Structural Denoising Contrastive Self-supervised Graph Meta-learning

Ningbo Huang, Gang Zhou(✉), Meng Zhang, Shiyu Wang, and Shunhang Li

State Key Laboratory of Mathematical Engineering and Advanced Computing
rylynn_ab@163.com,zhougang_ieu@126.com,zhangmeng_ieu@sina.com,
share_wind@163.com,baikal_sh@163.com

Abstract. Graph meta-learning aims to extract cross-task transferable knowledge from meta-training tasks and improve the generalization capability on few-shot tasks. However, existing methods suffer from the heavy reliance on labeled data for meta task construction and limited graph data representation capability. To address these issues, we propose a both label-efficient and effective model, **CSG-Meta**[1]. Specifically, we first design a contrastive self-supervised meta tasks construction method to generate effective meta-training tasks without any labeled data. Then, we introduce a task-adaptive graph few-shot classifier to mitigate overfitting caused by limited labeled data and task divergence, which performs parameter initialization and transformation with prototype and task presentation. To reduce the noise in constructed tasks, we design a structural denoising module to measure the node's confidence score with structural similarity and node importance, leading to more accurate prototype and task representation. Experimental results on four real-world attributed graph datasets show that **CSG-Meta** achieves better node classification performance on most few-shot learning scenarios. On the Amazon-Clothing dataset, the improvement rates of accuracy are from 4.4% to 9.6% compared to the current state-of-the-art supervised graph meta-learning model COSMIC.

Keywords: meta-learning · few-shot learning · node classification · graph self-supervised learning· structural denoising.

1 Introduction

Graph deep learning techniques, such as graph neural networks (GNN) [5, 13, 3], have emerged by developing end-to-end neural network models. GNNs are capable of broadly supporting various graph analysis tasks, such as spammer detection[14], citation prediction[10], and product recommendation[7]. However, GNNs typically require large amounts of labeled data to effectively optimize the model's parameters. For many graph analysis tasks, obtaining sufficient labeled data remains a significant challenge. Consequently, graph few-shot learning, which aims to learn an effective GNN model from a limited amount of labeled data, has emerged as a key research focus within the field of graph learning.

[1] The source code is available on https://github.com/hningbo/CSG-Meta.

To address graph few-shot learning problems, graph meta-learning (GML) integrates GNN with meta-learning techniques to extract generalizable meta knowledge across graph learning tasks. Utilizing the meta knowledge, GML model can fast adapt to the novel classes with very few labeled data. Typically, Meta-GNN [21] and GPN [1] extend MAML [2] and prototypical network [9] respectively to extract cross-task structural knowledge from node representation encoded by GNNs. G-Meta [4] proposes to perform meta-training on the node's local subgraphs to address scalability and cross-graph generalization challenges. Besides, models such as AMM-GNN[15], TENT [16], and Meta-GPS [6] capture the relations and differences between tasks for better generalizing to target tasks with different structure and feature distribution. Despite these achievements, challenges still remain regarding the label reliance and effectiveness of GML. On the one hand, **existing methods heavily rely on labeled data to generate meta-training tasks**. The meta-training tasks are constructed from the based classes with sufficient labels, which is impractical in scenarios where only very limited labeled data for target classes is available. On the other hand, **these methods exhibit limited capacity in graph data representation**. GML aims to learn transferable knowledge across diverse tasks, and can only extract shallow features for better generalization ability.

To address the above challenges, we propose **CSG-Meta**, a structural denoising **C**ontrastive **S**elf-supervised **G**raph **Meta**-learning model. **CSG-Meta** can simultaneously learn effective and transferable node representation by meta-training on the tasks constructed without relying on any labeled data. Specifically, we first propose a contrastive self-supervised meta tasks construction method, which utilizes the intra-class similarities and inter-class differences learned by graph contrastive learning (GCL) as supervisory signals to construct effective meta-training tasks. Secondly, to alleviate the overfitting problems caused by the task divergences and limited data, we design a task-adaptive graph few-shot classifier. It utilizes the prototype and task representation to perform parameter initialization and transformation, which can provide prior knowledge of classes and task distribution for fast generalization. Then, to reduce noise in contrastive self-supervised tasks which could introduce bias to the prototype and task representation, we propose a structural denoising module to calculate the nodes confidence scores and generate denoised prototype and task representation. By performing meta-training on the contrastive self-supervised tasks, **CSG-Meta** can learn cross-task structural knowledge while aligning the representation space of the GML model with the GCL model, thereby achieving better prediction performance on graph few-shot learning tasks.

