posterior probability tractably because of the term $p_\theta(y_i|G_i, k)$. If we directly calculate $p_\theta(y_i|G_i, k)$ according to Eq. (4), all the instances in the dataset $\mathbf{G}$ are needed for computing the denominator in Eq. (4) , which could be computationally prohibitive [25, 43]. To tackle this obstacle, several strategies are proposed in the literature, including memory bank [43, 38], dynamic dictionary [10], NT-Xent loss [5]. Here, we adopt NT-Xent loss on a minibatch $\mathcal{B} \subseteq \mathbf{G}$. So in practice, the instance discrimination under each latent factor is approximated by:

$$\hat{p}_\theta(y_i|G_i, k) = \frac{\exp \phi(\mathbf{z}_{i,k}, \mathbf{z}'_{i,k})}{\sum_{j\in\mathcal{B}, j\neq i}^{|\mathcal{B}|} \exp \phi(\mathbf{z}_{i,k}, \mathbf{z}'_{j,k})}. \tag{7}$$

Finally, we approximate the posterior probability $p_\theta(k|G_i, y_i)$ with a variational distribution defined as the inference step:

$$q_\theta(k|G_i, y_i) = \frac{p_\theta(k|G_i)\hat{p}_\theta(y_i|G_i, k)}{\sum_{k=1}^{K} p_\theta(k|G_i)\hat{p}_\theta(y_i|G_i, k)}. \tag{8}$$

Learning (M-step). In the M-step, we seek to learn the parameters θ of the disentangled graph encoder. More specifically, we fix $q_\theta(k|G_i, y_i)$ and maximize the ELBO over a mini-batch $\mathcal{B}$ using stochastic mini-batch gradient ascent: $\mathcal{L}(\theta, \mathcal{B}) = \sum_{i\in\mathcal{B}} \mathcal{L}(\theta, i)$.

Encourage disentanglement. Note that our objective and its optimization can inherently encourage disentanglement without adding extra regularization term (e.g., minimizing mutual information). The reason is that factorizing the instance discrimination into K factor-wise subtasks will enforce the independence of the learned graph representation $\mathbf{z}_i$. Besides, $q_\theta(k|G_i, y_i)$ is computed based on k^{th} and other $K-1$ latent factors. Thus, the graph encoder is forced to preserve exclusive information in each channel to get more accurate approximation to the posterior, if a tighter ELBO is expected. The strong inductive biases in **DGCL** encourage to learn disentangled graph representations that match the ground truth factors behind the graphs.

4 Experiments

We empirically evaluate our proposed method through experiments, and analyze its behavior on synthetic graph dataset to gain deeper insight. Ablation studies are conducted to show the effectiveness of the proposed method. We analyze the impact of the hyper-parameters in supplementary material.

4.1 Experimental Setup

Datasets. To demonstrate the advantages of our method, we conduct experiments on nine well-known graph classification datasets including four bioinformatics datasets, i.e., MUTAG, PTC-MR, NCI1, PROTEINS, and five social network datasets, i.e., COLLAB, IMDB-BINARY, IMDB-MULTI,

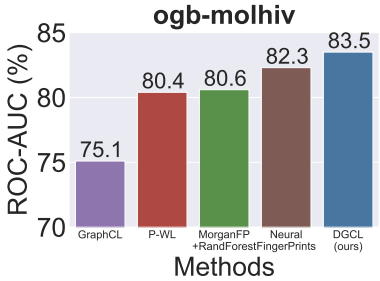


Figure 2: Performance on ogb-molhiv, an Open Graph Benchmark dataset.

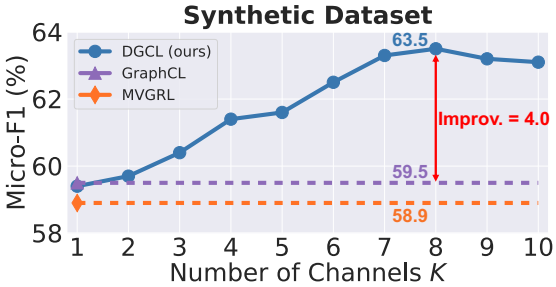


Figure 3: Micro-F1 (%) of two baselines and our **DGCL** with different number of channels K.

