
Disentangled Generative Graph Representation Learning

Anonymous Author(s)

Affiliation

Address

email

Abstract

1 Recently, generative graph models have shown promising results in learning graph
2 representations through self-supervised methods. However, most existing generative
3 graph representation learning (GRL) approaches rely on random masking
4 across the entire graph, which overlooks the entanglement of learned representations.
5 This oversight results in non-robustness and a lack of explainability. Furthermore,
6 disentangling the learned representations remains a significant challenge
7 and has not been sufficiently explored in GRL research. Based on these insights,
8 this paper introduces **DiGGR (Disentangled Generative Graph Representation
9 Learning)**, a self-supervised learning framework. DiGGR aims to learn latent
10 disentangled factors and utilizes them to guide graph mask modeling, thereby
11 enhancing the disentanglement of learned representations and enabling end-to-end
12 joint learning. Extensive experiments on 11 public datasets for two different graph
13 learning tasks demonstrate that DiGGR consistently outperforms many previous
14 self-supervised methods, verifying the effectiveness of the proposed approach.

15 1 Introduction

16 Self-supervised learning (SSL) has received much attention due to its appealing capacity for learning
17 data representation without label supervision. While contrastive SSL approaches are becoming
18 increasingly utilized on images [Chen et al., 2020] and graphs [You et al., 2020], generative SSL has
19 been gaining significance, driven by groundbreaking practices such as BERT for language [Devlin
20 et al., 2018], BEiT [Bao et al., 2021], and MAE [He et al., 2022a] for images. Along this line, there is
21 a growing interest in constructing generative SSL models for other modalities, such as graph masked
22 autoencoders (GMAE). Generally, the fundamental concept of GMAE [Tan et al., 2022] is to utilize
23 an autoencoder architecture to reconstruct input node features, structures, or both, which are randomly
24 masked before the encoding step. Recently, various well-designed GMAEs have emerged, achieving
25 remarkable results in both node classification and graph classification [Hou et al., 2022, Tu et al.,
26 2023, Tian et al., 2023].

27 Despite their significant achievements, most GMAE approaches typically treat the entire graph as
28 holistic, ignoring the graph’s latent structure. As a result, the representation learned for a node tends
29 to encapsulate the node’s neighborhood as a perceptual whole, disregarding the nuanced distinctions
30 between different parts of the neighborhood [Ma et al., 2019, Li et al., 2021, Mo et al., 2023].
31 For example, in a social network G , individual n is a member of both a mathematics group and
32 several sports interest groups. Due to the diversity of these different communities, she may exhibit
33 different characteristics when interacting with members from various communities. Specifically,
34 the information about the mathematics group may be related to her professional research, while the
35 information about sports clubs may be associated with her hobbies. However, the existing approach
36 overlooks the heterogeneous factors of node n , failing to identify and disentangle these pieces of

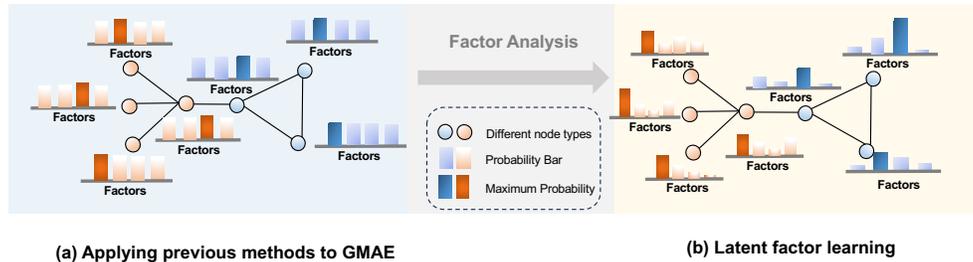


Figure 1: The number of latent factors is set to 4. In Fig. 1(a), the probabilities of nodes belonging to different latent groups are similar, resulting in nodes of the same type being incorrectly assigned to different factors. In contrast, Fig. 1(b) shows that the probabilities of node-factor affiliation are more discriminative, correctly categorizing nodes of the same type into the same latent group.

37 information effectively [Hou et al., 2022]. Consequently, the learned features may be easily influenced
 38 by irrelevant factors, resulting in poor robustness and difficulty in interpretation.

39 To alleviate the challenge described above, there is an increasing interest in disentangled graph
 40 representation learning [Bengio et al., 2013, Li et al., 2021, Ma et al., 2019, Mo et al., 2023, Xiao
 41 et al., 2022], which aims at acquiring representations that can disentangle the underlying explanatory
 42 factors of variation in the graph. Specifically, many of these methods rely on a latent factor detection
 43 module, which learns the latent factors of each node by comparing node representations with various
 44 latent factor prototypes. By leveraging these acquired latent factors, these models adeptly capture
 45 factor-wise graph representations, effectively encapsulating the latent structure of the graph. Despite
 46 significant progress, few studies have endeavored to adapt these methods to generative graph
 47 representation learning methods, such as GMAE. This primary challenge arises from the difficulty of
 48 achieving convergence in the latent factor detection module under the generative training target, thus
 49 presenting obstacles in practical implementation. As shown in Fig. 1(a), directly applying the previous
 50 factor learning method to GMAE would make the factor learning module difficult to converge,
 51 resulting in undistinguished probabilities and misallocation of similar nodes to different latent factor
 52 groups.

53 To address these challenges, we introduce **Disentangled Generative Graph Representation Learning**
 54 (**DiGGR**), a self-supervised graph generation representation learning framework. Generally speaking,
 55 DiGGR learns how to generate graph structures from latent disentangle factors z and leverages this to
 56 guide graph mask reconstruction, while enabling end-to-end joint learning. Specifically, *i*) To capture
 57 the heterogeneous factors in the nodes, we introduce the latent factor learning module. This module
 58 models how edges and nodes are generated from latent factors, allowing graphs to be factorized into
 59 multiple disentangled subgraphs. *ii*) To learn a deeper disentangled graph representation, we design a
 60 factor-wise self-supervised graph representation learning framework. For each subgraph, we employ
 61 a distinct masking strategy to learn an improved factor-specific graph representation. Evaluation
 62 shows that the proposed framework can achieve significant performance enhancement on various
 63 node and graph classification benchmarks.

64 The main contributions of this paper can be summarized as follows:

- 65 • We utilized the latent disentangled factor to guide mask modeling. A probabilistic graph
 66 generation model is employed to identify the latent factors within a graph, and it can be
 67 jointly trained with GMAE through variational inference.
- 68 • Introducing **DiGGR (Disentangled Generative Graph Representation Learning)** to further
 69 capture the disentangled information in the latent factors, enhancing the disentanglement of
 70 the learned node representations.
- 71 • Empirical results show that the proposed DiGGR outperforms many previous self-supervised
 72 methods in various node- and graph-level classification tasks.

73 2 Related works

74 **Graph Self-Supervised Learning:** Graph SSL has achieved remarkable success in addressing label
 75 scarcity in real-world network data, mainly consisting of contrastive and generative methods. Con-

76 trative methods, includes feature-oriented approaches [Hu et al., 2019, Zhu et al., 2020, Veličković
77 et al., 2018], proximity-oriented techniques [Hassani and Khasahmadi, 2020, You et al., 2020], and
78 graph-sampling-based methods [Qiu et al., 2020]. A common limitation across these approaches
79 is their heavy reliance on the design of pretext tasks and augmentation techniques. Compared to
80 contrastive methods, generative methods are generally simpler to implement. Recently, to tackle the
81 challenge of overemphasizing neighborhood information at the expense of structural information
82 [Hassani and Khasahmadi, 2020, Veličković et al., 2018], the Graph Masked Autoencoder (GMAE)
83 has been proposed. It applies a masking strategy to graph structure [Li et al., 2023a], node attributes
84 [Hou et al., 2022], or both [Tian et al., 2023] for representation learning. Unlike most GMAEs, which
85 employ random mask strategies, this paper builds disentangled mask strategies.

86 **Disentangled Graph Learning:** Disentangled representation learning aims to discover and isolate
87 the fundamental explanatory factors inherent in the data [Bengio et al., 2013]. Existing efforts in
88 disentangled representation learning have primarily focused on computer vision [Higgins et al.,
89 2017, Jiang et al., 2020]. Recently, there has been a surge of interest in applying these techniques
90 to graph-structured data [Li et al., 2021, Ma et al., 2019, Mercatali et al., 2022, Mo et al., 2023].
91 For example, DisenGCN [Ma et al., 2019] utilizes an attention-based methodology to discriminate
92 between distinct latent factors, enhancing the representation of each node to more accurately reflect
93 its features across multiple dimensions. DGCL [Li et al., 2021] suggests learning disentangled
94 graph-level representations through self-supervision, ensuring that the factorized representations
95 independently capture expressive information from various latent factors. Despite the excellent results
96 achieved by the aforementioned methods on various tasks, these methods are difficult to converge
97 in generative graph SSL, as we demonstrated in the experiment of Table.3. Therefore, this paper
98 proposes a disentangled-guided framework for generative graph representation learning, capable of
99 learning disentangled representations in an end-to-end self-supervised manner.

100 3 Proposed Method

101 In this section, we propose **DiGGR (Disentangled Generative Graph Representation Learning)** for
102 self-supervised graph representation learning with mask modeling. The framework was depicted
103 in Figure 2, comprises three primary components: Latent Factor Learning (Section 3.2), Graph
104 Factorization (Section 3.2) and Disentangled Graph Masked autoencoder (Section 3.3). Before
105 elaborating on them, we first show some notations.

106 3.1 Preliminaries

107 A graph G can be represented as a multi-tuple $\mathcal{G} = \{V, A, X\}$ with N nodes and M edges, where
108 $|V| = N$ is the node set, $|A| = M$ is the edge set, and $X \in \mathbb{R}^{N \times L}$ is the feature matrix for N
109 nodes with L dimensional feature vector. The topology structure of graph G can be found in its
110 adjacency matrix $A \in \mathbb{R}^{N \times N}$. $z \in \mathbb{R}^{N \times K}$ is the latent disentangled factor matrix, and K is the
111 predefined factor number. Since we aim to obtain the z to guide the mask modeling, we first utilize a
112 probabilistic graph generation model to factorize the graph before employing the mask mechanism.
113 Given the graph G , it is factorized into $\{G_1, G_2, \dots, G_K\}$, and each factor-specific graph G_k consists
114 of its factor-specific edges $A^{(k)}$, node set $V^{(k)}$ and node feature matrix $X^{(k)}$. Other notations will be
115 elucidated as they are employed.