2 Proposed method: CSG-Meta

2.1 Problem formulation

In this paper, we focus on N-way K-shot node classification problem: in each task, there are N classes to classify and each class only contains K labeled nodes where K is typically a very small number (e.g. 1/3/5). Formally, we denote an

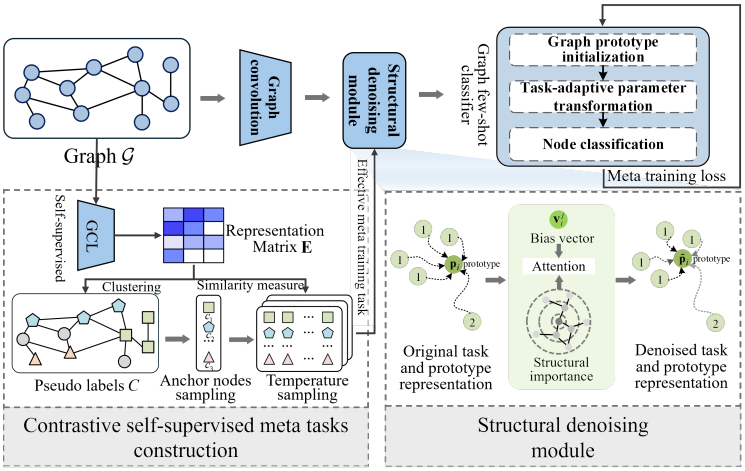

Fig. 1: Illustration of the **CSG-Meta**'s framework.

attributed graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ where $\mathcal{V} = \{v_1, v_2, ..., v_{|\mathcal{V}|}\}$ is the vertex set, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set, and $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{|\mathcal{V}|}]^T \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the node's attribute matrix. The class set $\mathcal{C} = \{c_1, c_2, ..., c_{|\mathcal{C}|}\}$ can be divided into based class set $\mathcal{C}_{\text{base}}$ and target class set $\mathcal{C}_{\text{target}}$, where class in $\mathcal{C}_{\text{base}}$ has sufficient labeled nodes but class in $\mathcal{C}_{\text{target}}$ only has K labeled data.

Graph meta-learning aims to learn effective and transferable knowledge from meta-training tasks $\mathcal{T} = \{\tau_1, \tau_2, ..., \tau_{|\mathcal{T}|}\}$, enabling fast generalization to target tasks τ_{T+1} with very few labeled data. In this paper, we aim to develop a **contrastive self-supervised GML** model, which utilizes the intra-class similarities and inter-class divergence of contrastive learning as supervisory signals to construct meta-training tasks without relying on $\mathcal{C}_{\text{base}}$.

2.2 Contrastive self-supervised meta task construction

We utilize the intra-class similarities and inter-class differences of node representation as supervised signals to construct meta-training tasks. Firstly we give two requirements for generating N-way K-shot tasks: 1) Sampling N anchor nodes from different natural classes. 2) For each anchor node, sampling $K-1$ nodes from the same class to construct support and query set $(\mathcal{S}, \mathcal{Q})$.

Through GCL algorithms such as GraphCL[20] and BGRL[12], we learn a distinguishable node representation: $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_{|\mathcal{V}|}]^T \in \mathbb{R}^{|\mathcal{V}| \times d}$ where $|\mathcal{V}|$ is the number of nodes in the graph, and d is the dimension of the node representation. Then, we explain how contrastive self-supervised method can satisfies the above two requirements.

Anchor node sampling For the first requirement, we leverage the characteristics of inter-class differences to design a clustering-based anchor node sampling method. Nodes that are far apart in the representation space tend to come

from different classes, so we can select nodes that are distant from each other as anchor nodes. Specifically, we use the distance-based clustering method K-means to partition nodes into clusters, where nodes in different clusters are far apart. Firstly, we apply K-means algorithm on the node representation space $\mathbf{E}$, partitioning the nodes into C distinct clusters where $C > N$, denoted as $\{c_1, c_2, \ldots, c_C\}$. Then, each node is assigned a pseudo-label according to the cluster it belongs. Finally, we randomly select N clusters and sample one node from each cluster to obtain N anchor nodes, denoted as $\mathcal{A} = \{v_1^0, v_2^0, ..., v_N^0\}$.

