Unsupervised Episode Generation for Graph Meta-learning

Jihyeong Jung [1] **Sangwoo Seo** [1] **Sungwon Kim** [2] **Chanyoung Park** [1 2]

Abstract

We propose Unsupervised Episode Generation method called **Neighbors as Queries (NAQ)** to solve the Few-Shot Node-Classification (FSNC) task by *unsupervised Graph Meta-learning*. Doing so enables full utilization of the information of all nodes in a graph, which is not possible in current supervised meta-learning methods for FSNC due to the label-scarcity problem. In addition, unlike unsupervised Graph Contrastive Learning (GCL) methods that overlook the downstream task to be solved at the training phase resulting in vulnerability to class imbalance of a graph, we adopt the episodic learning framework that allows the model to be aware of the downstream task format, i.e., FSNC. The proposed NAQ is a simple but effective *unsupervised* episode generation method that randomly samples nodes from a graph to make a support set, followed by similarity-based sampling of nodes to make the corresponding query set. Since NAQ is *model-agnostic*, any existing supervised graph meta-learning methods can be trained in an unsupervised manner, while not sacrificing much of their performance or sometimes even improving them. Extensive experimental results demonstrate the effectiveness of our proposed unsupervised episode generation method for graph meta-learning towards the FSNC task. Our code is available at: https://github.com/JhngJng/NaQ-PyTorch.

1. Introduction

Graph-structured data are useful and widely applicable in the real-world, thanks to their capability of modeling complex relationships between objects such as user-user relationships in social networks and product networks, etc. To handle tasks such as node classification on graph-structured data, Graph Neural Networks (GNNs) are widely used and have shown remarkable performance (Kipf & Welling, 2017; Veličković et al., 2018). However, it is well known that GNNs suffer from poor generalization when only a small number of labeled samples are provided (Zhou et al., 2019; Ding et al., 2020; Wang et al., 2022b).

To mitigate such issues inherent in the ordinary deep neural networks, few-shot learning methods have emerged, and the dominant paradigm was applying meta-learning algorithms such as MAML (Finn et al., 2017) and ProtoNet (Snell et al., 2017), which are based on the episodic learning framework (Vinyals et al., 2016). Inspired by these methods, recent studies proposed graph meta-learning methods (Zhou et al., 2019; Ding et al., 2020; Huang & Zitnik, 2020; Wang et al., 2022b) to solve the Few-Shot Node Classification (FSNC) task on graphs by also leveraging the episodic learning framework, which is the main focus of this study.

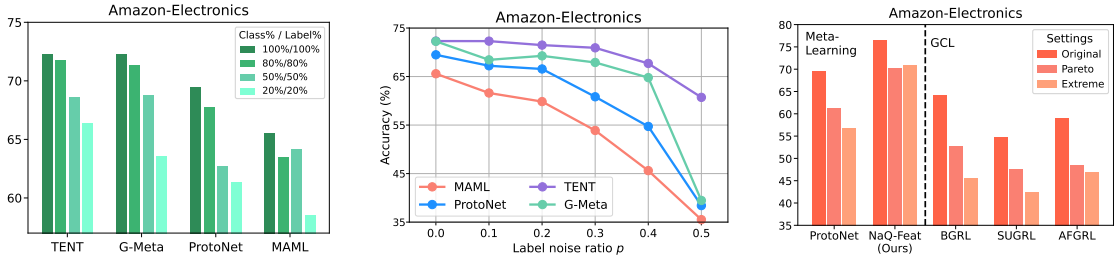
Despite their effectiveness, existing supervised graph meta-learning methods require *abundant* labeled samples from *diverse* base classes for the training. As shown in Figure 1(a), such label-scarcity causes a severe performance drop of representative methods (i.e., TENT (Wang et al., 2022b), G-Meta (Huang & Zitnik, 2020), ProtoNet (Snell et al., 2017), and MAML (Finn et al., 2017)) in FSNC. However, gathering enough labeled data and diverse classes may not be possible, and is costly in reality. More importantly, as these methods depend on a few labeled nodes from base classes, while not fully utilizing all nodes in the graph, they are also vulnerable to noisy labels in base classes (Figure 1(b)). In this respect, *unsupervised methods are indispensable to fundamentally address the label-dependence problem* of existing supervised graph meta-learning methods.

Most recently, TLP (Tan et al., 2022) empirically demonstrated that a simple linear probing with node embeddings pre-trained by Graph Contrastive Learning (GCL) methods outperforms existing supervised graph meta-learning methods in FSNC. This is because GCL methods tend to generate generic node embeddings, since all nodes in a graph are involved in the training.

However, despite the effectiveness of generic node embeddings, we argue that *they are vulnerable to class imbalance in the graph*, which might lead to a significant performance drop due to the lack of model generalizability resulting from

[1]Department of Industrial & Systems Engineering, KAIST [2]Graduate School of Data Science, KAIST. Correspondence to: Chanyoung Park <cy.park@kaist.ac.kr>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.



(a) Impact of the label-scarcity (b) Impact of the label noise (c) Impact of the class imbalance

Figure 1. (a): Impact of the label-scarcity on supervised graph meta-learning methods ('Class%': a rate of available base classes during training, 'Label%': a rate of available labeled samples for each class). (b): Impact of the (randomly injected) label noise p on supervised graph meta-learning methods. (c): Impact of the class imbalance ('Pareto' setting: we kept nodes for top-20% head classes, while keeping only 10 nodes for remaining classes; 'Extreme' setting: the only difference from the 'Pareto' setting is that we kept nodes only for top-5 head classes instead of top-20% classes). (5-way 1-shot)

the discrepancy in the objective between pre-training and fine-tuning (in downstream task) phase (Lu et al., 2021). If the given graph mainly consists of nodes from the majority classes, GCL methods have difficulty in learning embeddings of nodes from the minority classes, which results in poor FSNC performance on such minority classes. On the other hand, as each episode in the episodic learning framework provides the GNN encoder with the information about the downstream task format (i.e., FSNC), meta-learning methods are rather more robust to the class imbalance that may exist in the given graph[1]. To corroborate our argument, we modified the original graph to simulate two class imbalance settings (i.e., 'Pareto' and 'Extreme'), and evaluated GCL methods (i.e., BGRL (Thakoor et al., 2022), SUGRL (Mo et al., 2022), AFGRL (Lee et al., 2022b)) and meta-learning methods (i.e., ProtoNet and NAQ-FEAT (ours)) on the FSNC task (See Figure 1(c)). As expected, the performance deterioration of GCL methods was more severe than meta-learning methods under class imbalance settings.