116 3.2 Latent Factor Learning

117 In this subsection, we describe the latent factor learning method. In this phase, our objective is to
118 derive factor-specific node sets $\{V^{(1)}, V^{(2)}, \dots, V^{(K)}\}$ and adjacency matrices $\{A^{(1)}, A^{(2)}, \dots, A^{(K)}\}$,
119 serving as basic unit of the graph to guide the subsequent masking. The specific approach involves
120 modeling the distribution of nodes and edges, utilizing the generative process developed in EPM
121 [Zhou, 2015]. The generative process of EPM under the Bernoulli-Poisson link [Zhou, 2015] can be
122 described as:

$$123 \quad M_{uv} \sim \text{Poisson}\left(\sum_{k=1}^K \gamma_k z_{uk} z_{vk}\right), \quad z_{uk} \sim \text{Gamma}(\alpha, \beta), \quad u, v \in [1, N] \quad (1)$$

123 where K is the predefined number of latent factors, and u and v are the indexes of the nodes. Here,
124 M_{uv} is the latent count variable between node u and v ; γ_k is a positive factor activation level indicator,
125 which measures the node interaction frequency via factor k ; z_{uk} is a positive latent variable for node

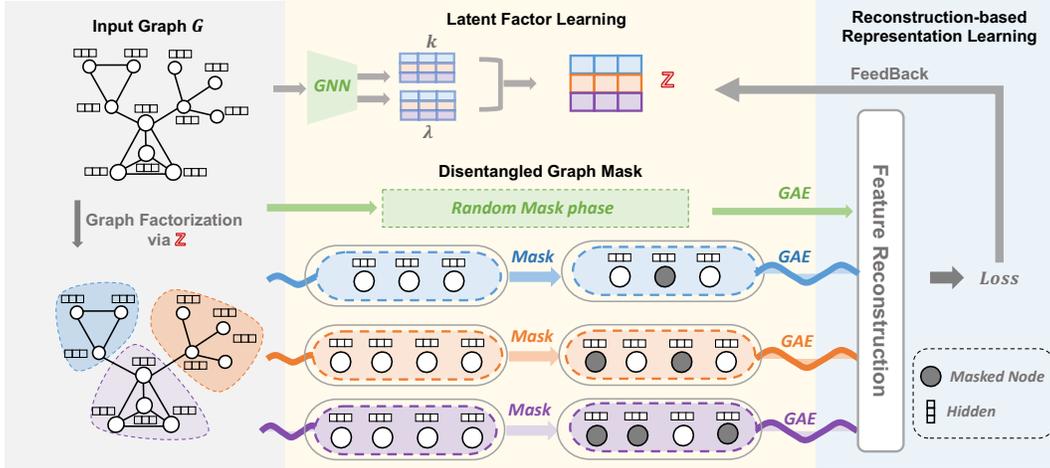


Figure 2: The overview of proposed DiGGR’s computation graph. The input data successively passes three modules described in Sections 3.2 and 3.3: Latent Factor Learning, Graph Factorization, and Disentangled Graph Mask Autoencoder. Graph information will be first processed through Latent Factor Learning and Graph Factorization, the former processed the input graph to get the latent factor z ; the latter performs graph factorization via z , such that in each factorized subgraph, nodes exchange more information with intensively interacted neighbors. Hence, during the disentangled graph masking phase, we will individually mask each factorized subgraph to enhance the disentanglement of the obtained node representations.

126 u , which measures how strongly node u is affiliated with factor k . The prior distribution of latent
 127 factor variable z_{uk} is set to Gamma distribution, where α and β are normally set to 1. Therefore, the
 128 intuitive explanation for this generative process is that, with z_{uk} and z_{vk} measuring how strongly
 129 node u and v are affiliated with the k -th factor, respectively, the product $\gamma_k z_{uk} z_{vk}$ measures how
 130 strongly nodes u and v are connected due to their affiliations with the k -th factor.

131 **Node Factorization:** Equation 1 can be further augmented as follows:

$$M_{uv} = \sum_k^K M_{ukv}, \quad M_{ukv} \sim \text{Poisson}(\gamma_k z_{uk} z_{vk}) \quad (2)$$

132 where M_{ukv} represents how often nodes u and v interact due to their affiliations with the k -th factor.
 133 To represent how often node u is affiliated with the k -th factor, we further introduce the latent count
 134 $M_{uk} = \sum_{v \neq u} M_{ukv}$. Then, we can soft assign node u to multiple factors in $\{k : M_{uk} \geq 1\}$, or
 135 hard assign node u to a single factor using $\arg \max_k (M_{uk})$. However, our experiments show that

136 soft assignment method results in significant overlap among node sets from different factor group,
 137 diminishing the distinctiveness. Note that previous study addressed a similar issue by selecting the
 138 top- k most attended regions [Kakogeorgiou et al., 2022]. Thus, we choose the hard assign strategy to
 139 factorize the graph node set V graph into factor-specific node sets $\{V^{(1)}, V^{(2)}, \dots, V^{(K)}\}$.

140 **Edge Factorization:** To create factor-specific edges $A^{(k)}$ for a factor-specific node set $V^{(k)}$, a
 141 straightforward method involves removing all external nodes connected to other factor groups. This
 142 can be defined as:

$$A_{uv}^{(k)} = \begin{cases} A_{uv}, & \forall u, v \in V^{(k)}; u, v \in [1, N]; \\ 0, & \exists u, v \notin V^{(k)}; u, v \in [1, N]. \end{cases} \quad (3)$$

143 Besides, the global graph edge A can also be factorized into positive-weighted edges [He et al.,
 144 2022b] for each latent factor as:

$$A_{uv}^{(k)} = A_{uv} \cdot \frac{\exp(\gamma_k z_{uk} z_{vk})}{\sum_{k'} \exp(\gamma_{k'} z_{uk'} z_{vk'})}; \quad k \in [1, K], u, v \in [1, N]. \quad (4)$$

145 Applying Equation 4 to all pairs of nodes yields weighted adjacency matrices $\{A^{(k)}\}_{k=1}^K$, with $A^{(k)}$
 146 corresponding to latent factor z_k . Note that $A^{(k)}$ has the same dimension as A and Equation 4

147 presents a trainable weight for each edge, which can be jointly optimized through network training,
 148 showcasing an advantage over Equation 3 in this aspect. Therefore, we apply Equation 4 for edge
 149 factorization.

150 **Variational Inference:** The latent factor variable z determines the quality of node and edge factor-
 151 ization, so we need to approximate its posterior distribution. Denoting $z_u = (z_{u1}, \dots, z_{uK})$, $z_u \in \mathbb{R}_+^K$,
 152 which measures how strongly node u is affiliated with all the K latent factors, we adopt a Weibull
 153 variational graph encoder [Zhang et al., 2018, He et al., 2022b]:

$$q(z_u | A, X) = \text{Weibull}(k_u, \lambda_u), \quad (k_u, \lambda_u) = \text{GNN}_{\text{EPM}}(A, X), \quad u \in [1, N] \quad (5)$$

154 where $\text{GNN}_{\text{EPM}}(\cdot)$ stands for graph neural networks, and we select a two-layer Graph Convolution
 155 Networks (*i.e.*, GCN [Kipf and Welling, 2016a]) for our models; $k_u, \lambda_u \in \mathbb{R}_+^K$ are the shape and
 156 scale parameters of the variational Weibull distribution, respectively. The latent variable z_u can be
 157 conveniently reparameterized as:

$$z_u = \lambda_u (-\ln(1 - \varepsilon))^{1/k_u}, \quad \varepsilon \sim \text{Uniform}(0, 1). \quad (6)$$

158 The optimization objective of latent factor learning phase can be achieved by maximizing the evidence
 159 lower bound (ELBO) of the log marginal likelihood of edge $\log p(A)$, which can be computed as:

$$\mathcal{L}_z = \mathbb{E}_{q(Z | A, X)} [\ln p(A | Z)] - \sum_{u=1}^N \mathbb{E}_{q(z_u | A, X)} \left[\ln \frac{q(z_u | A, X)}{p(z_u)} \right] \quad (7)$$

160 where the first term is the expected log-likelihood or reconstruction error of edge, and the second
 161 term is the Kullback–Leibler (KL) divergence that constrains $q(z_u)$ to be close to its prior $p(z_u)$. The
 162 analytical expression for the KL divergence and the straightforward reparameterization of the Weibull
 163 distribution simplify the gradient estimation of the ELBO concerning the decoder parameters and
 164 other parameters in the inference network.

165 3.3 Disentangled Graph Masked Autoencoder

166 With the latent factor learning phase discussed in 3.2, the graph can be factorized into a series of
 167 factor-specific subgraphs $\{G_1, G_2, \dots, G_K\}$ via the latent factor z . To incorporate the disentangled
 168 information encapsulated in z into the graph masked autoencoder, we proposed Disentangled Graph
 169 Masked Autoencoder in this section. Specifically, this section will first introduce the latent factor-wise
 170 GMAE and the graph-level GMAE.