Meta tasks construction with similarity-based temperature sampling As for the second requirement, we utilize the intra-class similarities to sample nodes that are likely to share the same labels as the anchor nodes. Nodes that are close in representation space are likely to belong to the same class and thus be partitioned into the same cluster. Therefore, we can perform temperature sampling based on the similarity between candidate nodes and anchor nodes in GCL's representation space to construct support set $\mathcal{S}$ and query set $\mathcal{Q}$. Specifically, the nodes from the same cluster as anchor node v_i^0 are selected as candidate nodes, denoted as $c_i = \{v_{c_i}^1, v_{c_i}^2, ..., v_{c_i}^{|c_i|}\}$. Then, we calculate the sampling probability based on the similarity scores between the representation of anchor node and candidate nodes using temperature softmax: $p_{c_i}^j = \frac{\exp(\text{sim}(\mathbf{e}_i^0 \cdot \mathbf{e}_{c_i}^j)/t)}{\sum_{v_{c_i}^k \in c_i} \exp(\text{sim}(\mathbf{e}_i^0 \cdot \mathbf{e}_{c_i}^k)/t)}$, where $\mathbf{e}_i^0$ and $\mathbf{e}_{c_i}^j$ are the node representation of anchor node v_i^0 and candidate node $v_{c_i}^j$ respectively, $p_{c_i}^j$ is the probability that the j-th node in cluster c_i is sampled, and t is the sampling temperature. As t increases, the sampling strategy gradually approximates the random sampling from cluster c_i, resulting higher information entropy. Conversely, as t approaches 0, the sampling strategy converges the top-K matching. By sampling with appropriate temperature, we can introduce diversity in meta tasks generation, which can sufficiently describe and capture the relative node similarity in GCL's representation space. For each anchor node, by sampling $K + M - 1$ distinct candidate nodes according to above sampling strategies, we can acquire the support set $\mathcal{S}$ and query set $\mathcal{Q}$ as follow.

$$\begin{aligned} \mathcal{S} &= \{(v_1^0, c_1), (v_1^1, c_1), \ldots, (v_N^{N\times(K-1)}, c_N)\}, \\ \mathcal{Q} &= \{(v_1^K, c_1), (v_1^{K+1}, c_1), \ldots, (v_N^{N\times(K+M-1)}, c_N)\}. \end{aligned} \tag{1}$$

2.3 Task-adaptive graph few-shot classifier

The divergence of tasks and the extremely limited labeled data lead to the overfitting problems for graph few-shot learning[17]. To alleviate this issue, we design a task adaptive graph few-shot classifier.

GNN encoder We firstly adopt a simplifying graph convolution layer[19] to acquire the node representation as follow:

$$\mathbf{Z} = \mathbf{S}^K \mathbf{X} \mathbf{W}_Z, \tag{2}$$

where $\mathbf{S} = \tilde{\mathbf{D}}^{\frac{1}{2}} \mathbf{A} \tilde{\mathbf{D}}^{\frac{1}{2}}$, $\mathbf{A}, \tilde{\mathbf{D}} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ are the adjacency matrix and normalized degree matrix of graph $\mathcal{G}$, $\mathbf{W}_Z \in \mathbb{R}^{d\times d'}$ is the learnable transformation matrix,

and $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}| \times d'}$ is the acquired node representation. In avoidance of a trivial solution, it should be noticed that the GNN used for meta-training is a separate encoder distinct from the GNN in GCL. Finally, node classification is performed with a linear layer based on the node representation $\mathbf{Z}$:

$$\mathbf{o}_i^j = \text{softmax}(\mathbf{W}_L \cdot \mathbf{z}_i^j), \tag{3}$$

where $\mathbf{W}_L \in \mathbb{R}^{N \times d'}$ is the randomly initialized linear transformation matrix, $\mathbf{z}_i^j$ is the node representation of v_i^j in the support set, d is the dimension of representation, $o_i^j[k]$ indicates the probability that v_i^j belongs to the k-th category.