Therefore, we argue that the FSNC performance can be further enhanced by **unsupervised Graph Meta-learning**, *which can achieve the best of both worlds*: 1) GCL that fully utilizes all nodes in a graph in an unsupervised manner, and 2) Meta-learning whose episodic learning framework is aware of the downstream task format (i.e., *FSNC*).

In this work, we propose a simple yet effective *unsupervised* episode generation method called **Neighbors as Queries (NAQ)**, which enables unsupervised graph meta-learning, to benefit from the generalization ability of meta-learning methods for the FSNC task, while fully utilizing all nodes in a graph. The main idea is to construct a support set by randomly choosing nodes from the entire graph, and generate a corresponding query set via sampling similar nodes based on pre-calculated node-node similarity. It is important to note that our unsupervised episode generation method is *model-agnostic*, i.e., NAQ can be used to train any existing supervised graph meta-learning methods in an unsupervised manner directly or only with minor modifications.

To sum up, our contributions are summarized as follows:

1. We present an *unsupervised episode generation* method, called NAQ, designed to solve the FSNC task via unsupervised graph meta-learning. To our best knowledge, this is the first study that focuses on the unsupervised episode generation of graph meta-learning framework.
2. NAQ is *model-agnostic*; that is, it can be used to train any existing supervised graph meta-learning methods in an unsupervised manner, while not sacrificing much of their performance or sometimes even improving them, without using any labeled nodes.
3. Extensive experimental results demonstrate the effectiveness of NAQ in the FSNC task and highlight the potential of the *unsupervised* graph meta-learning framework.

2. Preliminaries

2.1. Problem Statement

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$ be a graph, where $\mathcal{V}$, $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$, $X \in \mathbb{R}^{|\mathcal{V}| \times d}$ are a set of nodes, a set of edges, and a d-dimensional node feature matrix, respectively. We also use X to denote a set of node features, i.e., $X = \{x_v : v \in \mathcal{V}\}$. Let C be a set of total node classes. Here, we denote the *base classes*, a set of node classes that can be utilized during training, as C_b, and denote the *target classes*, a set of node classes that we aim to predict in downstream tasks given a few labeled samples, as C_t. Note that $C_b \cup C_t = C$ and $C_b \cap C_t = \emptyset$, and the target classes C_t are unknown during training. In common few-shot learning settings, the number of labeled nodes from classes of C_b is sufficient, while we only have a few labeled nodes from classes of C_t in downstream tasks. Now we formulate the ordinary supervised few-shot node classification (FSNC) problem as follows:

[1]Please refer to Section A.2.1 for more detail regarding how the episodic learning allows the model to be robust to class imbalance.

Definition 2.1 (Supervised FSNC)**.** Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$, labeled data (X_{C_b}, Y_{C_b}) and a model f_θ trained on (X_{C_b}, Y_{C_b}), the goal of supervised FSNC is making predictions for $x_q \in X_{C_t}$ (i.e. *query set*) based on a few labeled samples $(x_s, y_s) \in (X_{C_t}, Y_{C_t})$ (i.e., *support set*) during the testing phase.

Based on this problem formulation, we can formulate the unsupervised FSNC problem as below. The only difference is that labeled nodes are not available during training.

Definition 2.2 (Unsupervised FSNC)**.** Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$, **unlabeled data** $X = X_{C_b} \cup X_{C_t}$, and a model f_θ trained on X, the goal of unsupervised FSNC is making predictions for $x_q \in X_{C_t}$ (i.e., *query set*) based on a few labeled samples $(x_s, y_s) \in (X_{C_t}, Y_{C_t})$ (i.e., *support set*) during the testing phase.

Overall, the goal of FSNC is to adapt well to unseen target classes C_t only using a few labeled samples from C_t after training a model f_θ on training data. In this work, we study how to facilitate *unsupervised Graph Meta-learning* to solve the FSNC task. More formally, we consider solving a N-way K-shot FSNC task (Vinyals et al., 2016), where N is the number of distinct target classes and K is the number of labeled samples in a support set. Moreover, there are Q query samples to be classified in each downstream task.

2.2. Episodic Learning Framework

We follow the episodic training framework (Vinyals et al., 2016) that is formally defined as follows:

Definition 2.3 (Episodic Learning)**.** Episodic learning is a learning framework that utilizes a bundle of tasks $\{\mathcal{T}_t\}_{t=1}^{T}$, where $\mathcal{T}_t = (S_{\mathcal{T}_t}, Q_{\mathcal{T}_t})$, $S_{\mathcal{T}_t} = \{(x_{t,i}^{spt}, y_{t,i}^{spt})\}_{i=1}^{N\times K}$ and $Q_{\mathcal{T}_t} = \{(x_{t,i}^{qry}, y_{t,i}^{qry})\}_{i=1}^{N\times Q}$, instead of commonly used mini-batches in the stochastic optimization.

By mimicking the 'format' of the downstream task (i.e., *FSNC*), the episodic learning allows the model to be aware of the task to be solved in the testing phase. Note that existing supervised meta-learning methods require a large number of labeled samples in the training set (X_{C_b}, Y_{C_b}) and a sufficient number of base classes $|C_b|$ (i.e., *diverse base classes*) to generate informative training episodes. However, gathering enough labeled data and diverse classes may not be possible and is usually costly in the real world. As a result, supervised methods fall short of utilizing all nodes in the graph as they rely on a few labeled nodes, and thus lack generalizability.