171 3.3.1 Latent Factor-wise Graph Masked Autoencoder

172 To capture disentangled patterns within the latent factor z , for each latent subgraph
 173 $\mathcal{G}_k = (V^{(k)}, A^{(k)}, X^{(k)})$, the latent factor-wise GMAE can be described as:

$$H_d^{(k)} = \text{GNN}_{\text{enc}}(A^{(k)}, \bar{X}^{(k)}), \quad \tilde{X}^d = \text{GNN}_{\text{dec}}(A, H_d). \quad (8)$$

174 where $\bar{X}^{(k)}$ is the masked node feature matrix for the k -th latent factor, and \tilde{X}^d denotes the recon-
 175 structed node features. $\text{GNN}_{\text{enc}}(\cdot)$ and $\text{GNN}_{\text{dec}}(\cdot)$ are the graph encoder and decoder, respectively;
 176 $H_d^{(k)} \in \mathbb{R}^{N \times D}$ are factor-wise hidden representations, and $H_d = H_d^{(1)} \oplus H_d^{(2)} \dots \oplus H_d^{(K)}$. After the
 177 concatenation operation \oplus in feature dimension, the multi factor-wise hidden representation becomes
 178 $H_d \in \mathbb{R}^{N \times (K \cdot D)}$, which is used as the input of $\text{GNN}_{\text{dec}}(\cdot)$.

179 Regarding the mask operation, we uniformly random sample a subset of nodes $\bar{V}^{(k)} \in V^{(k)}$ and
 180 mask each of their features with a mask token, such as a learnable vector $X_{[M]} \in \mathbb{R}^d$. Thus, the node
 181 feature in the masked feature matrix can be defined as:

$$\bar{X}_i^{(k)} = \begin{cases} X_{[M]}; & v_i \in \bar{V}^{(k)} ; \\ X_i & ; v_i \notin \bar{V}^{(k)}. \end{cases} \quad (9)$$

182 The objective of latent factor-wise GMAE is to reconstruct the masked features of nodes in $\bar{V}^{(k)}$
 183 given the partially observed node signals $\bar{X}^{(k)}$ and the input adjacency matrix $A^{(k)}$. Another crucial
 184 component of the GMAE is the feature reconstruction criterion, often used in language as cross-
 185 entropy error [Devlin et al., 2018] and in the image as mean square error [He et al., 2022a]. However,

186 texts and images typically involve tokenized input features, whereas graph autoencoders (GAE) do
 187 not have a universal tokenizer. We adopt the scored cosine error of GraphMAE [Hou et al., 2022] as
 188 the loss function. Generally, given the original feature $X^{(k)}$ and reconstructed node feature $\tilde{X}^{(k)}$, the
 189 defined SCE is:

$$\mathcal{L}_D = \frac{1}{|\tilde{V}|} \sum_{i \in \tilde{V}} \left(1 - \frac{X_i^T \tilde{X}_i^d}{\|X_i\| \cdot \|\tilde{X}_i^d\|} \right)^\gamma, \quad \gamma \geq 1 \quad (10)$$

190 where $\tilde{V} = \tilde{V}^{(1)} \cup \tilde{V}^{(2)} \dots \cup \tilde{V}^{(K)}$ and Equation 10 are averaged over all masked nodes. The scaling
 191 factor γ is a hyper-parameter adjustable over different datasets. This scaling technique could also
 192 be viewed as adaptive sample reweighting, and the weight of each sample is adjusted with the
 193 reconstruction error. This error is also famous in the field of supervised object detection as the focal
 194 loss [Lin et al., 2017].

195 **Graph-level Graph Mask Autoencoder:** For the node classification task, we have integrated
 196 graph-level GMAE into DiGGR. We provide a detailed experimental analysis and explanation for
 197 this difference in Appendix A.1.2. The graph-level masked graph autoencoder is designed with the
 198 aim of further capturing the global patterns, which can be designed as:

$$H_g = \text{GNN}_{\text{enc}}(A, \bar{X}), \quad \tilde{X}^g = \text{GNN}_{\text{dec}}(A, H_g). \quad (11)$$

199 \bar{X} is the masked node feature matrix, whose mask can be generated by uniformly random sampling
 200 a subset of nodes $\tilde{V} \in V$, or obtained by concatenating the masks of all factor-specific groups
 201 $\tilde{V} = \tilde{V}^{(1)} \cup \tilde{V}^{(2)} \dots \cup \tilde{V}^{(K)}$. The global hidden representation encoded by $\text{GNN}_{\text{enc}}(\cdot)$ is H_g , which is
 202 then passed to the decoder. Similar to Equation 10, we can define the graph-level reconstruct loss as:

$$\mathcal{L}_G = \frac{1}{|\tilde{V}|} \sum_{i \in \tilde{V}} \left(1 - \frac{X_i^T \tilde{X}_i^g}{\|X_i\| \cdot \|\tilde{X}_i^g\|} \right)^\gamma, \quad \gamma \geq 1. \quad (12)$$

203 which is averaged over all masked nodes.

204 3.4 Joint Training and Inference

205 Benefiting from the effective variational inference method, the proposed latent factor learning and
 206 dsientangled graph masked autoencoder can be jointly trained in one framework. We combine the
 207 aforementioned losses with three mixing coefficient λ_d , λ_g and λ_z during training, and the loss for
 208 joint training can be written as

$$\mathcal{L} = \lambda_d \cdot \mathcal{L}_D + \lambda_g \cdot \mathcal{L}_G + \lambda_z \cdot \mathcal{L}_z. \quad (13)$$

209 Since Weibull distributions have easy reparameterization functions, these parameters can be jointly
 210 trained by stochastic gradient descent with low-variance gradient estimation. We summarize the
 211 training algorithm at Algorithm 1 in Appendix A.4. For downstream applications, the encoder is
 212 applied to the input graph without any masking in the inference stage. The generated factor-wise
 213 node embeddings H_d and graph-level embeddings H_g can either be concatenated in the feature
 214 dimensions or used separately. The resulting final representation H can be employed for various
 215 graph learning tasks, such as node classification and graph classification. For graph-level tasks, we
 216 use a non-parameterized graph pooling (readout) function, *e.g.*, MaxPooling and MeanPooling to
 217 obtain the graph-level representation.

218 **Time and space complexity:** Let’s recall that in our context, N , M , and K represent the number of
 219 nodes, edges, and latent factors in the graph, respectively. The feature dimension is denoted by F ,
 220 while L_1 , L_2 , L_3 , and L_4 represent the number of layers in the latent factor learning encoder, the
 221 latent factor-wise GMAE’s encoder, the graph-level GMAE’s encoder, and the decoder respectively.
 222 In DiGGR, we constrain the hidden dimension size in latent factor-wise GMAE’s encoder to be
 223 $1/K$ of the typical baseline dimensions. Consequently, the time complexity for training DiGGR can
 224 be expressed as $O((L_1 + L_2 + L_3)MF + (L_1 + L_2/K + L_3)NF^2 + N^2F + L_4NF^2)$, and the
 225 space complexity is $O((L_1 + L_2 + L_3 + L_4)NF + KM + (L_1 + L_2/K + L_3 + L_4)F^2)$, with
 226 $O((L_1 + L_2/K + L_3 + L_4)F^2)$ attributed to model parameters. We utilize the Bayesian factor model
 227 in our approach to reconstruct edges. Its time complexity aligns with that of variational inference
 228 in SeeGera Li et al. [2023b], predominantly at $O(N^2F)$; Therefore, the complexity of DiGGR is
 229 comparable to previous works.

Table 1: Experiment results for node classification. Micro-F1 score is reported for PPI, and accuracy for other datasets. The best unsupervised method scores in each dataset are highlighted in bold.

Methods	Cora	Citeseer	Pubmed	PPI
GCN [Kipf and Welling, 2016a]	81.50	70.30	79.00	75.70 ± 0.10
GAT [Velickovic et al., 2017]	83.00 ± 0.70	72.50 ± 0.70	79.00 ± 0.30	97.30 ± 0.20
DisenGCN[Ma et al., 2019]	83.7	73.4	80.5	-
VEPM[He et al., 2022b]	84.3 ± 0.1	72.5 ± 0.1	82.4 ± 0.2	-
MVGRL [Hassani and Khasahmadi, 2020]	83.50 ± 0.40	73.30 ± 0.50	80.10 ± 0.70	-
InfoGCL [Xu et al., 2021]	83.50 ± 0.30	73.50 ± 0.40	79.10 ± 0.20	-
DGI [Veličković et al., 2018]	82.30 ± 0.60	71.80 ± 0.70	76.80 ± 0.60	63.80 ± 0.20
GRACE [Zhu et al., 2020]	81.90 ± 0.40	71.20 ± 0.50	80.60 ± 0.40	69.71 ± 0.17
BGRL [Thakoor et al., 2021]	82.70 ± 0.60	71.10 ± 0.80	79.60 ± 0.50	73.63 ± 0.16
CCA-SSG [Zhang et al., 2021]	84.20 ± 0.40	73.10 ± 0.30	81.00 ± 0.40	73.34 ± 0.17
GAE [Kipf and Welling, 2016b]	71.50 ± 0.40	65.80 ± 0.40	72.10 ± 0.50	-
VGAE [Kipf and Welling, 2016b]	76.30 ± 0.20	66.80 ± 0.20	75.80 ± 0.40	-
Bandana [Zhao et al., 2024]	84.62 ± 0.37	73.60 ± 0.16	83.53 ± 0.51	-
GiGaMAE[Shi et al., 2023]	84.72 ± 0.47	72.31 ± 0.50	-	-
SEEGERA [Shi et al., 2023]	84.30 ± 0.40	73.00 ± 0.80	80.40 ± 0.40	-
GraphMAE [Hou et al., 2022]	84.20 ± 0.40	73.40 ± 0.40	81.10 ± 0.40	74.50 ± 0.29
GraphMAE2[Hou et al., 2023]	84.50 ± 0.60	73.40 ± 0.30	81.40 ± 0.50	-
DiGGR	84.96 ± 0.32	73.98 ± 0.27	81.30 ± 0.26	78.30 ± 0.71

230 4 Experiments

231 We compare the proposed self-supervised framework DiGGR against related baselines on two funda-
 232 mental tasks: unsupervised representation learning on *node classification* and *graph classification*.
 233 We evaluate DiGGR on 11 benchmarks. For node classification, we use 3 citation networks (Cora,
 234 Citeseer, Pubmed [Yang et al., 2016]), and protein-protein interaction networks (PPI) [Hamilton et al.,
 235 2017]. For graph classification, we use 3 bioinformatics datasets (MUTAG, NCI1, PROTEINS) and 4
 236 social network datasets (IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY and COLLAB). The
 237 specific information of the dataset and the hyperparameters used by the network are listed in the
 238 Appendix A.2 in table 5 and 6. We also provide the detailed experiment setup in Appendix A.2 for
 239 node classification (4.1) and graph classification (4.2)

240 4.1 Node Classification

241 The baseline models for node classification can be divided into three categories: *i*) supervised methods,
 242 including GCN [Kipf and Welling, 2016a], DisenGCN[Ma et al., 2019], VEPM[He et al., 2022b]
 243 and GAT [Velickovic et al., 2017]; *ii*) contrastive learning methods, including MVGRL [Hassani and
 244 Khasahmadi, 2020], InfoGCL [Xu et al., 2021], DGI [Veličković et al., 2018], GRACE [Zhu et al.,
 245 2020], BGRL [Thakoor et al., 2021] and CCA-SSG [Zhang et al., 2021]; *iii*) generative learning
 246 methods, including GraphMAE [Hou et al., 2022], GraphMAE2[Hou et al., 2023], Bandana[Zhao
 247 et al., 2024], GiGaMAE[Shi et al., 2023], SeeGera[Li et al., 2023b], GAE and VGAE [Kipf and
 248 Welling, 2016b]. The node classification results were listed in Table 1. DiGGR demonstrates
 249 competitive results on the provided dataset, achieving results comparable to those of supervised
 250 methods.