Graph prototype initialization Then, we use the prototype representation to adaptively initialize the model parameters, which can provide prior knowledge for the classifier, such as the similarities and differences between categories. Firstly, we calculate the prototype representation for each class as follow:

$$\mathbf{p}_i = \frac{1}{K} \sum_{j=1}^{K} \mathbf{z}_i^j, \tag{4}$$

where $\mathbf{z}_i^j$ is the representation of the j-th node from the i-th class in support set $\mathcal{S}$. We initialize the parameters with the prototype representation as follows:

$$\mathbf{W}_L = \mathbf{W}_P \cdot [\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_N]^T, \tag{5}$$

where $\mathbf{W}_P \in \mathbb{R}^{N \times N}$ is a learnable matrix. The prototype of each class provides prior knowledge to the linear classification layer, which constrains the parameter space of the linear classifier to a more accurate subspace. By optimizing parameters on the constrained parameter space, the classifier can better generalize to the novel classes with very few labeled nodes.

Task-adaptive parameter transformation Furthermore, to capture the distribution divergence across different tasks, we design a task-adaptive parameter transformation module. This module adaptively maps the model parameters to a task-specific parameter space based on the distribution of each task. Specifically, we represent the task τ as the average of the node representations in the support set: $\mathbf{t}_\tau = \frac{1}{N \times K} \sum_{i=1}^{N} \sum_{j=1}^{K} \mathbf{z}_i^j$. The divergence in structure and node features across tasks can be captured through $\mathbf{t}_\tau$, based on which we learn the parameter transformation coefficients λ_τ and μ_τ with MLP as follows:

$$\begin{aligned} \lambda_\tau &= \text{MLP}(\mathbf{t}_\tau, \theta_\lambda) \\ \mu_\tau &= \text{MLP}(\mathbf{t}_\tau, \theta_\mu). \end{aligned} \tag{6}$$

where λ_τ and μ_τ have the same shape as $\mathbf{W}_P$. We obtain the task-specific parameters with an affine transformation $\mathbf{W}_{P,\tau} = \lambda_\tau \odot \mathbf{W}_P + \mu_\tau$, where $\odot$ denotes the element-wise multiplication. This transformation can align the divergence on the distribution of representation across different tasks. Based on the prototype initialized parameters, we calculate the task-adaptive parameters as follow:

$$\mathbf{W}_{L,\tau} = \mathbf{W}_{P,\tau} \cdot [\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_N]^T. \tag{7}$$

The $\mathbf{W}_{L,\tau}$ is firstly initialized with the graph prototype to acquire prior knowledge of similarities and differences between categories, and then an affine transformation is performed to align the distribution in target task. With the task-adaptive graph few-shot classifier, we can perform node classification as follow:

$$\mathbf{o}_i^j = \text{softmax}(\mathbf{W}_{L,\tau} \cdot \mathbf{z}_i^j). \tag{8}$$

2.4 Structural denoising module

In the contrastive self-supervised meta tasks, not all the nodes in support set have the same ground-truth label as the anchor node, which will introduce bias into the prototype and task representation. Therefore, we design a structural denoising module, which utilizes the structural similarity and node importance to estimate the confidence scores of nodes in the constructed tasks.

Specifically, we firstly calculate bias vectors for support set nodes to measure their degree of bias from the prototype representation as follow.

$$\mathbf{v}_i^j = \mathbf{z}_i^j - \mathbf{p}_i, \tag{9}$$

where $\mathbf{z}_i^j$ is the representation of j-th node from the i-th class in the support set, $\mathbf{p}_i$ is the prototype representation of the i-th class, $\mathbf{v}_i^j$ represents the bias degree of node relative to its prototype. A larger bias indicates that the node is less likely to belong to the same class with other nodes, and therefore, should contributes smaller weight to the prototype and task representation. Furthermore, we also consider the structural importance to acquire more accurate node confidence scores. Nodes with higher importance contain richer structural information and can learn more accurate node representation[1]. Finally, we design a structural attention module to adaptively calculate the confidence scores according to the structural importance and representation bias:

$$\alpha_i^j = \frac{\exp(\log(d_i^j) \cdot \text{LeakyReLU}({\mathbf{v}_i^0}^T \mathbf{W}_D \mathbf{v}_i^j)}{\sum_{v_i^k \in \mathcal{S}_i} \exp(\log(d_i^k) \cdot \text{LeakyReLU}({\mathbf{v}_i^0}^T \mathbf{W}_D \mathbf{v}_i^k)}, \tag{10}$$