Therefore, we propose *unsupervised episode generation* methods not only to tackle the label-scarcity problem causing a limited utilization of nodes in the graph, but also to benefit from the episodic learning framework for downstream task-aware learning of node embeddings, thereby being robust to a class imbalance in a graph.

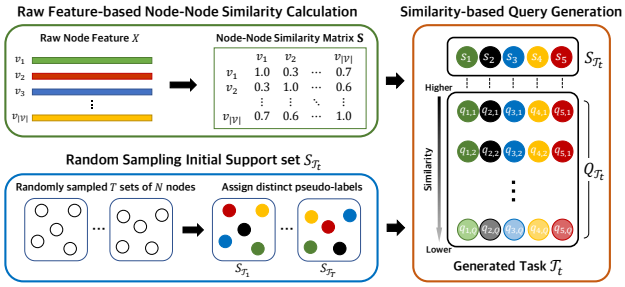


Figure 2. Overview of the NAQ-FEAT.

3. Proposed Method

3.1. Motivation: A Closer Look at Training Episodes

In the episodic learning framework, there are two essential components: 1) support set that provides basic information about the task to be solved, and 2) query set that enables the model to understand about how to solve the given task. *For this reason, the query set should share similar semantics with the support set.* Motivated by this characteristic of episodic learning, we consider the **similarity** condition as the key to our proposed query generation process. Note that in the ordinary supervised setting, the similarity condition is easily achievable, since labels of the support set and query set are known, and thus can be sampled from the same class. However, as our goal is to generate training episodes in an unsupervised manner, how to sample a query set that shares similar semantics with each support set is non-trivial.

3.2. NAQ: Neighbors as Queries

In this work, we propose a simple yet effective query generation method, called **Neighbors as Queries (NAQ)**, which leverages raw feature-level similar nodes as queries. The overview of NAQ can be found in Figure 2.

Support set generation. To generate training episodes $\{\mathcal{T}_t\}_{t=1}^{T}$, we start by randomly sampling T sets of N nodes from the entire graph for the support set generation. Next, we assign pseudo-labels $y_{t,i}$ to each node $x_{t,i} \in \mathcal{T}_t$, i.e., $S_{\mathcal{T}_t} = \{(x_{t,i}, y_{t,i}) \mid x_{t,i} \in \mathcal{V}\}_{i=1}^{N\times K}$. Note that we only generate 1-shot support set (i.e., $K = 1$) regardless of the downstream task setting, to assure that randomly sampled N support set nodes (corresponding to 'N-way') are as much distinguishable from one another as possible.

Query set generation. Then, we generate a corresponding query set $Q_{\mathcal{T}_t}$ with Top-Q similar nodes of each node $x_{t,i}$ in $S_{\mathcal{T}_t}$ based on a pre-calculated node-node similarity matrix $\mathbf{S}$, and give them the same pseudo-label $y_{t,i}$. Formally, we can express this query generation process as follows:

$$Q_{\mathcal{T}_t} = \bigcup\nolimits_{(x_{t,i}, y_{t,i}) \in S_{\mathcal{T}_t}} \mathrm{Top}(\mathbf{S}_{x_{t,i}}, \mathrm{Q}) \qquad (1)$$

where $\mathbf{S}_{x_{t,i}}$ denotes a row of the similarity matrix $\mathbf{S}$ corresponding to the node $x_{t,i}$, and $\text{Top}(\mathbf{S}_{x_{t,i}}, \text{Q})$ indicates a set of Q nodes corresponding to Q largest entries in $\mathbf{S}_{x_{t,i}}$ excluding $x_{t,i}$ itself.

Similarity Metric. For sampling 'similar' nodes to be used as queries, we used cosine similarity for node features such as bag-of-words, and Euclidean distance for the features such as word embeddings. Refer to Section A.3 in the Appendix for further discussions on the similarity metric.

3.2.1. An Extension to NAQ: NAQ-Diff

Since NAQ described above solely relies on the raw node feature X, the structural information that is inherent in graphs is overlooked, which plays an important role depending on the target domain. For example, in citation networks, since the citation relationship between papers implies that these papers usually share similar semantics (i.e., related paper topics), they have similar features even if their class labels are different. Hence, considering structurally similar nodes as queries can be more beneficial than solely relying on the feature-level similar nodes in such cases.

Hence, we present a variant of NAQ, called NAQ-Diff, which utilizes *structurally similar nodes* found by generalized graph diffusion (Gasteiger et al., 2019) as queries. Specifically, NAQ-Diff leverages diffusion matrix $\mathbf{S} = \Sigma_{k=0}^{\infty}\theta_k\mathbf{T}^k$ as node-node similarity matrix, with weighting coefficients θ_k, and the generalized transition matrix $\mathbf{T}$. As edge weights of the diffusion matrix $\mathbf{S}$ can be interpreted as structural closeness, we can sample similar nodes of each support set node from $\mathbf{S}$. It is important to note that computing the diffusion matrix does not require additional computation during training and can be readily calculated before the model training. The overview of NAQ-Diff can be found in Figure 10 in the Appendix, and detailed settings for NAQ-Diff can be found in Section A.4. Hereafter, we call the former version of NAQ that is based on the raw features as NAQ-Feat, and the latter version that is based on the graph structural information as NAQ-Diff.