251 4.2 Graph Classification

252 **Baseline Models** We categorized the baseline models into four groups: *i*) supervised methods,
 253 including GIN [Xu et al., 2018], DiffPool[Ying et al., 2018] and VEPM[He et al., 2022b]; *ii*) classical
 254 graph kernel methods: Weisfeiler-Lehman sub-tree kernel (WL) [Shervashidze et al., 2011] and
 255 deep graph kernel (DGK) [Yanardag and Vishwanathan, 2015]; *iii*) contrastive learning methods,
 256 including GCC [Qiu et al., 2020], graph2vec [Narayanan et al., 2017], Infograph [Sun et al., 2019],
 257 GraphCL [You et al., 2020], JOAO [You et al., 2021], MVGRL [Hassani and Khasahmadi, 2020],
 258 and InfoGCL [Xu et al., 2021]; 4) generative learning methods, including graph2vec [Narayanan
 259 et al., 2017], sub2vec [Adhikari et al., 2018], node2vec [Grover and Leskovec, 2016], GraphMAE
 260 [Hou et al., 2022], GraphMAE2[Hou et al., 2023], GAE and VGAE [Kipf and Welling, 2016b]. Per
 261 graph classification research tradition, we report results from previous papers if available.

Table 2: Experiment results in unsupervised representation learning for graph classification. We report accuracy (%) for all datasets. The optimal outcomes for methods, excluding supervised approaches (GIN and DiffPool), on each dataset are emphasized in bold.

Methods	IMDB-B	IMDB-M	MUTAG	NCII	REDDIT-B	PROTEINS	COLLAB
GIN	75.1 ± 5.1	52.3 ± 2.8	89.4 ± 5.6	82.7 ± 1.7	92.4 ± 2.5	76.2 ± 2.8	80.2 ± 1.9
DiffPool	72.6 ± 3.9	-	85.0 ± 10.3	-	92.1 ± 2.6	75.1 ± 3.5	78.9 ± 2.3
VEPM	76.7 ± 3.1	54.1 ± 2.1	93.6 ± 3.4	83.9 ± 1.8	90.5 ± 1.8	80.5 ± 2.8	-
WL	72.30 ± 3.44	46.95 ± 0.46	80.72 ± 3.00	80.31 ± 0.46	68.82 ± 0.41	72.92 ± 0.56	-
DGK	66.96 ± 0.56	44.55 ± 0.52	87.44 ± 2.72	80.31 ± 0.46	78.04 ± 0.39	73.30 ± 0.82	73.09 ± 0.25
Infograph	73.03 ± 0.87	49.69 ± 0.53	89.01 ± 1.13	76.20 ± 1.06	82.50 ± 1.42	74.44 ± 0.31	70.65 ± 1.13
GraphCL	71.14 ± 0.44	48.58 ± 0.67	86.80 ± 1.34	77.87 ± 0.41	89.53 ± 0.84	74.39 ± 0.45	71.36 ± 1.15
JOAO	70.21 ± 3.08	49.20 ± 0.77	87.35 ± 1.02	78.07 ± 0.47	85.29 ± 1.35	74.55 ± 0.41	69.50 ± 0.36
GCC	72.0	49.4	-	-	89.9	-	78.9
MVGRL	74.20 ± 0.70	51.20 ± 0.50	89.70 ± 1.10	-	84.50 ± 0.60	-	-
InfoGCL	75.10 ± 0.90	51.40 ± 0.80	91.20 ± 1.30	80.20 ± 0.60	-	-	80.00 ± 1.30
graph2vec	71.10 ± 0.54	50.44 ± 0.87	83.15 ± 9.25	73.22 ± 1.81	75.78 ± 1.03	73.30 ± 2.05	-
sub2vec	55.3 ± 1.5	36.7 ± 0.8	61.1 ± 15.8	52.8 ± 1.5	71.5 ± 0.4	53.0 ± 5.6	-
node2vec	-	-	72.6 ± 10.2	54.9 ± 1.6	-	57.5 ± 3.6	-
GAE	52.1 ± 0.2	-	84.0 ± 0.6	73.3 ± 0.6	74.8 ± 0.2	74.1 ± 0.5	-
VGAE	52.1 ± 0.2	-	84.4 ± 0.6	73.7 ± 0.3	74.8 ± 0.2	74.8 ± 0.2	-
GraphMAE	75.52 ± 0.66	51.63 ± 0.52	88.19 ± 1.26	80.40 ± 0.30	88.01 ± 0.19	75.30 ± 0.39	80.32 ± 0.46
GraphMAE2	73.88 ± 0.53	51.80 ± 0.60	86.63 ± 1.33	78.56 ± 0.26	76.84 ± 0.21	74.86 ± 0.34	77.59 ± 0.22
DiGGR	77.68 ± 0.48	54.77 ± 2.63	88.72 ± 1.03	81.23 ± 0.40	88.19 ± 0.28	77.40 ± 0.05	83.76 ± 3.70

262 **Performance Comparison** The graph classification results are presented in Table 2. In general, we
 263 find that DiGGR gained the best performance among other baselines on five out of seven datasets,
 264 while achieving competitive results on the other two datasets. The performance of DiGGR is
 265 comparable to that of supervised learning methods. For instance, the accuracy on IMDB-B and
 266 IMDB-M surpasses that of GIN and DiffPool. Moreover, within the reported datasets, our method
 267 demonstrates improved performance compared to random mask methods like GraphMAE, particularly
 268 on the IMDB-M, COLLAB, and PROTEINS datasets. This underscores the effectiveness of the
 269 proposed method.

270 4.3 Exploratory Studies

271 **Visualizing latent representations** To examine the influence of the learned latent factor on classifi-
 272 cation results, we visualized the latent disentangled factor z , which reflects the node-factor affiliation,
 273 and the hidden representation H used for classification. MUTAG is selected as the representative
 274 for classification benchmarks. We encode the representations into 2-D space via t-SNE [Van der
 275 Maaten and Hinton, 2008]. The result is shown in Figure 3(a), where each node is colored according
 276 to its node labels. The clusters in Figure 3(a) still exhibit differentiation in the absence of label
 277 supervision, suggesting that z obtained through unsupervised learning can enhance node information
 278 and offer a guidance for the mask modeling. We then visualize the hidden representation used for
 279 classification tasks, and color each node according to the latent factor to which it belongs. The results
 280 are depicted in Figure 3(b), showcasing separability among different color clusters. This illustrates
 281 the model’s ability to extract information from the latent factor, thereby enhancing the quality of the
 282 learned representations.

283 **Task-relevant factors** To assess the statistical correlation between the learned latent factor and the
 284 task, we follow the approach in [He et al., 2022b] and compute the Normalized Mutual Information
 285 (NMI) between the nodes in the factor label and the actual node labels. NMI is a metric that ranges
 286 from 0 to 1, where higher values signify more robust statistical dependencies between two random
 287 variables. In the experiment, we utilized the MUTAG dataset, comprising 7 distinct node types,
 288 and the NMI value we obtained was 0.5458. These results highlight that the latent factors obtained
 289 through self-supervised training are meaningful for the task, enhancing the correlation between the
 290 inferred latent factors and the task.

291 **Disentangled representations** To assess DiGGR’s capability to disentangle the learned represen-
 292 tation for downstream task, we provide a qualitative evaluation by plotting the correlation of the

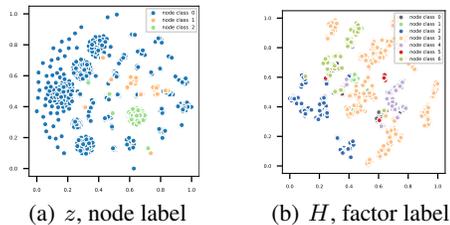


Figure 3: T-SNE visualization of MUTAG dataset, where z is the latent factor, H is the learned node representation used for downstream tasks.

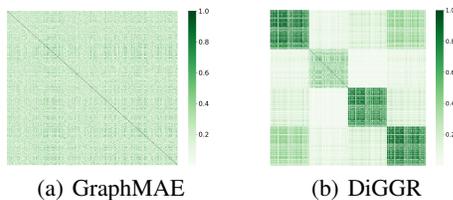


Figure 4: representation correlation matrix on Cora with number of factors $K = 4$. 4(a) depicts the representation of entanglement, while 4(b) illustrates disentanglement.

Table 3: The NMI between the latent factors extracted by DiGGR and Non-probabilistic factor learning method across various datasets, and its performance improvement compared to GraphMAE, are examined. A lower NMI indicates a more pronounced disentanglement between factor-specific graphs, resulting in a greater performance enhancement.