where d_i^j is the degree of v_i^j, and $\mathbf{W}_D \in \mathbb{R}^{d' \times d'}$ is the learnable parameters of the attention module. It first learns to estimate the nodes confidence score by utilizing nodes' bias vectors and their similarity to the bias vectors of corresponding anchor nodes, then reweights the scores with logarithmic node importance. Finally, we calculate the denoised prototype and task representations as follows:

$$\begin{aligned} \tilde{\mathbf{p}}_i &= \sum_{k=1}^{K} \alpha_i^k \mathbf{z}_i^k, \\ \tilde{\mathbf{t}}_i &= \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} \alpha_i^k \mathbf{z}_i^k. \end{aligned} \tag{11}$$

We initialize the model parameters in Eq. 5 with the denoised prototype representation and learn the affine transformation coefficients in Eq. 6 with the

denoised task representation. The structural denoising module reduces the discrepancy between contrastive self-supervised tasks and supervised tasks, thereby providing more accurate prior knowledge to perform meta-training.

3 Experiments

3.1 Experimental Settings

Datasets. We adopt four attributed graph datasets including citation network DBLP [11](40,672 nodes, 288,170 edges) and Cora-Full [5] (19,793 nodes, 65,311 edges), as well as product network Amazon-Clothing (24,919 nodes, 91,680 edges) and Amazon-Electronics [7] (42,318 nodes and 43,556 edges). Besides DBLP and Amazon-Electronics are datasets with relatively low graph homophily of 0.29 and 0.39. For Amazon-Clothing and Cora-Full, the homophily are 0.62 and 0.59. The graph homophily is the ratio of edges connected by nodes with the same labels.

Baselines. We adopt three categories of baseline models for evaluation: **(1) Unsupervised graph learning**: Deepwalk[8] and BGRL[12]. **(2) Supervised graph learning**: GCN[5] and SGC[19]. **(3) Supervised graph meta-learning**: G-Meta[4], TENT[16], Meta-GPS[6], and COSMIC[18].

Metric and experiment settings. Following recent work on graph few-shot learning[6], we use accuracy (ACC) as the metric to evaluate the performance on 5-way 3-shot, 5-way 5-shot, 10-way 3-shot, and 10-way 5-shot node classification problems. For supervised meta-learning models, we follow the strategy in Meta-GPS to divide the classes into training, validation, and testing sets.

For the detailed experiment settings, we implement the pretraining and meta-learning models with Pytorch and DGL, and optimize the model parameters with Adam optimizer. In the pretraining phrase, we use BGRL[12] algorithm to pretrain the GNN and acquire a node representation with a dimension of 128. The number of epoch is set to 300. During meta-training, the learning rates for the inner and outer loop are set to 0.5 and 0.003 respectively. Each task is iterated for 10 steps during meta-training and 20 steps during meta-testing.

3.2 Overall results

From the results shown in the table 1, we can draw following conclusions:

- Generally, **CSG-Meta** achieves the best performance across almost all the datasets and problem settings compared with the baseline models. Even without any labeled data for meta-training, **CSG-Meta** still achieve improvements of 4.4∼9.6%, 0.8∼1.6%, 0.8∼1.6%, 3.1∼5.0% on Amazon-Clothing, Amazon-Electronics, DBLP, and Cora-Full datasets, compared to the SOTA supervised GML model COSMIC. Although COSMIC also incorporates contrastive learning, it heavily relies on labeled data within the meta-training tasks to construct a supervised contrastive learning objective. Furthermore, COSMIC cannot capture the divergence across tasks, whereas **CSG-Meta** can model this divergence using the task-adaptive graph few-shot classifier.