3.3. Model Training with Episodes from NAQ

In this section, we explain how to train existing meta-learning models with episodes generated by NAQ. Let $\mathcal{T}_t = (S_{\mathcal{T}_t}, Q_{\mathcal{T}_t})$ be a generated episode and $\text{Meta}(\mathcal{T}_t; \theta)$ be any of existing graph meta-learning methods (e.g., MAML, ProtoNet, G-Meta, etc.) with parameter θ. For simplicity of explanation, we used the same notation here even for methods that use meta-batches like MAML. Regardless of whether $\mathcal{T}_t$ is generated from NAQ or an ordinary supervised episode generation, it follows the common format of $\mathcal{T}_t = (S_{\mathcal{T}_t}, Q_{\mathcal{T}_t})$, where $S_{\mathcal{T}_t} = \{(x_{t,i}^{spt}, y_{t,i}^{spt})\}_{i=1}^{N\times K}$ and $Q_{\mathcal{T}_t} = \{(x_{t,i}^{qry}, y_{t,i}^{qry})\}_{i=1}^{N\times Q}$. That is, the only difference is whether $y_{t,i}^{spt}$ and $y_{t,i}^{qry}$ are annotated based on the ground-truth label (supervised) or a psuedo-label (NAQ). Hence, *any of* $\text{Meta}(\mathcal{T}_t; \theta)$ *can be trained in the same way in an 'unsupervised manner with* NAQ*' as ordinary supervised meta-learning methods.* The details are presented in Algorithm 1.

Algorithm 1 Training Meta-learner $\text{Meta}(\,\cdot\,;\theta)$ with NAQ

input Bundle of training episodes $\{\mathcal{T}_t\}_{t=1}^{T}$, Graph Meta-learner $\text{Meta}(\,\cdot\,;\theta)$, learning rate η.
  Randomly initialize the model parameter θ
  for $t = 1, \cdots, T$ **do**
    Step 1: Calculate loss $\mathcal{L}$ by $\text{Meta}(\mathcal{T}_t; \theta)$
    Step 2: Update $\theta \leftarrow \theta - \eta\nabla_\theta\mathcal{L}$
  end for
output $\text{Meta}(\mathcal{T}_t; \theta)$

Remark. Supervised TENT (Wang et al., 2022b) additionally computes cross-entropy loss $\mathcal{L}_{CE}$ over the entire labeled data (X_{C_b}, Y_{C_b}) in Step 1 of Algorithm 1. Therefore, when we train TENT with our NAQ, $\mathcal{L}_{CE}$ is calculated over a single training episode.

It is important to note that since NAQ generates training episodes based on all nodes in a graph, it enables existing graph meta-learning methods to fully utilize all nodes in a graph, while the supervised episode generation fails to do so as it depends on a few labeled nodes from base classes. The detailed model training example in case of ProtoNet (Snell et al., 2017) can be found in Section A.7 in the Appendix.

3.4. Theoretical Insights

In this section, we provide some insights on conditions that enable NAQ to work within the episodic learning framework to justify our motivation of utilizing similar nodes as queries described in Section 3.1. Specifically, we investigate the learning behavior of MAML (Finn et al., 2017), which is one of the most widely adopted meta-learning methods in the perspective of 'generalization error' for a single episode during the training phase. Since each of the existing graph meta-learning methods has its own sophisticated architecture, we only consider MAML here. The formal definition of the expected generalization error is as follows (Gareth et al., 2013; Hastie et al., 2009).

Definition 3.1. Let S, Q, f_S, f be a given training set, test set, an encoder trained on S, and the unknown perfect estimation, respectively. With an error measure $\mathcal{L}$, for a given point $(x', y') \in Q$, an *expected generalization error* is defined as $\mathbb{E}[\mathcal{L}(y', f_S(x'))]$.

By assuming that $y = f(x) + \epsilon$ holds for an arbitrary input-output pair (x, y) ($\mathbb{E}[\epsilon] = 0, \text{Var}(\epsilon) = \sigma^2 < \infty$) and an error measure $\mathcal{L}$ is the mean squared error, we can decompose expected generalization error in Def. 3.1 as follows (Gareth

Table 1. Overall averaged FSNC accuracy (%) with 95% confidence intervals on product networks (Full ver. available at: Table 15)

Dataset	**Amazon-Clothing**					**Amazon- Electronics**						
Setting	5 way		10 way		Avg.	5 way		10 way		20 way		Avg.
Baselines	1 shot	5 shot	1 shot	5 shot	Rank	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot	Rank
MAML (Sup.)	76.13±1.17	84.28±0.87	63.77±0.83	76.95±0.65	10.25	65.58±1.26	78.55±0.96	57.31±0.87	67.56±0.73	46.37±0.61	60.04±0.52	9.33
ProtoNet (Sup.)	75.52±1.12	89.76±0.70	65.50±0.82	82.23±0.62	7.25	69.48±1.22	84.81±0.82	57.67±0.85	75.79±0.67	48.41±0.57	67.31±0.47	5.83
TENT (Sup.)	79.46±1.10	89.61±0.70	69.72±0.80	84.74±0.59	5.25	72.31±1.14	85.25±0.81	62.13±0.83	77.32±0.67	52.45±0.60	69.39±0.50	4.00
G-Meta (Sup.)	78.67±1.05	88.79±0.76	65.30±0.79	80.97±0.59	7.75	72.26±1.16	84.44±0.83	61.32±0.86	74.92±0.71	50.39±0.59	65.73±0.48	5.67
GLITTER (Sup.)	75.73±1.10	89.18±0.74	64.30±0.79	77.73±0.68	9.00	66.91±1.22	82.59±0.83	57.12±0.88	76.26±0.67	49.23±0.57	61.77±0.52	7.00
COSMIC (Sup.)	82.24±0.99	91.22±0.73	74.44±0.75	81.58±0.63	3.75	72.61±1.05	86.92±0.76	65.24±0.82	78.00±0.64	58.71±0.57	70.29±0.44	3.00
TLP-BGRL	81.42±1.05	90.53±0.71	72.05±0.86	83.64±0.63	4.25	64.20±1.10	81.72±0.85	53.16±0.82	73.70±0.66	44.57±0.54	65.13±0.47	8.67
TLP-SUGRL	63.32±1.19	86.35±0.78	54.81±0.77	73.10±0.63	11.50	54.76±1.06	78.12±0.92	46.51±0.80	68.41±0.71	36.08±0.52	57.78±0.49	11.67
TLP-AFGRL	78.12±1.13	89.82±0.73	71.12±0.81	83.88±0.63	5.25	59.07±1.07	81.15±0.85	50.71±0.85	73.87±0.66	43.10±0.56	65.44±0.48	9.00
VNT	65.09±1.23	85.86±0.76	62.43±0.81	80.87±0.63	10.50	56.69±1.22	78.02±0.97	49.98±0.83	70.51±0.73	42.10±0.53	60.99±0.50	10.83
NAQ-FEAT-Best (Ours)	**86.58±0.96**	**92.27±0.67**	**79.55±0.78**	**86.10±0.60**	**1.00**	**76.46±1.11**	**88.72±0.73**	**69.59±0.86**	**81.44±0.61**	**61.05±0.59**	**74.60±0.47**	**1.00**
NAQ-DIFF-Best (Ours)	84.40±1.01	91.72±0.69	73.39±0.79	84.82±0.58	2.25	74.16±1.08	87.09±0.75	65.95±0.81	79.13±0.60	60.40±0.59	73.75±0.42	2.00