REDDIT-BINARY, and REDDIT-MULTI-5K. We also adopt a larger graph dataset ogbg-molhiv from Open Graph BenchMark (OGB) [14]. More details are provided in the supplementary material.

Baselines. We compare **DGCL** with the following two groups of baselines. One group of baselines are graph kernels including Shortest Path Kernel (SP) [3], Graphlet Kernel (GK) [34], Weisfeiler-Lehman Sub-tree Kernel (WL) [33], Deep Graph Kernels (DGK) [46], and Multi-Scale Laplacian Kernel (MLG) [19]. The other group of baselines are classical unsupervised graph representation learning methods including node2vec [7], sub2vec [1], graph2vec [26], GVAE [17], and more recent contrastive graph representation learning methods including InfoGraph [37], GCC [31], MVGRL [9], and GraphCL [49]. See the supplementary material for more details.

Evaluation. To verify the effectiveness of our method, we follow the common evaluation protocols in the existing literature [26, 37, 46, 49], where graph embeddings are generated from each method and then fed into a downstream SVM classifier. We adopt the 10-fold cross validation accuracy, and report the mean accuracy (%) with standard variation after five repeated runs.

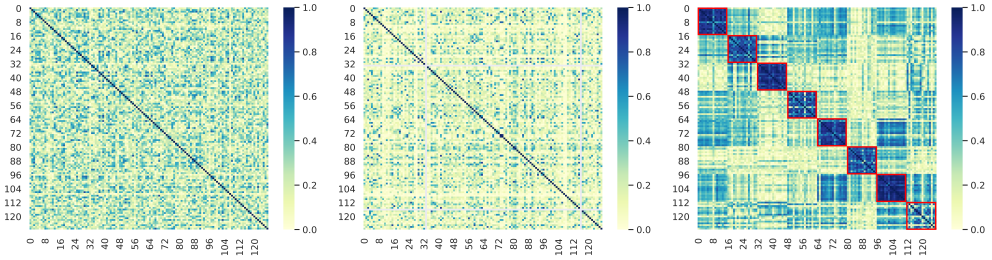
Implementation Details. For a fair comparison, the hyper-parameters of the graph augmentations are kept consistent with GraphCL. We use GIN [45] as the message-passing layers since it is shown to be one of the most expressive message-passing GNNs. Since the ground-truth number of the latent factors is unknown, we search the number of channels K from 1 to 10. More details about graph augmentation and hyper-parameters can be found in supplementary material.

4.2 Results on Real Benchmark Graphs

The results are reported in Table 1. We can see that the graph contrastive learning methods generally outperform the graph kernel methods or the classical unsupervised methods, which verify the effectiveness of contrastive learning. Our method **DGCL** consistently achieves the best performance compared with other contrastive methods (e.g., MVGRL, GrpahCL) and classical unsupervised methods (e.g., graph2vec, GVAE), demonstrating the superiority of our method. For example, our method increases the classification accuracy by 2.4%, 2.2%, and 2.9% compared with the second-best methods on MUTAG, PTC-MR, and RDT-B, respectively. We attribute the results to the fact that these existing methods fail to identify the underlying latent factors which are important in preserving graph properties and can not learn the disentangled representations. In contrast, we disentangle graph representations to explicitly consider the entanglement of heterogeneous factors. When compared to graph kernel methods, our method also has the best accuracy on all the datasets. Notice that none of these kernel methods is consistently competitive across all of the datasets, as opposed to our method.

The task on the ogbg-molhiv dataset is binary classification evaluated by ROC-AUC metric, instead of accuracy in the experiments above. Since our method is self-supervised, we train a logistic regression classifier on the learned graph embeddings. We adopt the provided evaluator and dataset splits for a fair comparison. We compare **DGCL** with the strong self-supervised baseline GraphCL and the top-3 ranking methods on the current leaderboard [1]: Neural FingerPrints, MorganFP+RandForest, and P-WL. We show the results in Figure 2. **DGCL** achieves significant performance improvement over GraphCL and even surpasses the current strongest method with a margin of 1.2% on the ROC-AUC. This result illustrates that our method is also able to handle large-scale graphs, demonstrating the benefit of learning disentangled graph representations in the contrastive manner.

