Exploring Correlations of Self-Supervised Tasks for Graphs

Taoran Fang [1] **Wei Zhou** [1] **Yifei Sun** [1] **Kaiqiao Han** [1] **Lvbin Ma** [2] **Yang Yang** * [1]

Abstract

Graph self-supervised learning has sparked a research surge in training informative representations without accessing any labeled data. However, our understanding of graph self-supervised learning remains limited, and the inherent relationships between various self-supervised tasks are still unexplored. Our paper aims to provide a fresh understanding of graph self-supervised learning based on task correlations. Specifically, we evaluate the performance of the representations trained by one specific task on other tasks and define correlation values to quantify task correlations. Through this process, we unveil the task correlations between various self-supervised tasks and can measure their expressive capabilities, which are closely related to downstream performance. By analyzing the correlation values between tasks across various datasets, we reveal the complexity of task correlations and the limitations of existing multi-task learning methods. To obtain more capable representations, we propose Graph Task Correlation Modeling (GraphTCM) to illustrate the task correlations and utilize it to enhance graph self-supervised training. The experimental results indicate that our method significantly outperforms existing methods across various downstream tasks.

1. Introduction

Graph neural networks (GNNs) have garnered widespread attention from researchers due to their remarkable success in graph representation learning (Kipf & Welling, 2017; Hamilton et al., 2017; Xu et al., 2019; Wu et al., 2019). Most existing studies on GNNs have focused on (semi-)supervised scenarios and have been widely applied in various downstream tasks such as node classification (Jiang et al., 2019), link prediction (Kipf & Welling, 2016b), vertex clustering (Ramaswamy et al., 2005), and recommendation systems (Ying et al., 2018). However, two fundamental challenges hinder the large-scale practical applications of GNNs. One is the scarcity of high-quality labeled data in the real world (Zitnik et al., 2018), and the other is the low generalization ability of the trained models caused by over-fitting (Rong et al., 2019; Feng et al., 2020; Fang et al., 2022).

To address these challenges, graph self-supervised learning (Liu et al., 2021; Wu et al., 2021; Xie et al., 2021) offers a promising training paradigm to reduce the reliance on task labels and demonstrates significant potential in strengthening GNNs in an unsupervised manner. These methods specifically construct a variety of graph self-supervised tasks based on structural and attribute information from the training graphs, wherein the supervision signals are automatically acquired from the graph itself without accessing any labeled data. By employing well-designed graph self-supervised tasks, these methods enable the GNN model to extract more informative representations from unlabeled data, leading to improved generalization (Hu et al., 2020c; Qiu et al., 2020; Hu et al., 2020b; Fang et al., 2024; Huang et al., 2024) and robustness (You et al., 2020b; Jovanovi'c et al., 2021) on downstream datasets.

However, two fundamental problems impede further comprehension of graph self-supervised learning. The first problem is the lack of understanding regarding the correlations between different graph self-supervised tasks. Despite the multitude of graph self-supervised tasks with diverse training objectives, discussions on the correlations between these tasks are scarce. Additionally, the differences in trained representations between various self-supervised tasks remain unexplored, limiting a comprehensive understanding of the expressive capabilities of the trained representations. The second problem revolves around how to train sufficiently general and robust self-supervised representations. While existing graph self-supervised methods can train representations that excel on their respective tasks, there is no direct guarantee of the performance of trained representations on downstream tasks. In practice, downstream tasks encompass diverse forms, posing a significant challenge in ensuring the universality of trained representations.

[1]College of Computer Science and Technology, Zhejiang University, Hangzhou, China [2]Zhejiang Huayun Information Technology, Hangzhou, China. Our code is available at: https://github.com/zjunet/GraphTCM. Correspondence to: Yang Yang <yangya@zju.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

In this paper, we present a novel perspective on graph self-supervised learning based on trained representations. Specifically, we obtain trained representations of these tasks on various datasets and assess the performance of these trained representations on other tasks. To address the first problem, we introduce *Correlation Values* to measure the correlation between two tasks quantitatively. More precisely, the correlation value $\text{Cor}(t_1, t_2)$ is defined as the relative loss when using representations trained by task t_1 on the objective function of task t_2. A lower value of $\text{Cor}(t_1, t_2)$ indicates that the representations trained by task t_1 perform better on task t_2. Furthermore, we propose Average Task Difficulty (ATD) and Average Relative Loss (ARL) to compare the task difficulty and the representation capability between various tasks, providing a comprehensive understanding of the expressive capabilities of the representations trained by this self-supervised task.

To tackle the second problem, we theoretically and empirically demonstrate that if trained representations excel across various self-supervised tasks, they are also likely to exhibit universality and effectiveness across a range of downstream tasks. In pursuing this objective, several pioneering works (Jin et al., 2022; Ju et al., 2023; Han et al., 2021) apply multi-task learning to combine the losses of different self-supervised tasks adaptively. However, upon evaluating the performance of the representations trained by these methods across various self-supervised tasks, our results indicate that these methods are unable to train representations that perform well across all self-supervised tasks, and they are even unable to surpass the representations trained by a single task. From a completely new perspective, we introduce GraphTCM to model the task correlations among various graph self-supervised tasks and utilize it to obtain representations that perform effectively on all involved self-supervised tasks. With a well-trained GraphTCM, we can obtain trained representations that perform competently across various self-supervised and downstream tasks by only introducing a linear transformation with limited parameters. Overall, the contributions of our work can be summarized as follows:

- Providing a new understanding of graph self-supervised learning based on trained representations, we calculate the correlation values to obtain quantitative characterizations of the correlations between various graph self-supervised tasks and the expressive capabilities of their trained representations.
- We explore the intricate correlations of various graph self-supervised tasks across diverse real datasets and verify that existing multi-task learning methods fail to train representations that perform effectively across all self-supervised tasks.
- We introduce GraphTCM to model task correlations and utilize it to enhance self-supervised training. Experimental results demonstrate that our GraphTCM-enhanced representations not only exhibit strong performance across all self-supervised tasks but also significantly outperform existing methods across all downstream tasks.

2. Related work

Graph Self-supervised Learning. Graph self-supervised learning addresses the scarcity of labeled data and combats low generalization in graph models by reducing reliance on task-specific labels and designing tasks capable of mining knowledge of input graphs (Wu et al., 2021; Xie et al., 2021). Existing graph self-supervised tasks can be primarily categorized into four types: feature-based, structure-based, auxiliary property-based, and contrastive-based tasks (Liu et al., 2021). Specifically, feature-based tasks (Jin et al., 2020; You et al., 2020b; Hou et al., 2022b; Hu et al., 2020b; Wang et al., 2017; Manessi & Rozza, 2020; Park et al., 2019) focus on reconstructing the graph feature, while structure-based tasks (Kipf & Welling, 2016b; Li et al., 2023; Hasanzadeh et al., 2019; Pan et al., 2018; Kim & Oh, 2022; Hu et al., 2019) reconstruct the graph adjacency or masked edges. Meanwhile, auxiliary property-based tasks (Jin et al., 2020; You et al., 2020b; Sun et al., 2019; Zhu et al., 2020b; Peng et al., 2020; Kang et al., 2021) guide models by predicting certain pre-computable auxiliary information (*e.g.*, node degree). Contrast-based tasks are also widely used in practice. These tasks (Velickovic et al., 2019b; Jiao et al., 2020; You et al., 2020a; Qiu et al., 2020; Xu et al., 2022; Zhang et al., 2023; Wang & Qi, 2022) do not explicitly incorporate the desired information as the prediction targets, but instead, they focus on the mutual information (MI) maximization (Hjelm et al., 2019) where the estimated MI between augmented instances of the same object is maximized.

Graph Multi-task Learning. Several pioneering approaches have employed multi-task training methods to integrate various graph self-supervised tasks to acquire representations that can effectively perform across a spectrum of self-supervised and downstream tasks. AUX-TS (Han et al., 2021) introduces an adaptive mechanism for selecting and combining diverse pre-training tasks with target tasks during the fine-tuning phase. AutoSSL (Jin et al., 2022) integrates multiple self-supervised tasks using learnable coefficients, leveraging the homophily assumption as a basis for the integration process. ParetoGNN (Ju et al., 2023) aims to dynamically adjust the coefficients of task combination during the self-supervised training process, with the objective of facilitating Pareto optimality. However, these methods primarily concentrate on combining losses from different tasks, thus neglecting the disparity between the trained and ideal representations and failing to characterize

the trained representations adequately.

3. Exploring the Task Correlations

3.1. Preliminaries

Notations. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}) \in \mathbb{G}$ represents a graph, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denote the node set and edge set respectively. The node features can be denoted as a matrix $\mathbf{X} = \{x_1, x_2, \ldots, x_N\} \in \mathbb{R}^{N \times F}$, where $x_i \in \mathbb{R}^F$ is the feature of the node v_i, and F is the dimensionality of node features. $\mathbf{A} \in \{0,1\}^{N \times N}$ denotes the adjacency matrix, where $\mathbf{A}_{ij} = 1$ if $(v_i, v_j) \in \mathcal{E}$.