	Dataset	RDT-B	MUTAG	NCI-1	IMDB-B	PROTEINS	COLLAB	IMDB-M
DiGGR	NMI	0.95	0.90	0.89	0.82	0.76	0.35	0.24
	ACC Gain	+ 0.18%	+ 0.53%	+ 0.83%	+ 2.16%	+ 2.1%	+ 3.44%	+ 3.14%
Non-probabilistic Factor Learning	NMI	1.00	1.00	0.80	1.00	0.60	1.00	0.94
	ACC Gain	-2.23%	-2.02%	-0.45%	-0.80%	-2.15%	-3.00%	-0.11%

293 node representation in Figure 4. The figure shows the absolute values of the correlation between the
 294 elements of 512-dimensional graph representation and representation obtained from GraphMAE and
 295 DiGGR, respectively. From the results, we can see that the representation produced by GraphMAE
 296 exhibits entanglement, whereas DiGGR’s representation displays a overall block-level pattern,
 297 indicating that DiGGR can capture mutually exclusive information in the graph and disentangle the
 298 hidden representation to some extent. Results for more datasets can be found in Appendix A.3.

299 **Why DiGGR works better:** To validate that disentangled learning can indeed enhance the quality of
 300 the representations learned by GMAE, we further conduct quantitative experiments. The Normalized
 301 Mutual Information (NMI) is used to quantify the disentangling degree of different datasets. Generally,
 302 the NMI represents the similarity of node sets between different factor-specific graphs, and the *lower*
 303 *NMI suggests a better-disentangled degree* with lower similarity among factor-specific graphs. The
 304 NMI between latent factors and the corresponding performance gain (compared to GraphMAE)
 305 are shown in the Table.3. As the results show, DiGGR’s performance improvement has a positive
 306 correlation with disentangled degree, where the better the disentangled degree, the more significant
 307 the performance improvement. For methods relying on Non-probabilistic Factor Learning, the NMI
 308 tends to approach 1. This is attributed to the challenges faced by the factor learning module in
 309 converging, thereby hindering the learning of distinct latent factors. The presence of confused latent
 310 factors offers misleading guidance for representation learning, consequently leading to decreased
 311 performance.
 312

313 5 Conclusions

314 In this paper, we propose DiGGR (Disentangled Generative Graph Representation Learning), de-
 315 signed to achieve disentangled representations in graph masked autoencoders by leveraging latent
 316 disentangled factors. In particular, we achieve this by two steps: 1) We utilize a probabilistic graph
 317 generation model to factorize the graph via the learned disentangled latent factor; 2) We develop a
 318 Disentangled Graph Masked Autoencoder framework, with the aim of integrating the disentangled in-
 319 formation into the representation learning of Graph Masked Autoencoders. Experiments demonstrate
 320 that our model can acquire disentangled representations, and achieve favorable results on downstream
 321 tasks.

322 References

- 323 Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for
324 contrastive learning of visual representations. In *International conference on machine learning*,
325 pages 1597–1607. PMLR, 2020.
- 326 Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph
327 contrastive learning with augmentations. *Advances in neural information processing systems*, 33:
328 5812–5823, 2020.
- 329 Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep
330 bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- 331 Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers.
332 *arXiv preprint arXiv:2106.08254*, 2021.
- 333 Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked
334 autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer
335 vision and pattern recognition*, pages 16000–16009, 2022a.
- 336 Qiaoyu Tan, Ninghao Liu, Xiao Huang, Rui Chen, Soo-Hyun Choi, and Xia Hu. Mgae: Masked
337 autoencoders for self-supervised learning on graphs. *arXiv preprint arXiv:2201.02534*, 2022.
- 338 Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang.
339 GraphMAE: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM
340 SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 594–604, 2022.
- 341 Wenxuan Tu, Qing Liao, Sihang Zhou, Xin Peng, Chuan Ma, Zhe Liu, Xinwang Liu, and Zhiping
342 Cai. Rare: Robust masked graph autoencoder. *arXiv preprint arXiv:2304.01507*, 2023.
- 343 Yijun Tian, Kaiwen Dong, Chunhui Zhang, Chuxu Zhang, and Nitesh V Chawla. Heterogeneous
344 graph masked autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*,
345 volume 37, pages 9997–10005, 2023.
- 346 Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. Disentangled graph convolutional
347 networks. In *International conference on machine learning*, pages 4212–4221. PMLR, 2019.
- 348 Haoyang Li, Xin Wang, Ziwei Zhang, Zehuan Yuan, Hang Li, and Wenwu Zhu. Disentangled
349 contrastive learning on graphs. *Advances in Neural Information Processing Systems*, 34:21872–
350 21884, 2021.
- 351 Yujie Mo, Yajie Lei, Jialie Shen, Xiaoshuang Shi, Heng Tao Shen, and Xiaofeng Zhu. Disentangled
352 multiplex graph representation learning. In *International Conference on Machine Learning*, pages
353 24983–25005. PMLR, 2023.
- 354 Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new
355 perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828,
356 2013.
- 357 Teng Xiao, Zhengyu Chen, Zhimeng Guo, Zeyang Zhuang, and Suhang Wang. Decoupled self-
358 supervised learning for graphs. *Advances in Neural Information Processing Systems*, 35:620–634,
359 2022.
- 360 Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec.
361 Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.
- 362 Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive
363 representation learning. *arXiv preprint arXiv:2006.04131*, 2020.
- 364 Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon
365 Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.
- 366 Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on
367 graphs. In *International conference on machine learning*, pages 4116–4126. PMLR, 2020.

- 368 Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang,
369 and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings*
370 *of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages
371 1150–1160, 2020.
- 372 Jintang Li, Ruofan Wu, Wangbin Sun, Liang Chen, Sheng Tian, Liang Zhu, Changhua Meng, Zibin
373 Zheng, and Weiqiang Wang. What’s behind the mask: Understanding masked graph modeling
374 for graph autoencoders. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge*
375 *Discovery and Data Mining*, pages 1268–1279, 2023a.
- 376 Irina Higgins, Loic Matthey, Arka Pal, Christopher P Burgess, Xavier Glorot, Matthew M Botvinick,
377 Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a
378 constrained variational framework. *ICLR (Poster)*, 3, 2017.
- 379 Wentao Jiang, Si Liu, Chen Gao, Jie Cao, Ran He, Jiashi Feng, and Shuicheng Yan. Psgan: Pose
380 and expression robust spatial-aware gan for customizable makeup transfer. In *Proceedings of the*
381 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5194–5202, 2020.
- 382 Giangiacomo Mercatali, André Freitas, and Vikas Garg. Symmetry-induced disentanglement on
383 graphs. *Advances in neural information processing systems*, 35:31497–31511, 2022.
- 384 Mingyuan Zhou. Infinite edge partition models for overlapping community detection and link
385 prediction. In *Artificial intelligence and statistics*, pages 1135–1143. PMLR, 2015.
- 386 Ioannis Kakogeorgiou, Spyros Gidaris, Bill Psomas, Yannis Avrithis, Andrei Bursuc, Konstantinos
387 Karantzas, and Nikos Komodakis. What to hide from your students: Attention-guided masked
388 image modeling. In *European Conference on Computer Vision*, pages 300–318. Springer, 2022.
- 389 Yilin He, Chaojie Wang, Hao Zhang, Bo Chen, and Mingyuan Zhou. A variational edge partition
390 model for supervised graph representation learning. *Advances in Neural Information Processing*
391 *Systems*, 35:12339–12351, 2022b.
- 392 Hao Zhang, Bo Chen, Dandan Guo, and Mingyuan Zhou. Whai: Weibull hybrid autoencoding
393 inference for deep topic modeling. *arXiv preprint arXiv:1803.01328*, 2018.
- 394 Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks.
395 *arXiv preprint arXiv:1609.02907*, 2016a.
- 396 Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense
397 object detection. In *Proceedings of the IEEE international conference on computer vision*, pages
398 2980–2988, 2017.
- 399 Xiang Li, Tiandi Ye, Caihua Shan, Dongsheng Li, and Ming Gao. Seegera: Self-supervised semi-
400 implicit graph variational auto-encoders with masking. In *Proceedings of the ACM web conference*
401 *2023*, pages 143–153, 2023b.
- 402 Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with
403 graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.
- 404 Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs.
405 *Advances in neural information processing systems*, 30, 2017.
- 406 Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio,
407 et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.
- 408 Dongkuan Xu, Wei Cheng, Dongsheng Luo, Haifeng Chen, and Xiang Zhang. Infogcl: Information-
409 aware graph contrastive learning. *Advances in Neural Information Processing Systems*, 34:
410 30414–30425, 2021.
- 411 Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L Dyer, Remi
412 Munos, Petar Veličković, and Michal Valko. Large-scale representation learning on graphs via
413 bootstrapping. *arXiv preprint arXiv:2102.06514*, 2021.

- 414 Hengrui Zhang, Qitian Wu, Junchi Yan, David Wipf, and Philip S Yu. From canonical correlation
415 analysis to self-supervised graph neural networks. *Advances in Neural Information Processing*
416 *Systems*, 34:76–89, 2021.
- 417 Zhenyu Hou, Yufei He, Yukuo Cen, Xiao Liu, Yuxiao Dong, Evgeny Kharlamov, and Jie Tang.
418 Graphmae2: A decoding-enhanced masked self-supervised graph learner. In *Proceedings of the*
419 *ACM Web Conference 2023*, pages 737–746, 2023.
- 420 Ziwen Zhao, Yuhua Li, Yixiong Zou, Jiliang Tang, and Ruixuan Li. Masked graph autoencoder with
421 non-discrete bandwidths. *arXiv preprint arXiv:2402.03814*, 2024.
- 422 Yucheng Shi, Yushun Dong, Qiaoyu Tan, Jundong Li, and Ninghao Liu. Gigamae: Generalizable
423 graph masked autoencoder via collaborative latent space reconstruction. In *Proceedings of the 32nd*
424 *ACM International Conference on Information and Knowledge Management*, pages 2259–2269,
425 2023.
- 426 Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*,
427 2016b.
- 428 Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural
429 networks? *arXiv preprint arXiv:1810.00826*, 2018.
- 430 Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hier-
431 archical graph representation learning with differentiable pooling. *Advances in neural information*
432 *processing systems*, 31, 2018.
- 433 Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M
434 Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.
- 435 Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM*
436 *SIGKDD international conference on knowledge discovery and data mining*, pages 1365–1374,
437 2015.
- 438 Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu,
439 and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv preprint*
440 *arXiv:1707.05005*, 2017.
- 441 Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-
442 supervised graph-level representation learning via mutual information maximization. *arXiv preprint*
443 *arXiv:1908.01000*, 2019.
- 444 Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. Graph contrastive learning automated.
445 In *International Conference on Machine Learning*, pages 12121–12132. PMLR, 2021.
- 446 Bijaya Adhikari, Yao Zhang, Naren Ramakrishnan, and B Aditya Prakash. Sub2vec: Feature
447 learning for subgraphs. In *Advances in Knowledge Discovery and Data Mining: 22nd Pacific-Asia*
448 *Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3-6, 2018, Proceedings, Part II 22*,
449 pages 170–182. Springer, 2018.
- 450 Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings*
451 *of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*,
452 pages 855–864, 2016.
- 453 Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine*
454 *learning research*, 9(11), 2008.
- 455 Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM*
456 *transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.
- 457 Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors.
458 In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages
459 475–486. IEEE, 2006.