et al., 2013; Hastie et al., 2009):

$$\mathbb{E}[\mathcal{L}(y', f_S(x'))] = \big(\mathbb{E}[f_S(x')] - f(x')\big)^2 + \big(\mathbb{E}\big[f_S(x')^2\big] - \mathbb{E}[f_S(x')]^2\big) + \sigma^2. \quad (2)$$

Let us consider the training process of MAML with an encoder f_θ and a training episode $\mathcal{T} = (S_\mathcal{T}, Q_\mathcal{T})$, where $S_\mathcal{T} = \{(x_i^{spt}, y_i^{spt})\}_{i=1}^{N\times K}$ and $Q_\mathcal{T}$ are the N-way K-shot support set and the query set, respectively. During the inner-loop optimization, MAML produces $f_{\theta'}$, where $\theta' = \text{argmin}_\theta \sum_{(x^{spt}, y^{spt})\in S_\mathcal{T}} \mathcal{L}\big(y^{spt}, f_\theta(x^{spt})\big)$.

If we regard the inner-loop optimization of MAML as a training process with training set $S = S_\mathcal{T}$, the outer-loop optimization (i.e., meta-optimization) as a testing process with test set $Q = Q_\mathcal{T}$, and the trained encoder $f_S = f_{S_\mathcal{T}} = f_{\theta'}$, *we can interpret that the meta-optimization actually reduces the generalization error in Eq. 2* over the query set $Q_\mathcal{T}$ with encoder $f_{\theta'}$ (Khodadadeh et al., 2019). With this interpretation, we can re-write Eq. 2 as follows:

$$\mathbb{E}[\mathcal{L}(y^{qry}, f_{\theta'}(x^{qry}))] = \big(\mathbb{E}[f_{\theta'}(x^{qry})] - f_\mathcal{T}(x^{qry})\big)^2 + \big(\mathbb{E}\big[f_{\theta'}(x^{qry})^2\big] - \mathbb{E}[f_{\theta'}(x^{qry})]^2\big) + \sigma^2, \quad (3)$$

where $f_\mathcal{T}$ is the unknown perfect estimation for $\mathcal{T}$. Without loss of generality, we considered a single query $(x^{qry}, y^{qry}) \in Q_\mathcal{T}$ to derive Eq. 3. *As Eq. 3 is used as a loss function, an accurate calculation of Eq. 3 is essential for a better model training* on $\mathcal{T}$ (Khodadadeh et al., 2019).

Remark. Let $s := (x^{spt}, y^{spt}) \in S_\mathcal{T}$ be a specific corresponding support set sample of the query $q := (x^{qry}, y^{qry})$ above. Let $\tilde{y}^{spt}$, $\tilde{y}^{qry}$ be the *true labels* of s, q, respectively. Note that the *same new labels* (i.e., y^{spt}, y^{qry} s.t. y^{spt}=y^{qry}) are assigned to each of x^{spt}, x^{qry} during the training episode generation (regardless of whether it is supervised or not), to perform classification of N classes instead of classifying $|C|$ classes (i.e., total number of classes in the entire dataset) in the training phase of the episodic learning framework. To get an accurate computation of Eq. 3, *it is essential to assure that $\tilde{y}^{spt} = \tilde{y}^{qry}$ holds*. Otherwise, we have $y^{qry} = f_\mathcal{T}(x^{qry}) + \epsilon + \delta$, where δ is an error resulting from $\tilde{y}^{spt} \neq \tilde{y}^{qry}$, which may lead to a suboptimal solution when training with loss defined by Eq. 3.

Unlike the ordinary supervised episode generation in which case $\delta = 0$ holds as condition that $\tilde{y}^{spt} = \tilde{y}^{qry}$ is naturally satisfied, our NAQ cannot guarantee $\delta = 0$ since no label information is given (i.e., $\tilde{y}^{spt}, \tilde{y}^{qry}$ are both unknown) during its episode generation phase. Hence, we argue that it is crucial to discover **class-level similar** query q_{NAQ} for each support set sample $s_{\text{NAQ}} = (x^{spt}_{\text{NAQ}}, y^{spt}_{\text{NAQ}}) \in S_\mathcal{T}$[2] during the query generation process of NAQ. If s and q are class-level similar, i.e., the difference between their corresponding true labels $\tilde{y}^{spt}_{\text{NAQ}}, \tilde{y}^{qry}_{\text{NAQ}}$ are small enough, we would have $|\delta| < \xi$ for some small enough $\xi > 0$ so that we can successfully train encoder f_θ.

In summary, the above analysis explains that discovering a query that is class-level similar enough to a given support set sample is crucial for minimizing the training loss (i.e., the generalization error defined in Eq. 3), which eventually yields a better f_θ. **In this regard, NAQ works well within the episodic learning framework**, since NAQ generates class-level similar query nodes using node-node similarity defined based on the raw node feature (i.e., NAQ-FEAT) and graph structural information (i.e., NAQ-DIFF).

Further discussions on why class-level similarity is sufficient for unsupervised episode generation (Section A.1.1) and an empirical result that our NAQ can find class-level similar queries (Section A.1.2) are provided in the Appendix.