Graph Self-supervised Learning. For a GNN model g parameterized by θ, a linear tuning head $\mathbf{W}$ (possibly not required), a graph $\mathcal{G} : (\mathbf{A}, \mathbf{X})$ and a self-supervised target $\mathbf{Y}_{\text{ssl}}$, we search for the parameters $\hat{\theta}$ and $\hat{\mathbf{W}}$ that minimize the self-supervised loss $\boldsymbol{l}_{\text{ssl}}$ during the training process, which can be expressed as:

$$\hat{\theta}, \hat{\mathbf{W}} = \arg\min_{\theta, \mathbf{W}} \boldsymbol{l}_{\text{ssl}}(g(\mathbf{A}, \mathbf{X}) \cdot \mathbf{W}, \mathbf{Y}_{\text{ssl}}) \tag{1}$$

Existing graph self-supervised methods can be categorized into four primary types (Liu et al., 2021): *feature-based (FB)*, *structure-based (SB)*, *auxiliary property-based (APB)* and *contrast-based (CB)*. To comprehensively understand the complex relationships in graph self-supervised tasks, we have chosen two representative methods from each category for detailed analysis. Table 1 provides a summary of their training targets, and their detailed descriptions can be found in Appendix A.1.

Table 1. Overview of the involved self-supervised tasks.

Method	Category	Training Target
GraphComp (You et al., 2020b)	FB	Node Feature
AttributeMask (Jin et al., 2020)	FB	PCA Node Feature
GAE (Kipf & Welling, 2016a)	SB	Adjacency Matrix
EdgeMask (Jin et al., 2020)	SB	Masked Edge
NodeProp (Jin et al., 2020)	APB	Degree Calculation
DisCluster (Jin et al., 2020)	APB	Distance Calculation
DGI (Velickovic et al., 2019a)	CB	Cross-scale Contrast
SubgCon (Jiao et al., 2020)	CB	Same-scale Contrast

3.2. Building the Task Correlations

We transfer trained representations from one self-supervised task to other tasks and evaluate the correlation between self-supervised tasks by examining the performance of the transferred representations on these tasks. According to Formula 1, the training process of graph self-supervision can be interpreted as an optimization process for the self-supervised objective function. Consequently, assessing the loss of representations for a self-supervised task can indicate whether the representation possesses the expressive capability required for that particular task. In contrast to subjective judgments based on training objectives or processes for different self-supervised tasks, our assessment is objective and definitive, revealing the actual characteristics of various self-supervised methods in practical scenarios.

For a graph $\mathcal{G} : (\mathbf{A}, \mathbf{X})$, we denote the set of self-superivsed tasks as $\mathcal{T} = \{t_1, t_2, ..., t_k\}$, where t_i denotes a specific self-supervised task. Additionally, we denote the loss function of the task t_i as $\boldsymbol{l}_{t_i}(\cdot)$, and denote the target matrix of the task t_i as $\mathbf{Y}_{t_i}$. Considering a GNN model g and a possible linear tuning head $\mathbf{W}$, we assume the model obtained through gradient descent based on the loss function $\boldsymbol{l}_{t_i}(g(\mathbf{A}, \mathbf{X}) \cdot \mathbf{W}, \mathbf{Y}_{t_i})$ is denoted as g_{t_i}. Furthermore, we denote the graph representations obtained by g_{t_i} as $\mathbf{H}_{t_i} = g_{t_i}(\mathbf{A}, \mathbf{X})$.

Definition 3.1. *(Correlation Value)* Given two self-supervised tasks $t_1, t_2 \in \mathcal{T}$, a graph $\mathcal{G} : (\mathbf{A}, \mathbf{X})$, we define the correlation value $\text{Cor}(t_1, t_2)$ as:

$$\text{Cor}(t_1, t_2) = \frac{\min_{\mathbf{W}_{t_1}} \boldsymbol{l}_{t_2}(\mathbf{H}_{t_1} \cdot \mathbf{W}_{t_1}, \mathbf{Y}_{t_2})}{\min_{\mathbf{W}_{t_2}} \boldsymbol{l}_{t_2}(\mathbf{H}_{t_2} \cdot \mathbf{W}_{t_2}, \mathbf{Y}_{t_2})} \tag{2}$$

Both the numerator and the denominator are optimized using gradient descent. It is important to note that the linear tuning heads $\mathbf{W}_{t_1}$ and $\mathbf{W}_{t_2}$ in the numerator and denominator of $\text{Cor}(t_1, t_2)$ do not have to be identical. $\text{Cor}(t_1, t_2)$ reflects the relative performance of the representations $\mathbf{H}_{t_1}$ trained by task t_1 on the task t_2. It is evident that $\text{Cor}(t_i, t_i)$ is equal to 1. The magnitude of the correlation values can be elucidated from two perspectives: *representation capability* and *task difficulty*. On one hand, $\text{Cor}(t_1, t_2)$ indicates the capability of the representations $\mathbf{H}_{t_1}$. If $\text{Cor}(t_1, t_2)$ is close to or less than 1, it signifies that the representations trained by task t_1 possess the expressive capability required for task t_2. On the other hand, $\text{Cor}(t_1, t_2)$ also indicates the difficulty of t_2. If $\text{Cor}(t_1, t_2)$ is large, it implies that task t_2 is challenging for the representations trained by task t_1. To further delineate representation capability and task difficulty, we introduce the following definitions.

Definition 3.2. *(Average Task Difficulty)* Given a graph $\mathcal{G} : (\mathbf{A}, \mathbf{X})$, a set of self-superivsed tasks $\mathcal{T} = \{t_1, t_2, ..., t_k\}$, the average task difficulty (ATD) for task t_i is calculated as:

$$\text{ATD}_{t_i} = \frac{1}{k-1} \sum_{j \neq i}^{k} \text{Cor}(t_j, t_i) \tag{3}$$

Definition 3.3. *(Average Relative Loss)* Given a graph $\mathcal{G} : (\mathbf{A}, \mathbf{X})$, a set of self-superivsed tasks $\mathcal{T} = \{t_1, t_2, ..., t_k\}$, the average relative loss (ARL) for task t_i is calculated as:

$$\text{ARL}_{t_i} = \frac{1}{k}(\text{ATD}_{t_i} + \sum_{j \neq i}^{k} \text{Cor}(t_i, t_j)) \tag{4}$$

ATD is directly associated with task difficulty, indicating that the more challenging a task is, the higher the ATD

(a) Cora (b) CiteSeer (c) PubMed (d) ogbn-arxiv

Figure 1. The real correlation values on various graph datasets. Here, the y-axis represents the task used to train the assessed representations, while the x-axis represents the self-supervised task on which the representations are evaluated. For instance, the intersection of the DGI row and the GAE column is denoted as Cor(DGI, GAE) according to Formula 2, signifying the comparative performance of the representations trained by DGI on the GAE task. A lower value indicates better performance.

value it achieves. As for ARL, the scenario is different. A low ARL value for a task signifies that the representations trained by this task exhibit high representation capability, as they perform well on other tasks. When computing ARL_{t_i}, we incorporate an additional ATD_{t_i} before calculating the average to prevent a task from disproportionately benefiting due to its high task difficulty. This adjustment enables a more equitable comparison of representation capabilities across different tasks. Subsequently, we will illustrate the intricate correlations among various graph self-supervised tasks across diverse datasets.

3.3. Unveiling the T ask Correlations

In this chapter, we compute the correlation values for the 8 graph self-supervised tasks outlined in Table 1 across 4 graph datasets (*i.e.*, Cora, CiteSeer, PubMed and ogbn-arxiv). A concise description of these datasets can be found in Appendix B.1. Figure 1 illustrates the correlation values among various tasks across these four datasets. Additionally, Table 2 and 3 summarize the values and rankings of ATD and ARL, respectively.

Table 2. The statistics of ATD for various self-supervised tasks (the larger value indicates higher task difficulty ↑). The numbers in the parentheses signify the *ranking* of task ATD within the same dataset, where a smaller number implies greater task difficulty.

Task	Cora	CiteSeer	PubMed	ogbn-arxiv
GraphComp	1.00 (**7**)	0.99 (**7**)	0.99 (**6**)	1.30 (**2**)
AttributeMask	0.88 (**8**)	0.90 (**8**)	0.93 (**8**)	0.97 (**8**)
GAE	1.48 (**6**)	1.56 (**6**)	1.85 (**3**)	1.02 (**6**)
EdgeMask	3.29 (**2**)	4.08 (**2**)	1.92 (**2**)	1.17 (**4**)
NodeProp	3.10 (**3**)	2.79 (**4**)	1.53 (**4**)	1.25 (**3**)
DisCluster	1.49 (**5**)	1.87 (**5**)	0.96 (**7**)	1.57 (**1**)
DGI	7.69 (**1**)	6.92 (**1**)	2.13 (**1**)	1.00 (**7**)
SubgCon	2.73 (**4**)	4.04 (**3**)	1.01 (**5**)	1.16 (**5**)

Table 3. The statistics of ARL for various self-supervised tasks (the lower value indicates stronger representation capability ↓). The numbers in the parentheses signify the *ranking* of task ARL within the same dataset, where a smaller number implies a stronger representation capability.