460 **A Appendix / supplemental material**

461 Optionally include supplemental material (complete proofs, additional experiments and plots) in
 462 appendix. All such materials **SHOULD be included in the main submission.**

463 **A.1 Ablation Study**

464 **A.1.1 Number of factors**

465 One of the crucial hyperparameters in DiGGR is the *number of latent factors*, denoted as K . When
 466 $K = 1$ DiGGR degenerates into ordinary GMAE, only performing random masking over the entire
 467 input graph on the nodes. The influence of tuning K is illustrated in Figure 5. Given the relatively
 468 small size of the graphs in the dataset, the number of meaningful latent disentangled factor z is
 469 not expected to be very large. The optimal number of z that maximizes performance tends to be
 470 concentrated in the range of 2-4.

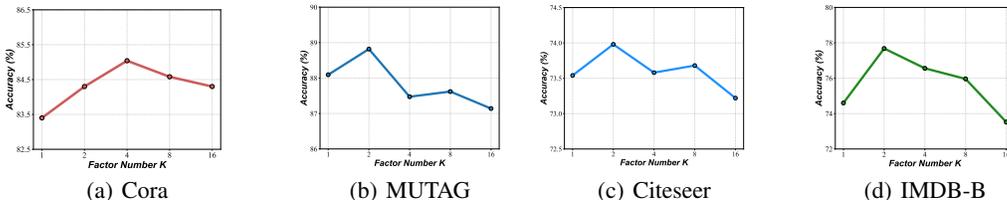


Figure 5: Performance of the task under different choices of latent factor number K , where the horizontal axis represents the change in K and the vertical axis is accuracy.

471 **A.1.2 Representation for downstream tasks**

472 We investigate the impact of various combinations of representation levels on downstream tasks. As
 473 illustrated in Table 4, for the node classification task, both H_d and H_g are required, *i.e.*, concatenating
 474 them in feature dimension, whereas for the graph classification task, H_d alone is sufficient. This
 475 difference may be due to the former not utilizing pooling operations, while the latter does. Specifically,
 476 the graph pooling operation aggregates information from all nodes, providing a comprehensive
 477 view of the entire graph structure. Thus, in node classification, where the node representation has
 478 not undergone pooling, a graph-level representation (H_g) is more critical. In contrast, in graph
 479 classification, the node representation undergoes pooling, making disentangled information H_d more
 effective.

Table 4: The average accuracy of datasets is calculated through 5 random initialization tests when using different representations.

H_d	H_g	Cora	IMDB-MULTI	Citeseer	PROTEINS
✓		61.10 ± 1.83	54.77 ± 2.63	71.82 ± 0.98	77.76 ± 2.46
	✓	84.22 ± 0.38	51.62 ± 0.61	73.41 ± 0.43	75.52 ± 0.49
✓	✓	84.96 ± 0.32	53.69 ± 2.06	73.98 ± 0.27	77.61 ± 0.97

480

481 **A.2 Implementation Details**

482 **Environment** All experiments are conducted on Linux servers equipped with an 12th Gen Intel(R)
 483 Core(TM) i7-12700, 256GB RAM and a NVIDIA 3090 GPU. Models of node and graph classification
 484 are implemented in PyTorch version 1.12.1, scikit-learn version 1.0.2 and Python 3.7.

485 **Experiment Setup for Node Classification** The node classification task involves predicting the
 486 unknown node labels in networks. Cora, Citeseer, and Pubmed are employed for transductive learning,
 487 whereas PPI follows the inductive setup outlined in GraphSage [Hamilton et al., 2017]. For evaluation,

Table 5: Statistics for node classification datasets.

	Dataset	Cora	Citeseer	Pubmed	PPI
Statistics	# node	2708	3327	19717	56944
	# feature	1433	3703	500	50
	# edges	5429	4732	44338	818736
	# classes	7(s)	6(s)	3(s)	121(m)
Hyper-parameters	Mask Rate	0.5	0.5	0.75	0.5
	Hidden Size	512	512	1024	1024
	Max Epoch	1750	200	1000	1000
	$\lambda_d; \lambda_g; \lambda_z$	1; 1; 1	1; 1; 2	1; 1; 1	1; 1; 1
	Learning Rate	0.001	0.0005	0.001	0.0001
	Factor_Num	4	4	2	2

488 we use the concatenated representations of H_d and H_g in the feature dimension for the downstream
489 task. We then train a linear classifier, report the mean accuracy on the test nodes through 5 random
490 initializations. The graph encoder $\text{GNN}_{\text{enc}(\cdot)}$ and decoder $\text{GNN}_{\text{dec}(\cdot)}$ are both specified as standard
491 GAT [Velickovic et al., 2017]. We train the model using Adam Optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$,
492 $\epsilon = 1 \times 10^8$, and we use the cosine learning rate decay without warmup. We follow the public data
493 splits of Cora, Citeseer, and PubMed.

494 **Experiment Setup for Graph Classification** The graph classification experiment was conducted on
495 7 benchmarks, in which node labels are used as input features in MUTAG, PROTEINS and NCI1, and
496 node degrees are used in IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY, and COLLAB. The
497 backbone of encoder and decoder is GIN [Xu et al., 2018], which is commonly used in previous graph
498 classification works. The evaluation protocol primarily follows GraphMAE [Hou et al., 2022]. Notice
499 that we only utilize the factor-wise latent representation H_d for the downstream task. Subsequently,
500 we feed it into a downstream LIBSVM [Chang and Lin, 2011] classifier to predict the label and
501 report the mean 10-fold cross-validation accuracy with standard deviation after 5 runs. We set the
502 initial learning rate to 0.0005 with cosine learning rate decay for most cases. For the evaluation, the
503 parameter C of SVM is searched in the sets $\{10^3, \dots, 10\}$.

504 **Data Preparation** The node features for the citation networks (Cora, Citeseer, Pubmed) are bag-of-
505 words document representations. For the protein-protein interaction networks (PPI), the features of
506 each node are composed of positional gene sets, motif gene sets and immunological signatures (50 in
507 total). For graph classification, the MUTAG, PROTEINS, and NCI1 datasets utilize node labels as
508 node features, represented in the form of one-hot encoding. For IMDB-B, IMDB-M, REDDIT-B, and
509 COLLAB, which lack node features, we utilize the node degree and convert it into a one-hot encoding
510 as a substitute feature. The maximum node degree is set to 400. Nodes with degrees surpassing 400
511 are uniformly treated as having a degree of 400, following the methodology of GraphMAE[Hou et al.,
512 2022]. Table 5 and Table 6 show the specific statistics of used datasets.

513 **Details for Visualization** MUTAG is selected as the representative benchmark for visualization
514 in 4.3. The MUTAG dataset comprises 3,371 nodes with seven node types. The distribution is
515 highly skewed, as 3,333 nodes belong to three types, while the remaining four types collectively
516 represent less than 1.2% of the nodes. For clarity in legend display, we have visualized only the nodes
517 belonging to the first three types.

518 A.3 Disentangled Representations Visualization

519 We chose PROTEINS and IMDB-MULTI as representatives of the graph classification dataset, and
520 followed the same methodology as in Section 4.3 to visualize their representation correlation matrices
521 on GraphMAE, and community representation correlation matrices on DiGGR, respectively. The
522 feature dimensions of PROTEINS and IMDB-MULTI are both 512 dimensions, and the number of
523 communities is set to 4.

Table 6: Statistics for graph classification datasets.

	Dataset	IMDB-B	IMDB-M	PROTEINS	COLLAB	MUTAG	REDDIT-B	NCII
Statistics	Avg. # node	19.8	13.0	39.1	74.5	17.9	429.7	29.8
	# features	136	89	3	401	7	401	37
	# graphs	1000	1500	1113	5000	188	2000	4110
	# classes	2	3	2	3	2	2	2
Hyper-parameters	Mask Rate	0.5	0.5	0.5	0.75	0.75	0.75	0.25
	Hidden Size	512	512	512	256	32	512	1024
	Max Epoch	300	200	50	20	20	200	200
	Learning Rate	0.0001	0.001	0.0005	0.001	0.001	0.0005	0.0005
	$\lambda_d; \lambda_g; \lambda_z$	1; 1; 1	1; 1; 1	1; 1; 1	1; 1; 1	1; 1; 1	1; 1; 1	1; 1; 1
	Batch_Size	32	32	32	32	32	16	32
	Pooling_Type	mean	mean	max	max	sum	max	max
Factor_Num	2	4	4	4	2	2	4	

524 The result is presented in Figure 6. We can see from the results that the graph representations of
 525 GraphMAE are entangled. In contrast, the correlation pattern exhibited by DiGGR reveals four
 526 distinct diagonal blocks. This suggests that DiGGR is proficient at capturing mutually exclusive
 527 information within the latent factor, resulting in disentangled representations.