Table 1: The comparison on few-shot node classification results

Methods	Amazon-Clothing				Amazon-Electronics			
	5-way 3-shot	5-way 5-shot	10-way 3-shot	10-way 5-shot	5-way 3-shot	5-way 5-shot	10-way 3-shot	10-way 5-shot
DeepWalk	36.7	46.5	21.3	35.3	23.5	26.1	14.7	16.0
GCN	54.3	59.3	41.3	44.8	53.8	59.6	42.3	47.4
SGC	56.8	62.2	43.1	46.3	54.6	60.8	43.2	50.0
G-Meta	72.6	73.5	58.3	62.1	73.2	76.6	66.3	68.0
TENT	77.6	79.2	71.1	72.3	75.8	79.4	67.6	69.7
Meta-GPS	80.3	82.6	73.1	74.1	82.8	85.2	78.5	79.3
COSMIC	83.9	85.3	75.9	77.9	84.4	87.1	78.3	79.1
BGRL	82.6	85.6	73.4	75.9	83.4	86.6	76.8	**82.5**
CSG-Meta	**87.2**	**89.7**	**79.3**	**85.4**	**86.1**	**87.8**	**79.6**	80.6
Methods	DBLP				Cora-Full			
	5-way 3-shot	5-way 5-shot	10-way 3-shot	10-way 5-shot	5-way 3-shot	5-way 5-shot	10-way 3-shot	10-way 5-shot
DeepWalk	44.7	62.4	33.8	45.1	24.9	27.9	16.7	19.2
GCN	59.6	68.3	43.9	51.2	34.1	39.4	25.6	30.2
SGC	57.3	65.0	40.2	50.3	37.5	41.2	29.4	33.7
G-Meta	75.1	77.6	63.2	64.2	57.5	62.4	53.9	58.1
TENT	79.0	82.8	65.4	72.3	64.8	69.2	51.7	56.0
Meta-GPS	83.9	85.5	75.9	77.9	65.1	69.2	61.2	64.2
COSMIC	83.6	85.3	75.6	77.3	69.3	75.8	65.5	69.2
BGRL	80.8	83.9	70.6	73.4	76.2	82.1	66.3	72.3
CSG-Meta	**84.3**	**87.3**	**76.8**	**77.9**	**78.6**	**84.8**	**68.8**	**75.9**

- In graph with higher homophily such as Amazon-Clothing and Cora-Full, the performance improvement is greater than those with lower homophily such as DBLP. The performance on BGRL follows similar tendency.This phenomena suggests that, by performing meta-training on the contrastive self-supervised tasks, the representation space of GML model is aligned with GCL used to construct meta-training tasks. The structural knowledge from the pretraining model is leveraged to achieve better prediction performance.
- Comparing with graph learning models without meta-learning, **CSG-Meta** consistently achieves better prediction performance across all problem settings. Above results indicate that **CSG-Meta** can learn effective prior knowledge from the contrastive self-supervised meta-training tasks, and achieves better generalization ability to novel classes with extremely few labels.

3.3 Ablation study

We propose variant models to analyze the effectiveness of different modules in **CSG-Meta**. **CSG-Meta-Random** removes the graph prototype initialization and task-adaptive parameter transformation module in Eq. 7, and perform node classification with a learnable randomly initialized matrix as Eq. 2. **CSG-Meta-Noise** replaces the structural attention denoising module with mean pooling to calculate the prototype and task representations. **CSG-Meta-Clustering** firstly performs clustering on the graph contrastive representation and assigns

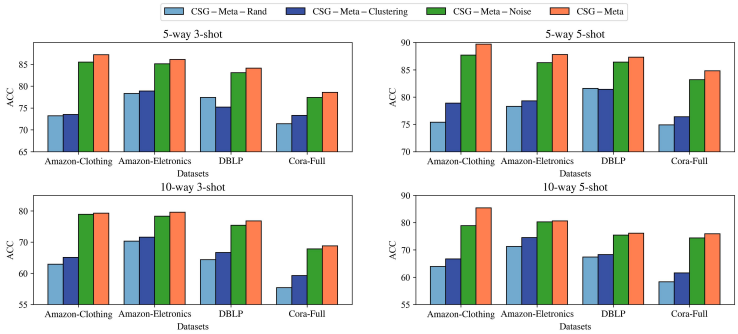

Fig. 2: The results of ablation study on four datasets.

pseudo labels according to the partitioned cluster, based on which the meta-training tasks are constructed. In Fig. 2, the results of ablation studies are shown.