Task	Cora	CiteSeer	PubMed	ogbn-arxiv
GraphComp	2.93 (**7**)	3.23 (**7**)	1.55 (**7**)	1.27 (**6**)
AttributeMask	3.00 (**8**)	3.36 (**8**)	1.56 (**8**)	1.12 (**4**)
GAE	2.58 (**3**)	2.59 (**2**)	1.29 (**2**)	1.39 (**8**)
EdgeMask	2.71 (**4**)	3.00 (**4**)	1.37 (**3**)	1.06 (**2**)
NodeProp	2.82 (**5**)	3.09 (**5**)	1.40 (**5**)	1.06 (**3**)
DisCluster	2.91 (**6**)	3.23 (**6**)	1.51 (**6**)	1.15 (**5**)
DGI	2.51 (**2**)	2.62 (**3**)	1.27 (**1**)	1.36 (**7**)
SubgCon	2.18 (**1**)	2.03 (**1**)	1.38 (**4**)	1.03 (**1**)

Overall Analysis. The correlation values depict the inherent relationships among different graph self-supervised tasks. According to our findings, the relationships between graph self-supervised tasks are considerably more intricate than previously assumed.

The correlations among graph self-supervised tasks exhibit variability across different datasets. In contrast to the consistent correlations observed for self-supervised tasks in other domains (Zamir et al., 2018), the correlations among graph self-supervised tasks fluctuate with shifts in the dataset, suggesting that merely considering the correlations of graph self-supervised tasks from the standpoint of training processes or objective functions is insufficient.

A straightforward relationship between representation capability and task difficulty is absent. The tables in 2 and 3 indicate that task difficulty and representation capability do not consistently display a positive or negative correlation. This suggests that opting for more challenging tasks does not automatically improve representation capability.

Feature-based graph self-supervised tasks generally exhibit lower task difficulty and are often overlapped by other tasks. This phenomenon may arise from the utilization of node features as auxiliary information in other self-supervised tasks, leading to representations learned by these tasks also encompassing information about node features.

The performance of multi-task training. To obtain representations that perform effectively across various tasks, most existing methods explore solutions from the perspective of multi-task training. Thus, *can we attain more capable representations through multi-task training?* To answer this question, we assess the ARL of representations obtained through three multi-task training methods (Multi-loss, AutoSSL (Jin et al., 2022), ParetoGNN (Ju et al., 2023)) as well as two straightforward representation mixing methods (representation addition and representation concat). The specific implementations for these methods can be found in Appendix A.2, with the base tasks for their multi-task training involving eight self-supervised tasks. Table 4 presents a summary of the evaluation results. From the findings presented in Table 4, it is evident that multi-task training scarcely enhances representation capability. These methods do not outperform the representation capability of single tasks on any dataset, and in certain instances, they even demonstrate notable declines in representation capability (*e.g.*, ogbn-arxiv). This suggests that optimizing the losses of multiple tasks together not only falls short in training representations with versatile capabilities, but also significantly compromises representation capability. The reasons for this phenomenon might stem from the varying scales of loss for different self-supervised tasks (Jin et al., 2022), making their combination challenging; the different optimization directions of various self-supervised tasks leading to confusion when aggregated (Han et al., 2021); and the inability of existing multi-task training methods to capture the intrinsic relationships between individual self-supervised tasks during the training process.

Table 4. The statistics of ARL for various multi-task representations. The numbers within the parentheses signify the *ranking* of the ARL values compared with the results in Table 3, where a smaller number implies stronger representations capability.

Method	Cora	CiteSeer	PubMed	ogbn-arxiv
Addition	2.71 (**5**)	2.79 (**4**)	2.17 (**9**)	2.78 (**9**)
Concat	2.32 (**2**)	2.26 (**2**)	1.32 (**3**)	1.82 (**9**)
Multi-loss	2.81 (**6**)	3.32 (**8**)	1.51 (**7**)	1.92 (**9**)
AutoSSL	4.53 (**9**)	2.73 (**4**)	2.37 (**9**)	2.38 (**9**)
ParetoGNN	3.01 (**9**)	2.94 (**4**)	2.31 (**9**)	1.90 (**9**)

3.4. Modeling the Task Correlations

The correlation values among different self-supervised tasks signify the intrinsic relationships between these tasks. However, existing graph self-supervised methods have not considered these vital relationships. We are the first to concentrate on the task correlations among graph self-supervised tasks and to leverage these correlations to enhance representation training. To model the correlations between self-supervised tasks, we introduce the **Graph T**ask **C**orrelation **M**odeling (GraphTCM). Subsequently, we employ this correlation-auxiliary model to assist us in training representations with high representation capability.

Overview. Our objective is to characterize the correlation values between different graph self-supervised tasks. Based on our earlier analysis, correlation values will fluctuate with changes in datasets. For a certain dataset $\mathcal{D}$, we have:

$$\text{Cor}(t_i, t_j) = f_{\mathcal{D}}(t_i, t_j) \tag{5}$$

where t_i and t_j are two distinct tasks, and $f_{\mathcal{D}}$ is a specific mapping determined by the dataset.

Graph Task Correlation Modeling. GraphTCM leverages the computed correlation values as ground truth, aiming to characterize the mapping $f_{\mathcal{D}}$. For a dataset $\mathcal{D}$, we utilize the representation $\mathbf{H}_{t_i}$ and $\mathbf{H}_{t_j}$ trained by tasks t_i and t_j as inputs to derive their correlation value, expressed as:

$$\text{GraphTCM}_{\mathcal{D}}(\mathbf{H}_{t_i}, \mathbf{H}_{t_j}) \rightarrow \text{Cor}(t_i, t_j) \tag{6}$$

The concrete implementation of GraphTCM can be expressed as:

$$q_{t_i} = \text{Readout}(\mathbf{H}_{t_i} \cdot \mathbf{W}_r) \quad k_{t_j} = \text{Readout}(\mathbf{H}_{t_j} \cdot \mathbf{W}_t)$$
$$\text{GraphTCM}_{\mathcal{D}}(\mathbf{H}_{t_i}, \mathbf{H}_{t_j}) = \exp(q_{t_i} \cdot k_{t_j}^T) \tag{7}$$

where $\mathbf{W}_r, \mathbf{W}_t \in \mathbb{R}^{d \times d'}$ are two linear transformations. The readout functions here can be selected from various pooling functions (*e.g.*, mean pooling). Our aim is for the values derived by GraphTCM to align closely with the actual correlation values. Consequently, for the dataset $\mathcal{D}$, the set of self-superivsed tasks $\mathcal{T} = \{t_1, t_2, ..., t_k\}$, the loss function for GraphTCM can be expressed as:

$$\boldsymbol{l} = \sum_{i,j \in [1,k]} \|\text{GraphTCM}_{\mathcal{D}}(\mathbf{H}_{t_i}, \mathbf{H}_{t_j}) - \text{Cor}(t_i, t_j)\| \tag{8}$$

The architecture of GraphTCM is innovative and sophisticated. GraphTCM accepts graph representations $\mathbf{H}$ as input, enabling its use even when the model architectures employed by the two self-supervised tasks differ, thus demonstrating considerable flexibility. Additionally, GraphTCM accounts for the non-negativity and asymmetry of correlation values, enhancing the precision and controllability of the modeling process.

Training More Capable Representations. Upon considering a dataset $\mathcal{D}$ and a set of graph self-supervised

tasks $\mathcal{T} = \{t_1, t_2, ..., t_k\}$ along with their trained representations $\mathcal{H} = \{\mathbf{H}_{t_1}, \mathbf{H}_{t_2}, ..., \mathbf{H}_{t_k}\}$, following the training of a GraphTCM model, we fix its parameters and utilize it to train self-supervised representations possessing high representation capability. The entire process is notably straightforward. For any model g with parameters ϕ capable of obtaining graph representations, we have:

$$\mathbf{H}_\phi = g_\phi(\mathcal{D}) \tag{9}$$

where g can represent a GNN model, an MLP, or even a linear transformation. Our objective is to minimize the correlation values of $\mathbf{H}_\phi$ across various tasks. Consequently, we aim to find the optimal parameters ϕ^* of the model g that satisfy the following equation:

$$\phi^* = \arg\min_\phi \sum_{i\in[1,k]} \frac{1}{\text{ATD}_{t_i}} \cdot \|\text{GraphTCM}(g_\phi(\mathcal{D}), \mathbf{H}_{t_i})\| \tag{10}$$

Through the utilization of a trained GraphTCM model and employing representation capability as the objective function, we have acquired representations $\mathbf{H}_{\phi^*}$ that are expected to perform effectively across various self-supervised tasks and $\mathbf{H}_{\phi^*}$ will be applied to downstream tasks.