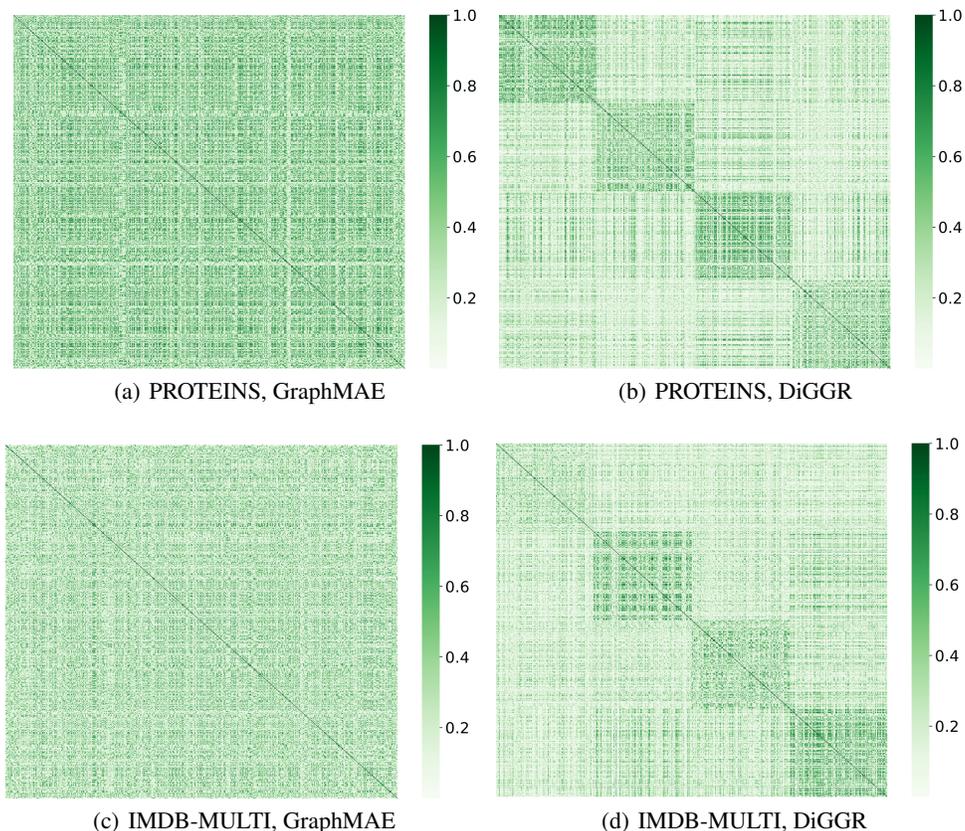


Figure 6: The absolute correlation between the representations learned by GraphMAE and DiGGR is measured on the **PROTEINS** and **IMDB-MULTI** datasets when $K = 4$.

528 **A.4 Training Algorithm**

Algorithm 1 The Overall Training Algorithm of DiGGR

- 1: **Input:** Graph $\mathcal{G} = \{V, A, X\}$; latent factor number K .
 - 2: **Parameters:** Θ in the inference network of Latent Factor Learning phase, Ω in the encoding network of DiGGR, Ψ in the decoding network of DiGGR.
 - 3: Initialize Θ , Ω , and Ψ ;
 - 4: **for** iter = 1, 2, \dots **do**
 - 5: Infer the *variational posterior* of \mathbf{z}_u based on Eq. 5;
 - 6: Sample latent factors \mathbf{z}_u from the variational posterior according to Eq. 6;
 - 7: Factorize the graph \mathcal{G} into K factor-wise groups $\{\mathcal{G}^{(k)}\}_{k=1}^K$ by node and edge factorization methods;
 - 8: Encoding $\{\mathcal{G}^{(k)}\}_{k=1}^K$ via latent factor-wise Graph Masked Autoencoder according to Eq. 8;
 - 9: Encoding \mathcal{G} via graph-level graph masked autoencoder according to Eq. 11;
 - 10: Calculate $\nabla_{\Theta, \Omega, \Psi} \mathcal{L}(\Theta, \Omega, \Psi; \mathcal{G})$ according to Eq. 13, and update parameters Θ , Ω , and Ψ jointly.
 - 11: **end for**=0
-

529 **A.5 Broader Impacts**

530 This paper presents work whose goal is to advance the field of Machine Learning. There are many
531 potential societal consequences of our work, none which we feel must be specifically highlighted
532 here.

533 **A.6 Limitations**

534 Despite the promising experimental justifications, our work might potentially suffer from limitation:
535 Although the complexity of the model is discussed in Section 3.4, and it is comparable to previously
536 published work, extending DiGGR to extremely large graph datasets remains challenging at this stage
537 due to the incorporation of an additional probabilistic model into the generative graph framework.
538 One potential solution to this problem could be utilizing PPR-Nibble [Andersen et al., 2006] for
539 efficient implementation, a method that has proven effective in some graph generative models [Hou
540 et al., 2023]. This approach will be pursued in our future work.

541 **NeurIPS Paper Checklist**

542 **1. Claims**

543 Question: Do the main claims made in the abstract and introduction accurately reflect the
544 paper's contributions and scope?

545 Answer: [\[Yes\]](#)

546 Justification: We listed our main contribution in the last paragraph of Section 1

547 Guidelines:

- 548 • The answer NA means that the abstract and introduction do not include the claims
549 made in the paper.
- 550 • The abstract and/or introduction should clearly state the claims made, including the
551 contributions made in the paper and important assumptions and limitations. A No or
552 NA answer to this question will not be perceived well by the reviewers.
- 553 • The claims made should match theoretical and experimental results, and reflect how
554 much the results can be expected to generalize to other settings.
- 555 • It is fine to include aspirational goals as motivation as long as it is clear that these goals
556 are not attained by the paper.

557 **2. Limitations**

558 Question: Does the paper discuss the limitations of the work performed by the authors?

559 Answer: [\[Yes\]](#)

560 Justification: We've discuss about the limitations in Appendix A.6

561 Guidelines:

- 562 • The answer NA means that the paper has no limitation while the answer No means that
563 the paper has limitations, but those are not discussed in the paper.
- 564 • The authors are encouraged to create a separate "Limitations" section in their paper.
- 565 • The paper should point out any strong assumptions and how robust the results are to
566 violations of these assumptions (e.g., independence assumptions, noiseless settings,
567 model well-specification, asymptotic approximations only holding locally). The authors
568 should reflect on how these assumptions might be violated in practice and what the
569 implications would be.
- 570 • The authors should reflect on the scope of the claims made, e.g., if the approach was
571 only tested on a few datasets or with a few runs. In general, empirical results often
572 depend on implicit assumptions, which should be articulated.
- 573 • The authors should reflect on the factors that influence the performance of the approach.
574 For example, a facial recognition algorithm may perform poorly when image resolution
575 is low or images are taken in low lighting. Or a speech-to-text system might not be
576 used reliably to provide closed captions for online lectures because it fails to handle
577 technical jargon.
- 578 • The authors should discuss the computational efficiency of the proposed algorithms
579 and how they scale with dataset size.
- 580 • If applicable, the authors should discuss possible limitations of their approach to
581 address problems of privacy and fairness.
- 582 • While the authors might fear that complete honesty about limitations might be used by
583 reviewers as grounds for rejection, a worse outcome might be that reviewers discover
584 limitations that aren't acknowledged in the paper. The authors should use their best
585 judgment and recognize that individual actions in favor of transparency play an impor-
586 tant role in developing norms that preserve the integrity of the community. Reviewers
587 will be specifically instructed to not penalize honesty concerning limitations.

588 **3. Theory Assumptions and Proofs**

589 Question: For each theoretical result, does the paper provide the full set of assumptions and
590 a complete (and correct) proof?

591 Answer: [\[NA\]](#)

592 Justification: This paper is not focus on theoretical explanations and assumptions

593 Guidelines:

- 594 • The answer NA means that the paper does not include theoretical results.
- 595 • All the theorems, formulas, and proofs in the paper should be numbered and cross-
596 referenced.
- 597 • All assumptions should be clearly stated or referenced in the statement of any theorems.
- 598 • The proofs can either appear in the main paper or the supplemental material, but if
599 they appear in the supplemental material, the authors are encouraged to provide a short
600 proof sketch to provide intuition.
- 601 • Inversely, any informal proof provided in the core of the paper should be complemented
602 by formal proofs provided in appendix or supplemental material.
- 603 • Theorems and Lemmas that the proof relies upon should be properly referenced.

604 4. Experimental Result Reproducibility

605 Question: Does the paper fully disclose all the information needed to reproduce the main ex-
606 perimental results of the paper to the extent that it affects the main claims and/or conclusions
607 of the paper (regardless of whether the code and data are provided or not)?

608 Answer: [Yes]

609 Justification: We have listed the specific settings of the experiment in the appendix A.2 of
610 the paper, including the datasets used and the hyperparameter settings of the model. We
611 have also uploaded the code in the supplementary material.

612 Guidelines:

- 613 • The answer NA means that the paper does not include experiments.
- 614 • If the paper includes experiments, a No answer to this question will not be perceived
615 well by the reviewers: Making the paper reproducible is important, regardless of
616 whether the code and data are provided or not.
- 617 • If the contribution is a dataset and/or model, the authors should describe the steps taken
618 to make their results reproducible or verifiable.
- 619 • Depending on the contribution, reproducibility can be accomplished in various ways.
620 For example, if the contribution is a novel architecture, describing the architecture fully
621 might suffice, or if the contribution is a specific model and empirical evaluation, it may
622 be necessary to either make it possible for others to replicate the model with the same
623 dataset, or provide access to the model. In general, releasing code and data is often
624 one good way to accomplish this, but reproducibility can also be provided via detailed
625 instructions for how to replicate the results, access to a hosted model (e.g., in the case
626 of a large language model), releasing of a model checkpoint, or other means that are
627 appropriate to the research performed.
- 628 • While NeurIPS does not require releasing code, the conference does require all submis-
629 sions to provide some reasonable avenue for reproducibility, which may depend on the
630 nature of the contribution. For example
 - 631 (a) If the contribution is primarily a new algorithm, the paper should make it clear how
632 to reproduce that algorithm.
 - 633 (b) If the contribution is primarily a new model architecture, the paper should describe
634 the architecture clearly and fully.
 - 635 (c) If the contribution is a new model (e.g., a large language model), then there should
636 either be a way to access this model for reproducing the results or a way to reproduce
637 the model (e.g., with an open-source dataset or instructions for how to construct
638 the dataset).
 - 639 (d) We recognize that reproducibility may be tricky in some cases, in which case
640 authors are welcome to describe the particular way they provide for reproducibility.
641 In the case of closed-source models, it may be that access to the model is limited in
642 some way (e.g., to registered users), but it should be possible for other researchers
643 to have some path to reproducing or verifying the results.