4. Experiments

Evaluation Datasets. We use five benchmark datasets that are widely used in FSNC to comprehensively evaluate the performance of our unsupervised episode genera-

[2]Here, we use $S_\mathcal{T}$ to denote the support set generated by NAQ. For details, see 'Support set generation' process in Section 3.2.

Table 2. Overall averaged FSNC accuracy (%) with 95% confidence intervals on citation networks (Full ver. available at: Table 16, OOT: Out Of Time, which means that the training was not finished in 24 hours, OOM: Out Of Memory on NVIDIA RTX A6000)

Dataset	**Cora-full**							**DBLP**						
Setting	5 way		10 way		20 way		Avg.	5 way		10 way		20 way		Avg.
Baselines	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot	Rank	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot	Rank
MAML (Sup.)	59.28±1.21	70.30±0.99	44.15±0.81	57.59±0.66	30.99±0.43	46.80±0.38	9.67	72.48±1.22	80.30±1.03	60.08±0.90	69.85±0.76	46.12±0.53	57.30±0.48	8.50
ProtoNet (Sup.)	58.61±1.21	73.91±0.93	44.54±0.79	62.15±0.64	32.10±0.42	50.87±0.40	7.67	73.80±1.20	81.33±1.00	61.88±0.86	73.02±0.74	48.70±0.52	62.42±0.45	4.33
TENT (Sup.)	61.30±1.18	77.32±0.81	47.30±0.80	66.40±0.62	36.40±0.45	55.77±0.39	4.50	74.01±1.20	82.54±1.00	62.95±0.85	73.26±0.77	49.67±0.53	61.87±0.47	2.67
G-Meta (Sup.)	59.88±1.26	75.36±0.86	44.34±0.80	59.59±0.66	33.25±0.42	49.00±0.39	7.50	74.64±1.20	79.96±1.08	61.50±0.88	70.33±0.77	46.07±0.52	58.38±0.47	7.00
GLITTER (Sup.)	55.17±1.18	69.33±0.96	42.81±0.81	52.76±0.68	30.70±0.41	40.82±0.41	11.50	73.50±1.25	75.90±1.19	OOT	OOT	OOM	OOM	9.50
COSMIC (Sup.)	62.24±1.15	73.85±0.83	47.85±0.77	59.11±0.60	42.25±0.43	47.28±0.38	6.33	72.34±1.18	80.83±1.03	59.21±0.80	70.67±0.71	49.52±0.51	59.01±0.42	7.50
TLP-BGRL	62.59±1.13	78.80±0.80	49.43±0.79	67.18±0.61	37.63±0.44	56.26±0.39	3.17	73.92±1.19	82.42±0.95	60.16±0.87	72.13±0.74	47.00±0.53	60.57±0.45	4.83
TLP-SUGRL	55.42±1.08	76.01±0.84	44.66±0.74	63.69±0.62	34.23±0.41	52.76±0.40	6.33	71.27±1.15	81.36±1.02	58.85±0.81	71.02±0.78	45.71±0.49	59.77±0.45	8.17
TLP-AFGRL	55.24±1.02	75.92±0.83	44.08±0.70	64.42±0.62	33.88±0.41	53.83±0.39	7.17	71.18±1.17	82.03±0.94	58.70±0.86	71.14±0.75	45.99±0.53	60.31±0.45	7.83
VNT	47.53±1.14	69.94±0.89	37.79±0.69	57.71±0.65	28.78±0.40	46.86±0.40	11.17	58.21±1.16	76.25±1.05	48.75±0.81	66.37±0.77	40.10±0.49	55.15±0.46	11.17
NAQ-FEAT-Best (Ours)	**66.30±1.15**	**80.09±0.79**	**52.23±0.73**	68.87±0.60	**44.13±0.47**	60.94±0.36	**1.33**	73.55±1.16	82.36±0.94	60.70±0.87	72.36±0.73	50.42±0.52	**64.90±0.43**	3.67
NAQ-DIFF-Best (Ours)	66.26±1.15	80.07±0.79	52.17±0.74	**69.34±0.63**	44.12±0.47	**60.97±0.37**	1.67	**76.58±1.18**	**82.86±0.95**	**64.31±0.87**	**74.06±0.75**	**51.62±0.54**	64.78±0.44	**1.17**

tion method: 1) Two product networks (**Amazon-Clothing**, **Amazon-Electronics** (McAuley et al., 2015)), 2) three citation networks (**Cora-Full** (Bojchevski & Günnemann, 2018), **DBLP** (Tang et al., 2008)) in addition to a large-scale dataset **ogbn-arxiv** (Hu et al., 2020). Detailed explanations of the datasets and their statistics are provided in Section A.5 in the Appendix.

Baselines. We use six graph meta-learning models as baselines, i.e., **MAML** (Finn et al., 2017), **ProtoNet** (Snell et al., 2017), **G-Meta** (Huang & Zitnik, 2020), **TENT** (Wang et al., 2022b), **GLITTER** (Wang et al., 2022a), and **COSMIC** (Wang et al., 2023b) to evaluate the performance of our proposed unsupervised episode generation methods, i.e., NAQ-FEAT and NAQ-DIFF. In addition, three recent GCL baselines, i.e., **BGRL** (Thakoor et al., 2022), **SUGRL** (Mo et al., 2022) and **AFGRL** (Lee et al., 2022b), are included as they have shown remarkable performance on the FSNC task without using labels (Tan et al., 2022). Lastly, we compare with **VNT** (Tan et al., 2023) that uses a pretrained graph transformer without labels and fine-tunes injected soft prompts to solve downstream FSNC task. For both NAQ-FEAT and NAQ-DIFF, we sampled $Q = 10$ queries for each support set sample to generate the training episodes. Details on compared baselines and their experimental settings are presented in Section A.6 in the Appendix.