3.5. Theoretical Analysis

In this chapter, we theoretically establish the relationship between representation capability and downstream task performance, demonstrating that trained representations with high representation capability lead to improved performance in downstream tasks. For analytical simplification, we assume that the loss functions for self-supervised tasks and downstream tasks adhere to the following forms:

$$\boldsymbol{l} = \|\mathbf{H} \cdot \mathbf{W} - \mathbf{Y}_{\text{target}}\| \tag{11}$$

where $\mathbf{H}$ denotes the trained representations, $\mathbf{W}$ denotes the learnable tuning head and $\mathbf{Y}_{\text{target}}$ denotes the task target. For a set of graph self-supervised tasks $\mathcal{T} = \{t_1, t_2, ..., t_k\}$ and a downstream task t_{ds}, we denote their $\mathbf{Y}_{\text{target}}$ as $\{\mathbf{Y}_{t_1}, \mathbf{Y}_{t_2}, ..., \mathbf{Y}_{t_k}\}$ and $\mathbf{Y}_{t_{ds}}$. Without loss of generality, we assume that these training targets $\mathbf{Y}_{\text{target}}$ have the same scale. According to Definition 3.1, given the trained representations $\mathcal{H} = \{\mathbf{H}_{t_1}, \mathbf{H}_{t_2}, ..., \mathbf{H}_{t_k}\}$, the correlation values can be calculated as:

$$\text{Cor}(t_1, t_2) = \frac{\min_{\mathbf{W}_{t_1}} \|\mathbf{H}_{t_1} \cdot \mathbf{W}_{t_1} - \mathbf{Y}_{t_2}\|}{\min_{\mathbf{W}_{t_2}} \|\mathbf{H}_{t_2} \cdot \mathbf{W}_{t_2} - \mathbf{Y}_{t_2}\|} \tag{12}$$

Theorem 3.4. *Given two task $t_1, t_2 \in \mathcal{T}$ and their trained representations $\mathbf{H}_{t_1}, \mathbf{H}_{t_2}$, the error of $\mathbf{H}_{t_2}$ on the downstream task is e_{t_2}, which can be expressed as:*

$$e_{t_2} = \min_{\mathbf{W}^*_{t_2}} \|\mathbf{H}_{t_2} \cdot \mathbf{W}^*_{t_2} - \mathbf{Y}_{t_{ds}}\| \tag{13}$$

Then, the error of $\mathbf{H}_{t_1}$ on the downstream task satisfies:

$$e_{t_1} \leq \text{Cor}(t_1, t_2) \cdot (e_{t_2} + \triangle) + \|\mathbf{Y}_{t_2} - \mathbf{Y}_{t_{ds}}\| \tag{14}$$

where $\triangle = \|\mathbf{Y}_{t_2} - \mathbf{Y}_{t_{ds}}\| + \|\mathbf{H}_{t_2} \cdot (\hat{\mathbf{W}}^*_{t_2} - \hat{\mathbf{W}}_{t_2})\|$, and $\hat{\mathbf{W}}^*_{t_2}, \hat{\mathbf{W}}_{t_2}$ denote the optimal heads for $\mathbf{W}^*_{t_2}$ and $\mathbf{W}_{t_2}$.

The complete proof of Theorem 3.4 can be found in Appendix A.5. This theorem establishes the relationship between correlation values and downstream tasks. According to Formula 14, as $\text{Cor}(t_1, t_2)$ decreases, the upper bound on the error of representations $\mathbf{H}_{t_1}$ on downstream tasks decreases, indicating that $\mathbf{H}_{t_1}$ can achieve better downstream task performance. In addition, we can also observe that in Formula14, there are several fixed terms related to task t_2, which are determined by the correlation between the self-supervised task t_2 and the downstream task t_{ds}. However, the relationship between various self-supervised tasks and downstream tasks is unknown during the self-supervised training phase. Therefore, we aim for trained representations to have low correlation values across all self-supervised tasks. If so, the representations are expected to perform well on any downstream task.

Theorem 3.5. *Given a set of tasks $\mathcal{T} = \{t_1, t_2, ..., t_k\}$ with their trained representations $\mathcal{H} = \{\mathbf{H}_{t_1}, \mathbf{H}_{t_2}, ..., \mathbf{H}_{t_k}\}$, for any downstream task t_{ds}, there exists a $\beta \geq \|\mathbf{Y}_{t_i} - \mathbf{Y}_{t_{ds}}\| + \|\mathbf{H}_{t_i} \cdot (\hat{\mathbf{W}}^*_{t_i} - \hat{\mathbf{W}}_{t_i})\|$ for any $t_i \in \mathcal{T}$. If a new task t' and its trained representations $\mathbf{H}_{t'}$ that satisfies $\delta \geq \text{Cor}(t', t_i)$ for any $t_i \in \mathcal{T}$, the error of $\mathbf{H}_{t'}$ on downstream task holds:*

$$e_{t'} \leq \delta \cdot (e_{\min} + \beta) + \beta \tag{15}$$

where $e_{\min} = \min\{e_{t_1}, e_{t_2}, ..., e_{t_k}\}$.

The complete proof of Theorem 3.5 can be found in Appendix A.6. Formula 15 does not involve any specific relationship between individual self-supervised tasks and downstream tasks; instead, it provides a global perspective closer to real-world scenarios. When we attempt to make representations have high representation capability across all self-supervised tasks, we are essentially minimizing the value of δ in Formula 15, which theoretically enhances the performance of the trained representations on any downstream task. Therefore, aiming for high representation capability and training representations according to Formula 10 not only makes practical sense but also has theoretical guarantees.

4. Experiments

To systematically validate the effectiveness of our method, we answer the following three questions through experiments: **Q1.** Can GraphTCM model the correlations between graph self-supervised tasks and demonstrate its generalization? **Q2.** Can a well-trained GraphTCM help us obtain

representations with high representation capability across all tasks? **Q3.** Do representations trained by GraphTCM perform better on downstream tasks? We answer these questions in Section 4.2, 4.3 and 4.4 respectively.

4.1. Experiment Setup

Datasets. We employ 6 graph datasets in our experiments, including 4 citation network datasets *Cora, CiteSeer, PubMed, ogbn-arxiv* and 2 product network datasets *Amazon-Computers, Amazon-Photo*. Their detailed introductions can be found in Appendix B.1.

Involved Graph Self-supervised Tasks. In our experiments, we still chose the 8 graph self-supervised tasks (*i.e.*, GraphComp, AttributeMask, GAE, EdgeMask, NodeProp, DisCluster, DGI and SubgCon) described in Table 1 as the base tasks. We trained 128-dimensional representations on Cora, CiteSeer, PubMed, Amazon-Computers and Amazon-Photo, and 32-dimensional representations on ogbn-arxiv.

Implementations. We perform five rounds of experiments with different random seeds for each experimental setting and report the average results. Further details of the experiments can be found in the appendix.

4.2. Capturing the Task Correlations

Setup. We initially obtain representations trained for each self-supervised task on the dataset. Subsequently, we calculate the correlation values between all tasks according to Definition 3.1, resulting in an 8×8 correlation value matrix. This matrix serves as the training target for a GraphTCM model. During the training process, we randomly partition 70% of the correlation values as the training set and the remaining as the validation set. Finally, we select the model with the lowest error on the validation set as the final model. Furthermore, we introduce three new self-supervised tasks (PairAttSim (Jin et al., 2020), GRACE (Zhu et al., 2020a), and GraphMAE (Hou et al., 2022a)) that are not involved during training to validate the generalization capability of the trained GraphTCM. Specifically, we calculate the correlation values of these three tasks on the eight base tasks and compare them with the predicted values from a well-trained GraphTCM.

Experimental Results. Figure 2 illustrates the training results of GraphTCM on three datasets (Cora, CiteSeer, and PubMed). Despite using only 70% of the correlation values for training, GraphTCM achieves an average error of 6.04% on the training set and 8.97% on the validation set. Figure 3 demonstrates the generalization results for the three new tasks. When using a trained GraphTCM to predict the correlation values of new self-supervised tasks on the base tasks, our method achieves an average error of

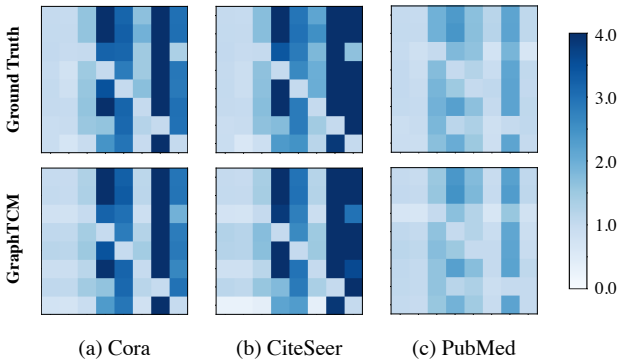

Figure 2. The modeling capability of GraphTCM. The top plots display real correlation value matrices, while the bottom plots exhibit reconstructed correlation value matrices from GraphTCM.

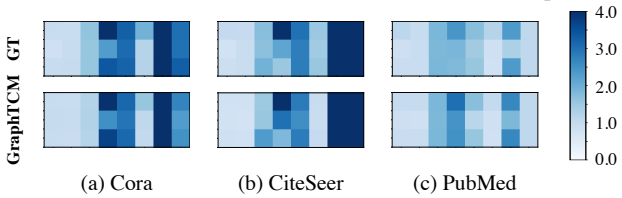

Figure 3. The generalization capability of GraphTCM. The top plots display real correlation value matrices of three unseen tasks (PairAttSim, GRACE and GraphMAE), while the bottom plots exhibit predicted correlation value matrices from GraphTCM.

13.56%. For the majority of graph self-supervised tasks, there already exists a 5%-15% random perturbation (Liu et al., 2021). Therefore, the error rate of GraphTCM is considered acceptable.