[1] `https://ogb.stanford.edu/docs/leader_graphprop/#ogbg-molhiv`



(a) MVGRL (b) GraphCL (c) **DGCL**

Figure 4: An analysis of feature correlation on the synthetic graphs with eight latent factors. The figures show the absolute value of the correlations between the elements of the representations learned by MVGRL, GraphCL, and **DGCL** with eight channels, respectively. We can see that the representations generated from **DGCL** present a more block-wise correlation pattern, indicating that the eight channels of the disentangled graph encoder in **DGCL** are able to capture mutually exclusive information and the latent features have indeed been disentangled.

Table 2: Ablation Studies on the variants of our method.

	MUTAG	PTC-MR	PROTEINS	NCI1	IMDB-B	IMDB-M	RDT-B	RDT-M5K	COLLAB
DGCL	**92.1±0.2**	**65.8±1.5**	**76.4±0.5**	**81.9±0.2**	**75.9±0.7**	**51.9±0.4**	**92.7±0.2**	**56.1±0.2**	**81.2±0.3**
Variant 1	89.3±0.3	64.3±1.3	74.9±0.2	78.5±0.5	73.4±0.5	50.3±0.2	91.1±0.7	55.9±0.3	77.5±0.4
Variant 2	86.7±0.4	63.7±1.1	74.1±0.5	78.1±0.3	71.6±0.4	49.9±0.4	90.1±0.4	55.8±0.3	75.5±0.2
Variant 3	86.5±0.6	63.5±1.6	73.9±0.6	77.7±0.6	70.9±0.5	49.8±0.3	89.7±0.6	55.7±0.2	71.5±0.2

4.3 Results on Synthetic Graphs

To further investigate the behavior of our method, we generate a synthetic dataset consisting of 1,000 graphs with known latent factors. Specifically, we generate synthetic graphs using the stochastic block model [13]. Each graph contains four communities and each community consists of 10 nodes. We define the latent factor as the probability p that two nodes are connected in a community. p can take value from $\{0.2, 0.3, \ldots, 0.9\}$, meaning that there are eight latent factors in the dataset in total. The probability for each community is drawn from the eight possible choices without replacement. Two nodes in different communities are connected with probability 0.05. The rows of the adjacency matrices are used as node features, and the ground-truth communities are used as labels, i.e., there are 8 classes and each graph has 4 labels. We train our method and two baseline methods, i.e., MVGRL and GraphCL on the generated synthetic dataset with self-supervision. Then we adopt the SVM classifier on the learned graph embeddings for each method and use the Micro-F1 (%) as the evaluation metric. Other settings are the same as Section 4.1.

We vary the number of channels K of our method and report the results in Figure 3. Our method reports better performance than the baselines. We also find that as K increases from 1 to 8, the results of **DGCL** improves, which verifies the importance of disentangling latent factors. After reaching the peak at $K = 8$, the performance slightly drops, but in general, our method is not very sensitive when K is not too large. Our method achieves the best results when K is equal to the ground-truth number of latent factors, indicating we can model the underlying structure of this simulation dataset.

Besides the quantitative evaluation, we also provide a qualitative evaluation by plotting the correlation of the latent features in Figure 4. The figure shows the absolute values of the correlation between the elements of 128-dimensional graph representation obtained from MVGRL, GraphCL, and our **DGCL** ($K = 8$) on the synthetic dataset. We can see from the results that the graph representations of MVGRL and GraphCL are entangled. In comparison, the correlation of our **DGCL** shows eight diagonal blocks, meaning that the channels of **DGCL** likely extract mutually exclusive information and output disentangled representations.