- It can be seen that the prediction performance of **CSG-Meta** is better than that of the three variants across different scenarios, indicating the effectiveness of the proposed three modules. The parameter initialization and transformation modules have the greatest impact on the performance. The classifier of **CSG-Meta-Random** degrades to a linear classifier, which can't capture the task divergence and suffers from overfitting problem.
- **CSG-Meta-Clustering** either does not rely on labeled data to construct meta-training tasks, but its prediction performance is much lower than that of **CSG-Meta**. Because the clustering based tasks can not capture the intra-class similarities learned by GCL, GML model training on clustering based tasks fails to align the representation space with GCL.
- **CSG-Meta-Noise** also hurts the model's performance. This shows that the denoising module can effectively provide more accurate prototype and task representations in task-adaptive graph few-shot classifier, and solve the overfitting problem caused by task divergence and limited labels.

4 Conclusion

In this paper, we focus on the challenges that existing supervised GML algorithms still heavily rely on abundant labeled data from base classes, and suffer from limited structural representation capability. We propose **CSG-Meta**, a structural denoising contrastive self-supervised graph meta-learning model. **CSG-Meta** first leverages the intra-class similarities and inter-class differences as supervisory signals to construct meta-training tasks. **CSG-Meta** can simultaneously learn cross-task meta knowledge and align the representation space of GML and GCL model, generating both distinguishable and transferable node representation for target unseen tasks. We conduct a comprehensive series of experiments on four attributed graph datasets. The results show the effectiveness of **CSG-Meta** and its corresponding modules.

References

1. Ding, K., Wang, J., Li, J., Shu, K., Liu, C., Liu, H.: Graph prototypical networks for few-shot learning on attributed networks. In: CIKM. pp. 295–304 (2020)
2. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: Proceedings of the 34th International Conference on Machine Learning (ICML). pp. 1126–1135 (2017)
3. Hamilton, W.L., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NIPS. vol. 30, pp. 1024–1034 (2017)
4. Huang, K., Zitnik, M.: Graph meta learning via local subgraphs. In: NIPS. pp. 5862–5874 (2020)
5. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2016)
6. Liu, Y., Li, M., Li, X., Giunchiglia, F., Feng, X., Guan, R.: Few-shot node classification on attributed networks with graph meta-learning. In: SIGIR. pp. 471–481 (2022)
7. McAuley, J., Pandey, R., Leskovec, J.: Inferring networks of substitutable and complementary products. In: SIGKDD. pp. 785–794 (2015)
8. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: Online learning of social representations. In: SIGKDD. pp. 701–710 (2014)
9. Snell, J., Swersky, K., Zemel, R.S.: Prototypical networks for few-shot learning. In: NIPS. pp. 4077–4087 (2017)
10. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: LINE: large-scale information network embedding. In: WWW. pp. 1067–1077 (2015)
11. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: ArnetMiner: Extraction and mining of academic social networks. In: SIGKDD. pp. 990–998 (2008)
12. Thakoor, S., Tallec, C., Azar, M.G., Munos, R., Veličković, P., Valko, M.: Bootstrapped representation learning on graphs. In: ICLR Workshop on Geometrical and Topological Representation Learning (2021)
13. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: ICLR (2018)
14. Wang, G., Xie, S., Liu, B., Philip, S.Y.: Review graph based online store review spammer detection. In: ICDM. pp. 1242–1247 (2011)
15. Wang, N., Luo, M., Ding, K., Zhang, L., Li, J., Zheng, Q.: Graph few-shot learning with attribute matching. In: CIKM. pp. 1545–1554 (2020)
16. Wang, S., Chen, C., Li, J.: Graph few-shot learning with task-specific structures. In: NIPS. vol. 35, pp. 38925–38936 (2022)
17. Wang, S., Ding, K., Zhang, C., Chen, C., Li, J.: Task-adaptive few-shot node classification. In: SIGKDD. pp. 1910–1919 (2022)
18. Wang, S., Tan, Z., Liu, H., Li, J.: Contrastive meta-learning for few-shot node classification. In: SIGKDD (2023)
19. Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., Weinberger, K.: Simplifying graph convolutional networks. In: ICML. pp. 6861–6871 (2019)
20. You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., Shen, Y.: Graph contrastive learning with augmentations. In: NIPS. pp. 5812–5823 (2020)
21. Zhou, F., Cao, C., Zhang, K., Trajcevski, G., Zhong, T., Geng, J.: Meta-GNN: On few-shot node classification in graph meta-learning. In: CIKM. pp. 2357–2360 (2019)