4.3. Generating Capable Representations

Setup. Once we have trained a GraphTCM on a particular dataset, we can leverage it to train representations with high representation capability. According to Formula 9 and 10, GraphTCM can help to train models of arbitrary forms. In our experiments, we only train a linear transformation to obtain the GraphTCM-enhanced representations. Specifically, given a set of tasks $\mathcal{T} = \{t_1, t_2, ..., t_k\}$ with their trained representations $\mathcal{H} = \{\mathbf{H}_{t_1}, \mathbf{H}_{t_2}, ..., \mathbf{H}_{t_k}\} \in \mathbb{R}^{N \times d}$, we train a matrix $\mathbf{W} \in \mathbb{R}^{k \times d}$ and our GraphTCM-enhanced representations $\mathbf{H}'$ satisfy $\mathbf{H}'(a, b) = \sum_i^k \mathbf{H}_{t_i}(a, b) \cdot \mathbf{W}(i, b)$, where $\mathbf{H}(a, b)$ denotes the value at the a-th row and the b-th column in $\mathbf{H}$. During the training process, we optimize the parameters of $\mathbf{W}$ according to Formula 10.

Experimental Results. Figure 4 illustrates the comparison of ARL values for our method and base tasks. Our GraphTCM-enhanced representations achieve the lowest ARL values across all datasets. Specifically, on three datasets (Cora, CiteSeer and PubMed), the ARL values of the GraphTCM-enhanced representations decrease by an average of 8.42% compared to the lowest ARL values of the base tasks and decrease by an average of 26.53% compared

Table 5. The statistics of the downstream performance for trained representations of various graph self-supervised methods. The optimal results are in **bold**, while the sub-optimal results are underlined.

Method \ Task & Dataset	Node Classification (Accuracy %)						Link Prediction (ROC-AUC %)		
	Cora	CiteSeer	PubMed	ogbn-arxiv	A-Computers	A-Photo	Cora	CiteSeer	PubMed
GraphComp	62.04±0.60	62.74±0.65	73.92±0.31	51.10±0.25	81.81±0.43	88.32±0.40	87.54±1.40	82.50±1.84	90.95±1.35
AttributeMask	59.55±0.69	59.38±0.61	73.40±0.34	53.35±0.21	81.65±0.36	88.77±0.58	81.71±2.24	80.76±1.55	88.41±2.08
GAE	65.48±0.76	57.06±0.18	76.00±0.68	49.98±0.09	81.81±0.43	89.04±0.27	91.60±0.22	82.50±1.84	93.91±0.10
EdgeMask	63.10±1.01	53.78±0.40	70.54±0.67	54.13±0.24	82.51±0.28	88.29±0.40	92.88±0.24	90.07±0.64	95.61±0.09
NodeProp	34.10±2.72	31.58±1.58	58.22±1.73	52.14±0.15	81.08±0.38	86.77±0.35	83.31±1.80	84.49±0.91	93.12±0.67
DisCluster	64.97±1.40	62.18±1.12	70.74±0.43	53.33±0.15	81.62±0.61	88.65±0.33	84.71±1.84	86.02±2.25	91.93±1.04
DGI	76.92±0.42	67.01±0.38	76.96±0.24	50.97±0.13	81.66±0.50	88.53±0.50	92.04±2.07	89.46±1.30	95.00±0.04
SubgCon	77.28±0.15	68.57±0.51	76.76±0.56	52.56±0.30	80.25±0.31	87.33±0.08	91.29±0.17	91.23±0.29	91.29±0.14
Addition	74.12±0.47	67.86±0.48	75.64±0.93	52.16±0.24	83.57±0.23	90.02±0.24	92.54±0.13	92.29±0.29	94.92±0.14
Concat	76.30±0.37	67.92±0.54	76.74±0.22	53.88±0.44	82.43±0.38	91.07±0.42	92.86±0.04	92.59±0.16	94.44±0.10
Multi-loss	57.76±2.87	52.74±3.10	67.08±2.09	51.26±0.13	81.55±0.72	90.06±0.38	85.70±1.17	83.88±2.83	92.52±0.79
AutoSSL	76.64±0.71	70.30±0.46	75.78±0.69	53.45±0.23	83.42±0.30	90.70±0.41	92.00±3.66	92.01±0.04	96.55±0.36
ParetoGNN	52.03±0.30	59.73±0.50	76.54±0.50	53.84±0.26	79.26±0.26	84.50±0.53	77.29±5.20	87.09±4.65	96.06±0.18
GraphTCM (Ours)	**81.50**±0.52	**72.83**±0.64	**77.24**±0.48	**54.73**±0.18	**84.95**±0.34	**92.06**±0.19	**95.52**±0.15	**96.14**±0.24	**97.13**±0.06

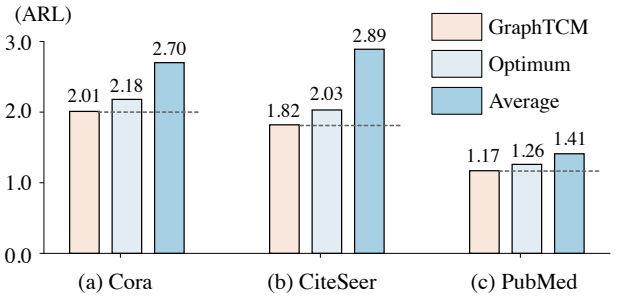

Figure 4. The statistics of ARL values for representations trained by GraphTCM and base tasks ("Optimum" represents the lowest ARL values among base tasks while "Average" represents the average ARL values for base tasks).

to the average ARL values of the base tasks. Our experimental results demonstrate the effectiveness of GraphTCM, indicating that we can obtain representations with high capability with the assistance of a trained GraphTCM.

4.4. Evaluating on Downstream Tasks

Setup. Next, we evaluate the performance of the trained representations on downstream tasks. During the downstream training, we fix the representations obtained from the self-supervised tasks and only train linear tuning heads to adapt the self-supervised representations to the downstream tasks. We conduct self-supervised training on the six datasets described in Section 4.1 and evaluate the performance of the trained representations for downstream node classification on these datasets. In addition, we select three datasets (Cora, CiteSeer and PubMed) to validate the ability of the self-supervised representations for link prediction. To further demonstrate the effectiveness of GraphTCM, we involve three multi-task methods (Multi-loss, AutoSSL, ParetoGNN) and two naive representation mixing methods (representation addition and concat) as our baselines. Their detailed descriptions can be found in Appendix A.2.

Experimental Results. Table 5 presents the comprehensive results for various graph self-supervised tasks. Our GraphTCM-enhanced representations achieve optimal performance across *all* datasets and downstream tasks. For node classification tasks, our GraphTCM-enhanced representations exhibit an average improvement of 1.7% compared to the sub-optimal self-supervised representations. Meanwhile, for link prediction, the average improvement is 2.2%. However, the performance of representations trained by multi-task methods falls far short of expectations. In most cases, these methods are inferior to concatenating trained representations from base tasks. Moreover, in some situations, the representations trained by these multi-task methods even exhibit significant performance degradation compared to the representations trained by single base tasks. In summary, our proposed GraphTCM successfully models the correlations between self-supervised tasks and can be used to train representations with high capability. On the downstream tasks, GraphTCM-enhanced representations demonstrate far superior effectiveness and universality compared to representations trained by other methods.

5. Conclusion

In this paper, we have unveiled the task correlations for graph self-supervised learning by calculating the correlation values between various graph self-supervised tasks. By demonstrating the actual correlation values between tasks across various datasets, we reveal the complexity of task correlations and the limitations of existing multi-task learning methods. To obtain more capable representations, we propose GraphTCM to model these task correlations. The experimental results indicate that GraphTCM can effectively characterize task correlations, and GraphTCM-enhanced representations have achieved outstanding performance on various downstream tasks.

Impact Statement

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

Acknowledgements

This work is supported by National Natural Science Foundation of China (No. 62322606, No. 62441605).

References

Fang, T., Xiao, Z., Wang, C., Xu, J., Yang, X., and Yang, Y. Dropmessage: Unifying random dropping for graph neural networks. In *AAAI Conference on Artificial Intelligence*, 2022.

Fang, T., Zhang, Y., Yang, Y., Wang, C., and Chen, L. Universal prompt tuning for graph neural networks. *Advances in Neural Information Processing Systems*, 36, 2024.

Feng, W., Zhang, J., Dong, Y., Han, Y., Luan, H., Xu, Q., Yang, Q., Kharlamov, E., and Tang, J. Graph random neural networks for semi-supervised learning on graphs. *NeurIPS*, 33, 2020.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *NeurIPS*, pp. 1024–1034, 2017.

Han, X., Huang, Z., An, B., and Bai, J. Adaptive transfer learning on graph neural networks. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021.

Hasanzadeh, A., Hajiramezanali, E., Duffield, N. G., Narayanan, K. R., Zhou, M., and Qian, X. Semi-implicit graph variational auto-encoders. *ArXiv*, abs/1908.07078, 2019.

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. Learning deep representations by mutual information estimation and maximization. *ICLR*, 2019.

Hou, Z., Liu, X., Cen, Y., Dong, Y., Yang, H., Wang, C., and Tang, J. Graphmae: Self-supervised masked graph autoencoders. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022a.