644 5. Open access to data and code

645 Question: Does the paper provide open access to the data and code, with sufficient instruc-
646 tions to faithfully reproduce the main experimental results, as described in supplemental
647 material?

648 Answer: [Yes]

649 Justification: We have uploaded the code in the supplementary material.

650 Guidelines:

- 651 • The answer NA means that paper does not include experiments requiring code.
- 652 • Please see the NeurIPS code and data submission guidelines ([https://nips.cc/
653 public/guides/CodeSubmissionPolicy](https://nips.cc/public/guides/CodeSubmissionPolicy)) for more details.
- 654 • While we encourage the release of code and data, we understand that this might not be
655 possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not
656 including code, unless this is central to the contribution (e.g., for a new open-source
657 benchmark).
- 658 • The instructions should contain the exact command and environment needed to run to
659 reproduce the results. See the NeurIPS code and data submission guidelines ([https:
660 //nips.cc/public/guides/CodeSubmissionPolicy](https://nips.cc/public/guides/CodeSubmissionPolicy)) for more details.
- 661 • The authors should provide instructions on data access and preparation, including how
662 to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- 663 • The authors should provide scripts to reproduce all experimental results for the new
664 proposed method and baselines. If only a subset of experiments are reproducible, they
665 should state which ones are omitted from the script and why.
- 666 • At submission time, to preserve anonymity, the authors should release anonymized
667 versions (if applicable).
- 668 • Providing as much information as possible in supplemental material (appended to the
669 paper) is recommended, but including URLs to data and code is permitted.

670 6. Experimental Setting/Details

671 Question: Does the paper specify all the training and test details (e.g., data splits, hyper-
672 parameters, how they were chosen, type of optimizer, etc.) necessary to understand the
673 results?

674 Answer: [Yes]

675 Justification: We have listed the specific experimental setup in Appendix A.2

676 Guidelines:

- 677 • The answer NA means that the paper does not include experiments.
- 678 • The experimental setting should be presented in the core of the paper to a level of detail
679 that is necessary to appreciate the results and make sense of them.
- 680 • The full details can be provided either with the code, in appendix, or as supplemental
681 material.

682 7. Experiment Statistical Significance

683 Question: Does the paper report error bars suitably and correctly defined or other appropriate
684 information about the statistical significance of the experiments?

685 Answer: [Yes]

686 Justification: In the main experiment 4, we ran 5 different random seeds for each dataset
687 and reported the average results and variances in the table 6 and 5.

688 Guidelines:

- 689 • The answer NA means that the paper does not include experiments.
- 690 • The authors should answer "Yes" if the results are accompanied by error bars, confi-
691 dence intervals, or statistical significance tests, at least for the experiments that support
692 the main claims of the paper.
- 693 • The factors of variability that the error bars are capturing should be clearly stated (for
694 example, train/test split, initialization, random drawing of some parameter, or overall
695 run with given experimental conditions).

- 696 • The method for calculating the error bars should be explained (closed form formula,
697 call to a library function, bootstrap, etc.)
- 698 • The assumptions made should be given (e.g., Normally distributed errors).
- 699 • It should be clear whether the error bar is the standard deviation or the standard error
700 of the mean.
- 701 • It is OK to report 1-sigma error bars, but one should state it. The authors should
702 preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis
703 of Normality of errors is not verified.
- 704 • For asymmetric distributions, the authors should be careful not to show in tables or
705 figures symmetric error bars that would yield results that are out of range (e.g. negative
706 error rates).
- 707 • If error bars are reported in tables or plots, The authors should explain in the text how
708 they were calculated and reference the corresponding figures or tables in the text.

709 8. Experiments Compute Resources

710 Question: For each experiment, does the paper provide sufficient information on the com-
711 puter resources (type of compute workers, memory, time of execution) needed to reproduce
712 the experiments?

713 Answer: [Yes]

714 Justification: We listed the experiment environment in AppendixA.2

715 Guidelines:

- 716 • The answer NA means that the paper does not include experiments.
- 717 • The paper should indicate the type of compute workers CPU or GPU, internal cluster,
718 or cloud provider, including relevant memory and storage.
- 719 • The paper should provide the amount of compute required for each of the individual
720 experimental runs as well as estimate the total compute.
- 721 • The paper should disclose whether the full research project required more compute
722 than the experiments reported in the paper (e.g., preliminary or failed experiments that
723 didn't make it into the paper).

724 9. Code Of Ethics

725 Question: Does the research conducted in the paper conform, in every respect, with the
726 NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines?>

727 Answer: [Yes]

728 Justification: We conform with the NeurIPS Code of Ethics in every respect for this paper.

729 Guidelines:

- 730 • The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- 731 • If the authors answer No, they should explain the special circumstances that require a
732 deviation from the Code of Ethics.
- 733 • The authors should make sure to preserve anonymity (e.g., if there is a special consid-
734 eration due to laws or regulations in their jurisdiction).

735 10. Broader Impacts

736 Question: Does the paper discuss both potential positive societal impacts and negative
737 societal impacts of the work performed?

738 Answer: [Yes]

739 Justification: We have discussed in appendix A.5

740 Guidelines:

- 741 • The answer NA means that there is no societal impact of the work performed.
- 742 • If the authors answer NA or No, they should explain why their work has no societal
743 impact or why the paper does not address societal impact.
- 744 • Examples of negative societal impacts include potential malicious or unintended uses
745 (e.g., disinformation, generating fake profiles, surveillance), fairness considerations
746 (e.g., deployment of technologies that could make decisions that unfairly impact specific
747 groups), privacy considerations, and security considerations.

- 748
- 749
- 750
- 751
- 752
- 753
- 754
- 755
- 756
- 757
- 758
- 759
- 760
- 761
- 762
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
 - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
 - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

763 **11. Safeguards**

764 Question: Does the paper describe safeguards that have been put in place for responsible
765 release of data or models that have a high risk for misuse (e.g., pretrained language models,
766 image generators, or scraped datasets)?

767 Answer: [NA]

768 Justification: this paper poses no such risks.

769 Guidelines:

- 770
- 771
- 772
- 773
- 774
- 775
- 776
- 777
- 778
- 779
- The answer NA means that the paper poses no such risks.
 - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
 - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
 - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

780 **12. Licenses for existing assets**

781 Question: Are the creators or original owners of assets (e.g., code, data, models), used in
782 the paper, properly credited and are the license and terms of use explicitly mentioned and
783 properly respected?

784 Answer: [Yes]

785 Justification: All these are properly credited.

786 Guidelines:

- 787
- 788
- 789
- 790
- 791
- 792
- 793
- 794
- 795
- 796
- 797
- 798
- 799
- The answer NA means that the paper does not use existing assets.
 - The authors should cite the original paper that produced the code package or dataset.
 - The authors should state which version of the asset is used and, if possible, include a URL.
 - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
 - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
 - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
 - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

800 • If this information is not available online, the authors are encouraged to reach out to
801 the asset’s creators.

802 13. **New Assets**

803 Question: Are new assets introduced in the paper well documented and is the documentation
804 provided alongside the assets?

805 Answer: [NA]

806 Justification: This paper does not release new assets.

807 Guidelines:

- 808 • The answer NA means that the paper does not release new assets.
- 809 • Researchers should communicate the details of the dataset/code/model as part of their
810 submissions via structured templates. This includes details about training, license,
811 limitations, etc.
- 812 • The paper should discuss whether and how consent was obtained from people whose
813 asset is used.
- 814 • At submission time, remember to anonymize your assets (if applicable). You can either
815 create an anonymized URL or include an anonymized zip file.

816 14. **Crowdsourcing and Research with Human Subjects**

817 Question: For crowdsourcing experiments and research with human subjects, does the paper
818 include the full text of instructions given to participants and screenshots, if applicable, as
819 well as details about compensation (if any)?

820 Answer: [NA]

821 Justification: This paper does not involve crowdsourcing nor research with human subjects.

822 Guidelines:

- 823 • The answer NA means that the paper does not involve crowdsourcing nor research with
824 human subjects.
- 825 • Including this information in the supplemental material is fine, but if the main contribu-
826 tion of the paper involves human subjects, then as much detail as possible should be
827 included in the main paper.
- 828 • According to the NeurIPS Code of Ethics, workers involved in data collection, curation,
829 or other labor should be paid at least the minimum wage in the country of the data
830 collector.

831 15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human 832 Subjects**

833 Question: Does the paper describe potential risks incurred by study participants, whether
834 such risks were disclosed to the subjects, and whether Institutional Review Board (IRB)
835 approvals (or an equivalent approval/review based on the requirements of your country or
836 institution) were obtained?

837 Answer: [NA]

838 Justification: This paper does not involve crowdsourcing nor research with human subjects.

839 Guidelines:

- 840 • The answer NA means that the paper does not involve crowdsourcing nor research with
841 human subjects.
- 842 • Depending on the country in which research is conducted, IRB approval (or equivalent)
843 may be required for any human subjects research. If you obtained IRB approval, you
844 should clearly state this in the paper.
- 845 • We recognize that the procedures for this may vary significantly between institutions
846 and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the
847 guidelines for their institution.
- 848 • For initial submissions, do not include any information that would break anonymity (if
849 applicable), such as the institution conducting the review.