4.4 Ablation Studies

We perform ablation studies over a number of key components of our method to understand their functionalities more deeply. We compare **DGCL** with the following three variants: (1) Variant 1: it sets $p_\theta(k|G_i) = 1/K$ a uniform distribution of latent factors; (2) Variant 2: it keeps one output

channel for disentangled graph encoder $f(\cdot)$, and the only output embedding $\mathbf{z}_{i,k}$ is then assigned to $\mathbf{z}_{i,1}, \mathbf{z}_{i,2}, \ldots, \mathbf{z}_{i,K}$. (3) Variant 3: it sets $K = 1$ directly, so that our method will degenerate to the traditional entangled graph contrastive learning model. The results of **DGCL** and its variants are shown in Table 2. We observe a drop in performance of Variant 1, demonstrating the efficacy of inferring the latent factors of the graphs. In Variant 2, the latent factors are entangled in the graph representation, potentially making difficulties for the follow-up factor-wise contrastive learning. The deterioration of performance verifies the significance of disentangled graph encoder. We observe similar results in Variant 3 compared with entangled graph contrastive learning baselines. The drop in performance indicates the significance of the proposed factor-wise contrastive learning.

5 Related Work

Graph Neural Networks. Recently, graph neural networks (GNNs) [22, 18, 40, 45] have revolutionized the field of graph representation learning. GNNs generally adopt a neighborhood aggregation (message passing) paradigm, i.e., the embedding of node is iteratively updated by aggregating embeddings of its neighbors [40, 45]. The representation of the whole graph is summarized on node embeddings through the readout function, i.e., graph pooling [45]. However, for achieving state-of-the-art performance, most famous GNNs, including their variants, are trained end-to-end with task-specific labels, which could be extremely scarce for some graph datasets. Compared with these supervised models, our model, based on self-supervised contrastive learning, can largely eliminate the over-dependence on the manual labels, which is crucial for graph representation learning.

Contrastive Learning on Graphs. Recently, contrastive learning, adopting the instance discrimination as the pretext task, has become a dominant component in self-supervised learning methods [10, 5, 43, 39, 28]. Some literatures utilizing contrastive learning for graph data are proposed [37, 31, 9, 49, 50, 32]. The key of these methods is to maximize the agreement (i.e., similarity) between proper transformations or different views of the input graph. However, the existing graph contrastive learning methods explore general settings where entanglement is severe and do not incorporate disentangled representation learning. They fail to recognize and disentangle the heterogeneous latent factors behind complex graph data. These holistic methods have limited capacity in preserving detailed graph properties, which easily result in suboptimal representations for downstream tasks.

Disentangled Representation Learning. Disentangled representation learning is to learn factorized representations that identify and disentangle the underlying explanatory factors hidden in the observed data [2]. It has been raising a surge of interest in graph-structured data recently [24, 23, 44, 47]. This line of works attempts to learn disentangled representations for graphs but heavily relies on the annotated labels, which largely restricts their applications where labeled data are unavailable or expensive to collect. On the other hand, some works [17, 35] are based on the generative model, namely utilizing Variational Autoencoders (VAEs) on graph for disentanglement, since the hyper-parameter β of VAEs can balance the reconstruction and disentanglement [12, 4]. However, the reconstruction in generative methods could be computationally expensive [8, 20] and even introduce bias that has a negative effect on the learned representation [28]. In addition, the reconstruction for graph-structured data often involves discrete decisions that are not differentiable [35]. How to learn disentangled representation on graph-structured data with contrastive learning is largely unexplored.

6 Conclusions

In this paper, we propose a disentangled graph contrastive learning (**DGCL**) model to solve the problem of disentangled self-supervised graph representation learning for the first time. We design a disentangled graph encoder with a tailored multi-channel message passing layer, which is capable of aggregating features in a disentangled manner. We further propose a factor-wise contrastive learning approach to solve instance discrimination task under each latent factor independently, so that the learned representations of **DGCL** are encouraged to not only best describe the graphs but also be disentangled. Therefore, each component of the disentangled representations tends to characterize a disentangled aspect of the graph that is pertinent to a latent factor. Extensive experiments on both synthetic and real-world datasets demonstrate the superiority of our method against several state-of-the-art baselines in self-supervised graph representation learning.