Hou, Z., Liu, X., Cen, Y., Dong, Y., Yang, H., Wang, C., and Tang, J. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 594–604, 2022b.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020a.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V. S., and Leskovec, J. Strategies for pre-training graph neural networks. *ICLR*, 2020b.

Hu, Z., Fan, C., Chen, T., Chang, K.-W., and Sun, Y. Pre-training graph neural networks for generic structural feature extraction. *ArXiv*, abs/1905.13728, 2019.

Hu, Z., Dong, Y., Wang, K., Chang, K.-W., and Sun, Y. Gpt-gnn: Generative pre-training of graph neural networks. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020c.

Huang, R., Xu, J., Jiang, X., Pan, C., Yang, Z., Wang, C., and Yang, Y. Measuring task similarity and its implication in fine-tuning graph neural networks. In *AAAI Conference on Artificial Intelligence*, 2024.

Jiang, B., Zhang, Z., Lin, D., Tang, J., and Luo, B. Semi-supervised learning with graph learning-convolutional networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11305–11312, 2019.

Jiao, Y., Xiong, Y., Zhang, J., Zhang, Y., Zhang, T., and Zhu, Y. Sub-graph contrast for scalable self-supervised graph representation learning. In *2020 IEEE international conference on data mining (ICDM)*, pp. 222–231. IEEE, 2020.

Jin, W., Derr, T., Liu, H., Wang, Y., Wang, S., Liu, Z., and Tang, J. Self-supervised learning on graphs: Deep insights and new direction. *ArXiv*, abs/2006.10141, 2020.

Jin, W., Liu, X., Zhao, X., Ma, Y., Shah, N., and Tang, J. Automated self-supervised learning for graphs. *ICLR*, 2022.

Jovanovi'c, N., Meng, Z., Faber, L., and Wattenhofer, R. Towards robust graph contrastive learning. *ArXiv*, abs/2102.13085, 2021.

Ju, M., Zhao, T., Wen, Q., Yu, W., Shah, N., Ye, Y., and Zhang, C. Multi-task self-supervised graph neural networks enable stronger task generalization. *ICLR*, 2023.

Kang, Z., Lin, Z., Zhu, X., and Xu, W. Structured graph learning for scalable subspace clustering: From single view to multiview. *IEEE Transactions on Cybernetics*, 52 (9):8976–8986, 2021.

Kim, D. and Oh, A. H. How to find your friendly neighborhood: Graph attention design with self-supervision. *ArXiv*, abs/2204.04879, 2022.

Kipf, T. and Welling, M. Variational graph auto-encoders. *ArXiv*, abs/1611.07308, 2016a.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. In *ArXiv*, volume abs/1611.07308, 2016b.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Li, X., Ye, T., Shan, C., Li, D., and Gao, M. Seegera: Self-supervised semi-implicit graph variational auto-encoders with masking. In *Proceedings of the ACM Web Conference 2023*, pp. 143–153, 2023.

Liu, Y., Pan, S., Jin, M., Zhou, C., Xia, F., and Yu, P. S. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35:5879–5900, 2021.

Manessi, F. and Rozza, A. Graph-based neural network models with multiple self-supervised auxiliary tasks. *Pattern Recognit. Lett.*, 148:15–21, 2020.

Pan, S., Hu, R., Long, G., Jiang, J., Yao, L., and Zhang, C. Adversarially regularized graph autoencoder. *ArXiv*, abs/1802.04407, 2018.

Park, J., Lee, M., Chang, H. J., Lee, K., and Choi, J. Y. Symmetric graph convolutional autoencoder for unsupervised graph representation learning. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6518–6527, 2019.

Peng, Z., Dong, Y., Luo, M., Wu, X.-M., and Zheng, Q. Self-supervised graph representation learning via global context prediction. *ArXiv*, abs/2003.01604, 2020.

Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding, M., Wang, K., and Tang, J. Gcc: Graph contrastive coding for graph neural network pre-training. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

Ramaswamy, L., Gedik, B., and Liu, L. A distributed approach to node clustering in decentralized peer-to-peer networks. In *TPDS*, volume 16, pp. 814–829, 2005.

Rong, Y., Huang, W., Xu, T., and Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. In *ICLR*, 2019.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. volume 29, pp. 93–93, 2008.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. *ArXiv*, abs/1811.05868, 2018.

Sun, K., Zhu, Z., and Lin, Z. Multi-stage self-supervised learning for graph convolutional networks. *ArXiv*, abs/1902.11038, 2019.

Velickovic, P., Fedus, W., Hamilton, W. L., Lio', P., Bengio, Y., and Hjelm, R. D. Deep graph infomax. *ICLR*, 2019a.

Velickovic, P., Fedus, W., Hamilton, W. L., Lio', P., Bengio, Y., and Hjelm, R. D. Deep graph infomax. *ICLR*, 2019b.

Wang, C., Pan, S., Long, G., Zhu, X., and Jiang, J. Mgae: Marginalized graph autoencoder for graph clustering. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017.

Wang, X. and Qi, G.-J. Contrastive learning with stronger augmentations. *IEEE transactions on pattern analysis and machine intelligence*, 45(5):5549–5560, 2022.

Wu, L., Lin, H., Gao, Z., Tan, C., and Stan.Z.Li. Self-supervised learning on graphs: Contrastive, generative, or predictive. *IEEE Transactions on Knowledge and Data Engineering*, 35:4216–4235, 2021.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32:4–24, 2019.

Xie, Y., Xu, Z., Wang, Z., and Ji, S. Self-supervised learning of graph neural networks: A unified review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45: 2412–2429, 2021.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *ICLR*, 2019.

Xu, X., Zhou, F., Zhang, K., and Liu, S. Ccgl: Contrastive cascade graph learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):4539–4554, 2022.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In *SIGKDD*, pp. 974–983, 2018.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. Graph contrastive learning with augmentations. *NeurIPS*, 2020a.

You, Y., Chen, T., Wang, Z., and Shen, Y. When does self-supervision help graph convolutional networks? *Proceedings of machine learning research*, 119:10871–10880, 2020b.

Zamir, A. R., Sax, A., Shen, B. W., Guibas, L. J., Malik, J., and Savarese, S. Taskonomy: Disentangling task transfer learning. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3712–3722, 2018.

Zhang, Z., Xu, M., Jamasb, A., Chenthamarakshan, V., Lozano, A., Das, P., and Tang, J. Protein representation learning by geometric structure pretraining. *ICLR*, 2023.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. Deep graph contrastive representation learning. *ArXiv*, abs/2006.04131, 2020a.

Zhu, Y., Xu, Y., Yu, F., Wu, S., and Wang, L. Cagnn: Cluster-aware graph neural networks for unsupervised graph representation learning. *ArXiv*, abs/2009.01674, 2020b.

Zitnik, M., Sosi, R., and Leskovec, J. Prioritizing network communities. *Nature Communications*, 9, 2018.

A. Extra Materials for Section 3

A.1. The Detailed Descriptions of Involved Self-supervised Tasks

In our pursuit of a comprehensive understanding of the intricate relationships within graph self-supervised tasks, we have carefully selected two exemplary self-supervised tasks for each category. Detailed introductions to these tasks are provided as below.

- *GraphComp* (You et al., 2020b). Analogous to image completion, GraphComp involves masking the features of specific target nodes, which are then input into the Graph Convolutional Networks (GCNs). The objective is to reconstruct the masked features, teaching the network to extract features from the context, thereby classifying it as a typical feature-based (FB) category.
- *AttributeMask* (Jin et al., 2020). AttributeMask aims to reconstruct the dense feature matrix generated by Principal Component Analysis (PCA) rather than the raw features, addressing challenges associated with reconstructing high-dimensional and sparse feature representations.
- *GAE* (Kipf & Welling, 2016a). The graph autoencoder (GAE) is a straightforward example of a structure generation method. Within the GAE framework, a GCN encoder initially generates node representations from the original graph. Subsequently, an inner production function with a sigmoid activation function acts as the decoder, aiming to reconstruct the adjacency matrix using the node representations.
- *EdgeMask* (Jin et al., 2020). The EdgeMask task aims to acquire finer-grained local structural information by employing link prediction as a pretext task. It selectively masks certain edge connections, and the primary learning objective revolves around predicting the presence or absence of a link between a specified node pair.
- *NodeProp* (Jin et al., 2020). NodeProp utilizes a node-level pretext task, predicting properties for individual nodes, including attributes such as degree, local node importance, and local clustering coefficient.
- *DisCluster* (Jin et al., 2020). DisCluster's objective is to perform regression on the distances between each node and predefined graph clusters. It partitions the graph into multiple clusters and designates the node with the highest degree within each cluster as its cluster center. The task is to predict the distances from each node to all cluster centers.
- *DGI* (Velickovic et al., 2019a). DGI maximizes mutual information between representations from subgraphs with differing scales, facilitating the graph encoder in attaining a comprehensive grasp of both localized and global semantic information.
- *SubgCon* (Jiao et al., 2020). Sub-graph Contrast (SubgCon) captures regional structural insights by capitalizing on the robust correlation between central nodes and their sampled subgraphs. It fosters the learning of node representations via a contrastive loss defined using subgraphs sampled from the original graph, adopting an innovative data augmentation strategy.