Evaluation. For each dataset except for Amazon-Clothing, we evaluate the performance of the models in 5/10/20-way, 1/5-shot settings, i.e., six settings in total. For Amazon-Clothing, as the validation set contains 17 classes, evaluations on 20-way cannot be conducted. Instead, the evaluation is done in 5/10-way 1/5-shot settings, i.e., four settings in total. In the validation and testing phases, we sampled 50 validation tasks and 500 testing tasks for all settings with 8 queries each. For all the baselines, validation/testing tasks are fixed, and we use linear probing on frozen features to solve each downstream task except for GLITTER and VNT as they use different strategies for solving downstream tasks. We report average accuracy and 95% confidence interval over sampled testing tasks.

4.1. Overall Performance Analysis

The overall results on five datasets are presented in Table 1, 2, and 3. Note that since NAQ is *model-agnostic*, we apply NAQ with all the supervised graph meta-learning models contained in our baselines, and report the best performance among them. We have the following observations.

First, our proposed methods outperform the existing supervised baselines. We attribute this to the episode generation strategy of NAQ that allows the model to extensively utilize all nodes in the graph without reliance on node labels. It is worth noting that for each training episode while other supervised methods use 5-shot support sets, our methods use 1-shot support sets to ensure that the support set nodes are as much distinguishable from one another as possible. Hence, we expect that our methods can be further improved if we develop methods to generate additional support set samples that would make each support set even more distinguishable from one another, which we leave as future work.

Second, our proposed methods outperform methods utilizing the pre-trained encoder in an unsupervised manner (i.e., GCL methods and VNT). Unlike these methods, by applying the episodic learning framework for model training, our methods can capture information about the downstream task 'format' during the model training, leading to generally better performance in the FSNC task.

Third, NAQ-DIFF outperforms NAQ-FEAT in citation networks (Table 2). This result verifies our motivation for presenting NAQ-DIFF in Section 3.2.1, which was to capture the structural information instead of the raw node features in domains where the structural information is more beneficial. On the other hand, NAQ-FEAT outperforms NAQ-DIFF in product networks. This is because products in 'also-viewed' or 'bought-together' relationships are not always similar or related in case of product networks (Zhang et al., 2022), implying that discovering query sets based on 'raw-feature'

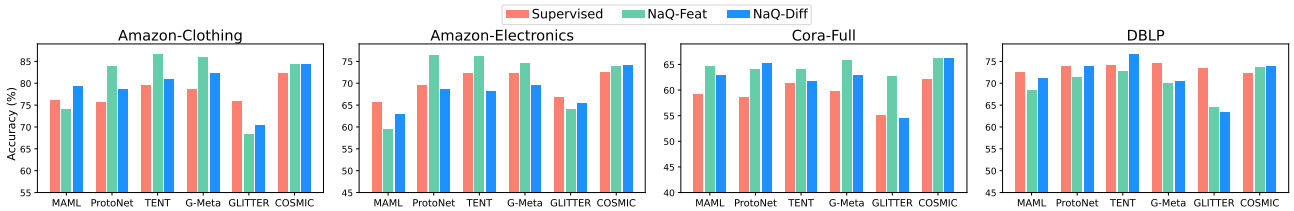


Figure 3. Result of applying NAQ-FEAT and NAQ-DIFF to existing graph meta-learning models (5-way 1-shot).

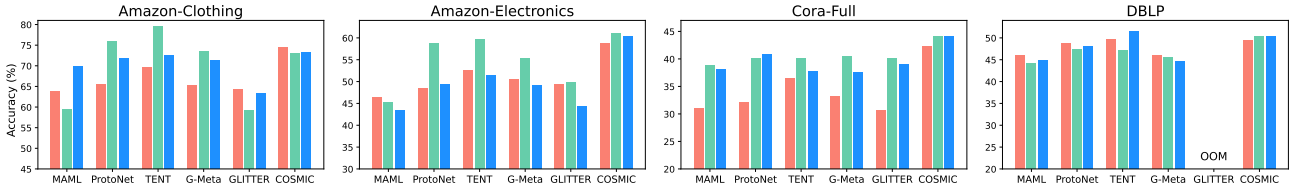


Figure 4. Result of applying NAQ-FEAT and NAQ-DIFF to existing graph meta-learning models in higher way settings (Amazon-Clothing: 10-way 1-shot, Others: 20-way 1-shot).

similarity is more beneficial.

Lastly, NAQ outperforms other baselines on the ogbn-arxiv dataset, which is a large-scale dataset (Table 3). It is worth noting that the performances of two variants of NAQ are at the best and second best in a more challenging one-shot setting. One interesting observation is that NAQ-FEAT outperforms NAQ-DIFF, even though ogbn-arxiv is a citation network. We attribute this to the fact that the raw node features of ogbn-arxiv are 'embeddings' extracted from the skip-gram model. This implies that *high-quality node feature enables* NAQ-FEAT *to find high-quality queries, which leads to a better FSNC performance of* NAQ-FEAT.

In summary, NAQ resolves the label-scarcity problem of supervised graph meta-learning methods and achieve performance enhancement on FSNC tasks by providing training episodes that contain both the information of all nodes in the graph, and the information of the downstream task format to the model.

4.2. Model-agnostic Property of NAQ

In this section, we verify that NAQ can be applied to any existing graph meta-learning models while not sacrificing much of their performance.

In Figure 3 and 4, we observe that our methods retained or even improved the performance of existing graph meta-learning methods across various few-shot settings with only a few exceptions. Particularly, in higher way settings shown in Figure 4, which are more challenging, NAQ generally outperforms supervised methods. Therefore, we argue that our methods allow existing graph meta-learning models to be trained to generate more generalizable embeddings without any use of label information thanks to the full utilization of all nodes in a graph.