References

[1] Bijaya Adhikari, Yao Zhang, Naren Ramakrishnan, and B Aditya Prakash. Sub2vec: Feature learning for subgraphs. In *PAKDD*, pages 170–182, 2018.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *TPAMI*, 35(8):1798–1828, 2013.

[3] Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *ICDM'05*, pages 8–pp. IEEE, 2005.

[4] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-vae. *arXiv:1804.03599*, 2018.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv:2002.05709*, 2020.

[6] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[7] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *SIGKDD*, pages 855–864, 2016.

[8] Xiaojie Guo, Liang Zhao, Zhao Qin, Lingfei Wu, Amarda Shehu, and Yanfang Ye. Interpretable deep graph generation with node-edge co-disentanglement. *arXiv:2006.05385*, 2020.

[9] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. *arXiv:2006.05582*, 2020.

[10] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[11] Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. *arXiv:1812.02230*, 2018.

[12] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

[13] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

[14] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv:2005.00687*, 2020.

[15] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2021.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv:1611.07308*, 2016.

[18] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[19] Risi Kondor and Horace Pan. The multiscale laplacian graph kernel. *arXiv:1603.06186*, 2016.

[20] Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. Contrastive representation learning: A framework and review. *IEEE Access*, 2020.

[21] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, volume 32, 2018.

[22] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. Gated graph sequence neural networks. In *ICLR*, 2016.

[23] Yanbei Liu, Xiao Wang, Shu Wu, and Zhitao Xiao. Independence promoted graph disentangled networks. In *AAAI*, pages 4916–4923, 2020.

[24] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. Disentangled graph convolutional networks. In *ICML*, pages 4212–4221, 2019.

[25] Zhuang Ma and Michael Collins. Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. *arXiv:1809.01812*, 2018.

[26] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv:1707.05005*, 2017.

[27] Radford M Neal and Geoffrey E Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*. 1998.

[28] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv:1807.03748*, 2018.

[29] Chanyoung Park, Donghyun Kim, Jiawei Han, and Hwanjo Yu. Unsupervised attributed multiplex network embedding. In *AAAI*, pages 5371–5378, 2020.

[30] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. Graph representation learning via graphical mutual information maximization. In *The Web Conference*, pages 259–270, 2020.

[31] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *SIGKDD*, pages 1150–1160, 2020.

[32] Yuxiang Ren, Bo Liu, Chao Huang, Peng Dai, Liefeng Bo, and Jiawei Zhang. Heterogeneous deep graph infomax. *arXiv:1911.08538*, 2019.

[33] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

[34] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*, pages 488–495. PMLR, 2009.

[35] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, pages 412–422. Springer, 2018.

[36] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NeurIPS*, pages 1857–1865, 2016.

[37] Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *ICLR*, 2019.

[38] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv:1906.05849*, 2019.

[39] Tsung Wei Tsai, Chongxuan Li, and Jun Zhu. Mi{ce}: Mixture of contrastive experts for unsupervised image clustering. In *ICLR*, 2021.

[40] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv:1710.10903*, 2017.

[41] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. 2019.

[42] Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. Normface: L2 hypersphere embedding for face verification. In *ACM MM*, pages 1041–1049, 2017.

[43] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, pages 3733–3742, 2018.

[44] Zhang Xinyi and Lihui Chen. Capsule graph neural network. In *ICLR*, 2018.

[45] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

[46] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *SIGKDD*, pages 1365–1374, 2015.

[47] Yiding Yang, Zunlei Feng, Mingli Song, and Xinchao Wang. Factorizable graph convolutional networks. *arXiv:2010.05421*, 2020.

[48] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*, pages 4800–4810, 2018.

[49] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *NeurIPS*, 2020.

[50] Shichang Zhang, Ziniu Hu, Arjun Subramonian, and Yizhou Sun. Motif-driven contrastive learning of graph representations, 2021.

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? [Yes]
   (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]
   (b) Did you include complete proofs of all theoretical results? [Yes]
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]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (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]
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? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

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]