The training objectives of the self-supervised methods we have chosen are well-defined and singular, widely applicable across diverse graph self-supervised scenarios. These methods are highly representative, and consequently, the majority of existing graph self-supervised methods can be broken down into combinations of the aforementioned self-supervised tasks.

A.2. The Detailed Descriptions of Involved Multi-task Training Methods

Given a set of base graph self-supervised tasks $\mathcal{T} = \{t_1, t_2, ..., t_k\}$ with their trained representations $\mathcal{H} = \{\mathbf{H}_{t_1}, \mathbf{H}_{t_2}, ..., \mathbf{H}_{t_k}\}$ and losses $\mathcal{L} = \{\boldsymbol{l}_{t_1}, \boldsymbol{l}_{t_2}, ..., \boldsymbol{l}_{t_k}\}$, these methods can be expressed as below:

- *Addition, Concat.* Representation addition sums all the trained representations to obtain $\mathbf{H}_{\text{add}} = \sum_i^k \mathbf{H}_{t_i}$ while representation concat concatenates all the trained representations to obtain $\mathbf{H}_{\text{concat}} = [\mathbf{H}_{t_1}|\mathbf{H}_{t_2}|...|\mathbf{H}_{t_k}]$.
- *Multi-loss.* These methods combine the losses from multiple tasks for self-supervised training. Multi-loss combines the losses from different tasks with weighted summation $\boldsymbol{l} = \sum_i^k \alpha_i \boldsymbol{l}_{t_i}$, where $\alpha_1, \alpha_2, ..., \alpha_k$ are learnable task weight coefficients.

- *AutoSSL.* AutoSSL (Jin et al., 2022) can automatically search over combinations of various self-supervised tasks using their pseudo-homophily. Specifically, its optimization objective is defined as $\min_{\lambda_1,\ldots,\lambda_k} \mathcal{H}(\theta^*), s.t. \theta^* = \arg\min_{\theta} \mathcal{L}(\{\theta_i\},\{\alpha_i\},\{l_i\}) = \arg\min_{\theta_i} \sum_{i=1}^{k} \alpha_i l_i$, where H represents an unsupervised quality metric employed to assess the quality of the acquired embedding. α_i ,λ_i and θ correspond to the underlying self-supervised task ,their respective weights and the learnable parameters of neural networks.
- *ParetoGNN.* ParetoGNN (Ju et al., 2023) dynamically allocates task weights to facilitate Pareto optimality. Specifically, within the combination of self-supervised tasks, it is formulated as $\boldsymbol{l} = \sum_{i}^{k} \alpha_i \boldsymbol{l}_{t_i}$, where α_i represents the task weight coefficient.

A.3. What Makes Better Representations

Under ideal conditions, our objective is to acquire representations through self-supervised training that demonstrate low correlation values across all self-supervised tasks. Naturally, representations trained with minimal correlation across these tasks should possess the expressive capacity necessary for all self-supervised tasks. Typically, such trained representations will also display strong generality and deliver high performance in downstream tasks. Therefore, it is entirely reasonable for us to establish high representation capability as our training goal. In Section 3.5, we theoretically analyze the relationship between representation capability and downstream performance. Our theoretical findings align with our intuition, indicating that high representation capability significantly contributes to enhanced performance in downstream tasks. Thus, in comparison to task difficulty, our primary focus lies in the representation capability of the task, as it directly impacts the performance of the trained representation in downstream tasks.

A.4. Task Correlations Within the Same Category and Across Different Categories

Intuitively, graph self-supervised tasks within the same category should exhibit stronger mutual correlations compared to tasks from different categories. However, our experiments do not consistently support this notion; only structure-based graph self-supervised tasks demonstrate such a tendency. Feature-based tasks, on the other hand, tend to have relatively lower task difficulty, indicating that other self-supervised tasks have smaller correlation values with feature-based tasks. Consequently, the correlation between different feature-based tasks is not notably strong. Similarly, auxiliary property-based tasks often show significant variations in their practical implementations, with no explicit connection between tasks such as node degree prediction and node clustering. Hence, different auxiliary property-based tasks also do not demonstrate stronger task correlations between them. The same observation applies to contrast-based tasks, where different tasks have notably different objectives. Conversely, structure-based tasks typically center on graph structure reconstruction, leading to stronger correlations among tasks in this category. In general, self-supervised tasks with closely aligned objectives should exhibit stronger task correlations. However, it is crucial to note that not all tasks within the same category have closely aligned objectives.

A.5. Proof of Theorem 3.4

Theorem 3.4. *Given two task $t_1, t_2 \in \mathcal{T}$ and their trained representations $\mathbf{H}_{t_1}, \mathbf{H}_{t_2}$, the error of $\mathbf{H}_{t_2}$ on the downstream task is e_{t_2}, which can be expressed as:*

$$e_{t_2} = \min_{\mathbf{W}^*_{t_2}} \|\mathbf{H}_{t_2} \cdot \mathbf{W}^*_{t_2} - \mathbf{Y}_{t_{ds}}\|$$

Then, the error of $\mathbf{H}_{t_1}$ on the downstream task satisfies:

$$e_{t_1} \leq \text{Cor}(t_1, t_2) \cdot (e_{t_2} + \triangle) + \|\mathbf{Y}_{t_2} - \mathbf{Y}_{t_{ds}}\|$$

where $\triangle = \|\mathbf{Y}_{t_2} - \mathbf{Y}_{t_{ds}}\| + \|\mathbf{H}_{t_2} \cdot (\hat{\mathbf{W}}^*_{t_2} - \hat{\mathbf{W}}_{t_2})\|$, and $\hat{\mathbf{W}}^*_{t_2}, \hat{\mathbf{W}}_{t_2}$ denote the optimal heads for $\mathbf{W}^*_{t_2}$ and $\mathbf{W}_{t_2}$.

Proof. According to Formula 12, the correlation value $\text{Cor}(t_1, t_2)$ can be expressed as:

$$\text{Cor}(t_1, t_2) = \frac{\min_{\mathbf{W}_{t_1}} \|\mathbf{H}_{t_1} \cdot \mathbf{W}_{t_1} - \mathbf{Y}_{t_2}\|}{\min_{\mathbf{W}_{t_2}} \|\mathbf{H}_{t_2} \cdot \mathbf{W}_{t_2} - \mathbf{Y}_{t_2}\|} \tag{16}$$

We assume that for task t_2, the optimal heads for $\mathbf{H}_{t_1}$ and $\mathbf{H}_{t_2}$ are $\hat{\mathbf{W}}_{t_1}$ and $\hat{\mathbf{W}}_{t_2}$. Meanwhile, the optimal head for $\mathbf{H}_{t_2}$ on downstream task is $\hat{\mathbf{W}}^*_{t_2}$:

$$\hat{\mathbf{W}}_{t_1} = \arg\min_{\mathbf{W}_{t_1}} \|\mathbf{H}_{t_1} \cdot \mathbf{W}_{t_1} - \mathbf{Y}_{t_2}\| \tag{17}$$

$$\hat{\mathbf{W}}_{t_2} = \arg\min_{\mathbf{W}_{t_2}} \|\mathbf{H}_{t_2} \cdot \mathbf{W}_{t_2} - \mathbf{Y}_{t_2}\| \tag{18}$$

$$\hat{\mathbf{W}}^*_{t_2} = \arg\min_{\mathbf{W}^*_{t_2}} \|\mathbf{H}_{t_2} \cdot \mathbf{W}^*_{t_2} - \mathbf{Y}_{t_{ds}}\| \tag{19}$$

Then, the error of $\mathbf{H}_{t_1}$ on the downstream tasks can be calculated as follows:

$$\begin{aligned}
e_{t_1} &= \min_{\mathbf{W}^*_{t_1}} \|\mathbf{H}_{t_1} \cdot \mathbf{W}^*_{t_1} - \mathbf{Y}_{t_{ds}}\| && (20)\\
&= \min_{\mathbf{W}^*_{t_1}} \|\mathbf{H}_{t_1} \cdot \mathbf{W}^*_{t_1} - \mathbf{Y}_{t_2} + \mathbf{Y}_{t_2} - \mathbf{Y}_{t_{ds}}\| && (21)\\
&\leq \min_{\mathbf{W}^*_{t_1}} \|\mathbf{H}_{t_1} \cdot \mathbf{W}^*_{t_1} - \mathbf{Y}_{t_2}\| + \|\mathbf{Y}_{t_2} - \mathbf{Y}_{t_{ds}}\| && (22)\\
&= \|\mathbf{H}_{t_1} \cdot \hat{\mathbf{W}}_{t_1} - \mathbf{Y}_{t_2}\| + \|\mathbf{Y}_{t_2} - \mathbf{Y}_{t_{ds}}\| && (23)\\
&= \mathrm{Cor}(t_1, t_2) \cdot \|\mathbf{H}_{t_2} \cdot \hat{\mathbf{W}}_{t_2} - \mathbf{Y}_{t_2}\| + \|\mathbf{Y}_{t_2} - \mathbf{Y}_{t_{ds}}\| && (24)\\
&= \mathrm{Cor}(t_1, t_2) \cdot \|\mathbf{H}_{t_2} \cdot \hat{\mathbf{W}}^*_{t_2} - \mathbf{Y}_{t_{ds}} + \mathbf{H}_{t_2} \cdot (\hat{\mathbf{W}}_{t_2} - \hat{\mathbf{W}}^*_{t_2}) + \mathbf{Y}_{t_{ds}} - \mathbf{Y}_{t_2}\| + \|\mathbf{Y}_{t_2} - \mathbf{Y}_{t_{ds}}\| && (25)\\
&\leq \mathrm{Cor}(t_1, t_2) \cdot (\|\mathbf{H}_{t_2} \cdot \hat{\mathbf{W}}^*_{t_2} - \mathbf{Y}_{t_{ds}}\| + \|\mathbf{H}_{t_2} \cdot (\hat{\mathbf{W}}_{t_2} - \hat{\mathbf{W}}^*_{t_2})\| + \|\mathbf{Y}_{t_{ds}} - \mathbf{Y}_{t_2}\|) + \|\mathbf{Y}_{t_2} - \mathbf{Y}_{t_{ds}}\| && (26)\\
&= \mathrm{Cor}(t_1, t_2) \cdot (e_{t_2} + \|\mathbf{H}_{t_2} \cdot (\hat{\mathbf{W}}_{t_2} - \hat{\mathbf{W}}^*_{t_2})\| + \|\mathbf{Y}_{t_{ds}} - \mathbf{Y}_{t_2}\|) + \|\mathbf{Y}_{t_2} - \mathbf{Y}_{t_{ds}}\| && (27)
\end{aligned}$$

If we denote $\triangle = \|\mathbf{Y}_{t_2} - \mathbf{Y}_{t_{ds}}\| + \|\mathbf{H}_{t_2} \cdot (\hat{\mathbf{W}}^*_{t_2} - \hat{\mathbf{W}}_{t_2})\|$, then we have:

$$e_{t_1} \leq \mathrm{Cor}(t_1, t_2) \cdot (e_{t_2} + \triangle) + \|\mathbf{Y}_{t_2} - \mathbf{Y}_{t_{ds}}\| \tag{28}$$

Thus, we have proved the correctness of Theorem 3.4.

□

A.6. Proof of Theorem 3.5

Theorem 3.5. *Given a set of tasks $\mathcal{T} = \{t_1, t_2, ..., t_k\}$ with their trained representations $\mathcal{H} = \{\mathbf{H}_{t_1}, \mathbf{H}_{t_2}, ..., \mathbf{H}_{t_k}\}$, for any downstream task t_{ds}, there exists a $\beta \geq \|\mathbf{Y}_{t_i} - \mathbf{Y}_{t_{ds}}\| + \|\mathbf{H}_{t_i} \cdot (\hat{\mathbf{W}}^*_{t_i} - \hat{\mathbf{W}}_{t_i})\|$ for any $t_i \in \mathcal{T}$. If a new task t' and its trained representations $\mathbf{H}_{t'}$ that satisfies $\delta \geq \mathrm{Cor}(t', t_i)$ for any $t_i \in \mathcal{T}$, the error of $\mathbf{H}_{t'}$ on downstream task holds:*

$$e_{t'} \leq \delta \cdot (e_{\min} + \beta) + \beta$$

where $e_{\min} = \min\{e_{t_1}, e_{t_2}, ..., e_{t_k}\}$.

Proof. According to Formula 27, for any task $t_i \in \mathcal{T}$, we can obtain the following inequality:

$$e_{t'} \leq \mathrm{Cor}(t', t_i) \cdot (e_{t_i} + \|\mathbf{H}_{t_i} \cdot (\hat{\mathbf{W}}_{t_i} - \hat{\mathbf{W}}^*_{t_i})\| + \|\mathbf{Y}_{t_{ds}} - \mathbf{Y}_{t_i}\|) + \|\mathbf{Y}_{t_i} - \mathbf{Y}_{t_{ds}}\| \tag{29}$$

We have $\forall t_i \in \mathcal{T}, \beta \geq \|\mathbf{Y}_{t_i} - \mathbf{Y}_{t_{ds}}\| + \|\mathbf{H}_{t_i} \cdot (\hat{\mathbf{W}}^*_{t_i} - \hat{\mathbf{W}}_{t_i})\|$. Formula 29 can be transformed into the following form:

$$\forall t_i \in \mathcal{T} \quad e_{t'} \leq \mathrm{Cor}(t', t_i) \cdot (e_{t_i} + \beta) + \beta \tag{30}$$

We also have $\forall t_i \in \mathcal{T}, \delta \geq \mathrm{Cor}(t', t_i)$. Then, we can obtain:

$$\forall t_i \in \mathcal{T} \quad e_{t'} \leq \delta \cdot (e_{t_i} + \beta) + \beta \tag{31}$$

Therefore, we can obtain the conclusion in Theorem 3.5:

$$e_{t'} \leq \delta \cdot (e_{\min} + \beta) + \beta \tag{32}$$

where $e_{\min} = \min\{e_{t_1}, e_{t_2}, ..., e_{t_k}\}$.

□

B. More Information on Experiments

B.1. Detailed Descriptions of Datasets

We employ 6 graph datasets in our experiments, including 4 citation network datasets *Cora, CiteSeer, PubMed, ogbn-arxiv* and 2 product network datasets *Amazon-Computers, Amazon-Photo*. Table 6 presents the statistics of involved datasets.

- *Cora, CiteSeer, PubMed, ogbn-arxiv:* These 4 different citation networks are widely used as graph benchmarks (Sen et al., 2008; Hu et al., 2020a). Their downstream task is to determine the research area of papers/researchers.

- *Amazon-Computers, Amazon-Photo:* These are 2 product networks (Shchur et al., 2018) from Amazon. The nodes here represent goods and edges represent that two goods are frequently bought together. The target is to map goods to their respective product category.

Table 6. The statistics of involved datasets.

Dataset	Nodes	Edges	Feature	Classes	Train/Val/Test
Cora	2708	5429	1433	7	140 / 500 / 1000
CiteSeer	3327	4732	3703	6	120 / 500 / 1000
PubMed	19717	44338	500	3	60 / 500 / 1000
ogbn-arxiv	169343	1166243	128	40	90941 / 29799 / 48603
Amazon-Computers	13752	491722	767	10	10% / 10% / 80%
Amazon-Photo	7650	238162	745	8	10% / 10% / 80%

B.2. Training the Self-supervised Representations

The hyperparameters for the aforementioned Self-supervised Tasks can be found in Table 7. All backbone models for these tasks have utilized the Graph Convolutional Network (GCN) with a linear architecture, and the parameter settings are largely consistent with the original paper. We trained 128-dimensional representations on Cora, CiteSeer, PubMed, Amazon-Computers, and Amazon-Photo, and 32-dimensional representations on ogbn-arxiv. Additionally, for the GraphComp model, the (learning rate, weight decay) values are set to (5e-4, 0.7) for the *Citeseer* dataset and (5e-4, 0.5) for the *Pubmed* dataset.

Table 7. The hyperparameters of training self-supervised representations. ES indicates the adoption of the early stopping strategy; # GCNs and # linears respectively denote the number of GCNs and linear layers in the self-supervised task.

	optimizer	learning rate	weight decay	dropout	epochs	# GCNs	# linears
GraphComp	Adam	0.008	8e-5	0.5	500	0	3
AttributeMask	Adam	0.001	5e-4	0.5	200	1	1
GAE	Adam	0.01	0	0	500	2	0
EdgeMask	Adam	0.001	5e-4	0.5	200	1	1
NodeProp	Adam	0.001	5e-4	0.5	200	1	1
DisCluster	Adam	0.001	5e-4	0.5	200	1	1
DGI	Adam	0.001	0	0	ES	1	1
SubgCon	Adam	0.001	0	0	50	1	0

B.3. Adapting to Downstream Tasks

When applying the trained representations to downstream tasks, we exclusively train a linear transformation $\mathbf{W} \in \mathbb{R}^{d \times d'}$, where d denotes the dimension of the trained representations, and d' denotes the dimension required for the downstream tasks. Across all datasets and methods, we maintain consistent hyperparameters, setting the learning rate to 0.001, weight decay to 0.0005, and conducting 300 training epochs.

B.4. Ablation Study

GraphTCM aims to estimate correlation values between various graph self-supervised tasks. To validate the effectiveness of each component, we designed three variants: GraphTCM w/o $\mathbf{W}_r$, GraphTCM w/o $\mathbf{W}_t$, and GraphTCM w/o Exp. Specifically, GraphTCM w/o $\mathbf{W}_r$ removes the learnable parameter $\mathbf{W}_r$, GraphTCM w/o $\mathbf{W}_t$ removes the learnable parameter$\mathbf{W}_t$, while GraphTCM w/o Exp removes the exponential function used in computing correlation values. Table 8 presents the average relative errors of each variant in characterizing the correlation values on three citation datasets: Cora, CiteSeer and PubMed.

Table 8. The average relative errors of each variant.

Model	Average training error (%)	Average validating error (%)
GraphTCM	6.04	8.97
GraphTCM w/o $\mathbf{W}_r$	83.44	105.36
GraphTCM w/o $\mathbf{W}_t$	91.76	128.30
GraphTCM w/o Exp	21.03	51.05