Lastly, it is important to note again that the performances

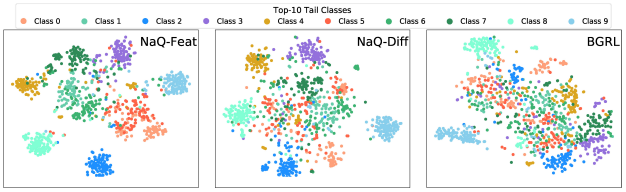
Table 3. Overall averaged FSNC accuracy (%) with 95% confidence intervals on ogbn-arxiv (NAQ base-model: ProtoNet, OOM: Out Of Memory on NVIDIA RTX A6000)

Dataset	**ogbn-arxiv**			
Setting	5 way		10 way	
Baselines	1 shot	5 shot	1 shot	5 shot
MAML (Sup.)	40.61±0.89	58.75±0.89	27.32±0.55	43.87±0.56
ProtoNet (Sup.)	43.34±1.01	58.30±0.95	28.17±0.60	46.11±0.60
TENT (Sup.)	48.06±0.97	63.45±0.88	33.85±0.65	48.14±0.59
G-Meta (Sup.)	41.06±0.87	59.43±0.87	27.20±0.53	45.04±0.53
GLITTER (Sup.)	35.64±0.97	34.51±0.85	20.95±0.50	21.84±0.47
COSMIC (Sup.)	50.32±0.95	63.54±0.80	38.41±0.62	49.31±0.51
TLP-BGRL	49.88±1.01	69.10±0.82	36.40±0.62	56.15±0.54
TLP-SUGRL	49.25±0.97	62.15±0.92	32.87±0.61	45.76±0.60
TLP-AFGRL	OOM	OOM	OOM	OOM
VNT	OOM	OOM	OOM	OOM
NAQ-FEAT (Ours)	**54.09±1.03**	**69.94±0.84**	**41.61±0.68**	**58.18±0.59**
NAQ-DIFF (Ours)	51.45±1.04	66.73±0.89	39.27±0.67	55.93±0.56

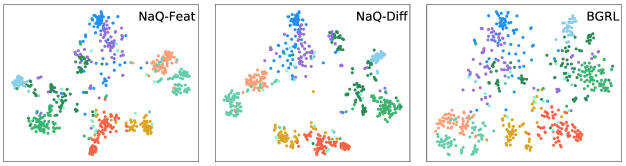
of the supervised models reported in our experiments are only achievable *when they have access to all labeled samples of entire base classes and the given labeled samples in the base classes are clean.* On the other hand, when there is only a limited amount of labeled samples within a limited number of base classes (Figure 1(a)) or there is inherent label noise in the base classes (Figure 1(b)), the performance of supervised models severely drops, while our proposed unsupervised methods would not be affected at all. Furthermore, as will be demonstrated in Section 4.4, the performance of NAQ can be improved by adjusting the number of queries.

4.3. Regarding the Class Imbalance

In this section, we visualize t-SNE (Van der Maaten & Hinton, 2008) embeddings of nodes that belong to the top-10 tail classes. Doing so can further justify our motivation for



(a) Amazon-Electronics



(b) Cora-Full

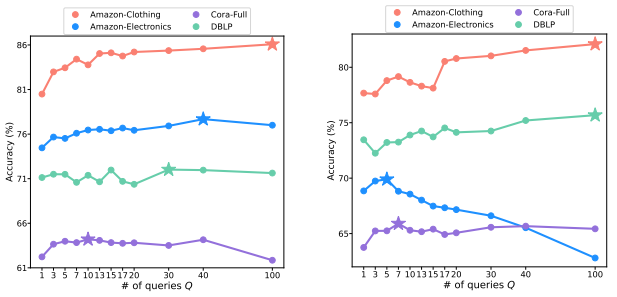
Figure 5. The t-SNE plot of tail-class embeddings (base-model: ProtoNet, NAQ: trained with 5-way 1-shot training episodes)

using unsupervised graph meta-learning on FSNC problems rather than using GCL methods.

As shown in Figure 5, we can observe that our NAQ can learn clearly separable embeddings for tail-class nodes than GCL method BGRL. This result further supports our claim that GCL methods have difficulty in learning embeddings of nodes from minority classes. Therefore, we can verify that additional downstream task 'format' information provided by episodic learning is beneficial for learning tail-class nodes when solving the FSNC problem. Further discussions on why NAQ can attain robustness against the class imbalance (Section A.2.1) and additional results on various dataset biases, such as structure or feature noise (Section A.2.2), are presented in the Appendix.

4.4. Hyperparameter Sensitivity Analysis

So far, the experiments have been conducted with a fixed number of queries, $Q = 10$. In this section, we investigate the effect of the number of queries on the performance of NAQ. To thoroughly explore the effect of the number of queries on NAQ, we check the performance of NAQ with ProtoNet by changing the number of queries $Q \in \{1, 3, 5, 7, 10, 13, 15, 17, 20, 30, 40, 100\}$. We have the following observations from Figure 6: **(1)** In Amazon-Clothing, since both NAQ-FEAT and NAQ-DIFF can discover highly class-level similar queries (Figure 7), they exhibit an increasing tendency in performance as Q increases. **(2)** In the case of Amazon-Electronics, NAQ-FEAT shows a similar tendency as in Amazon-Clothing due to the same reason, while there is a slight performance drop when $Q = 100$. In contrast, NAQ-DIFF shows clearly decreasing performance after $Q = 5$, as its queries have relatively low class-level similarity (Figure 7). From the results above, we can conclude that sampling a proper number of queries Q during the episode generation phase is essential. Otherwise, a significant level of label noise in the generated episode might hinder the model training. **(3)** In the DBLP dataset, NAQ-FEAT shows a nearly consistent performance tendency, while the performance of NAQ-DIFF can be enhanced by increasing the number of queries for training. This is because NAQ-DIFF can sample more class-level similar queries than NAQ-FEAT (Figure 7). From this observation, we again validate the motivation of utilizing structural neighbors as queries in such datasets (Section 3.2.1).



(a) NAQ-FEAT (b) NAQ-DIFF

Figure 6. Effect of the number of queries NAQ (5-way 1-shot, base-model: ProtoNet, star marker: maximal point)

5. Related Work

5.1. Few-Shot Node Classification (FSNC)

Few-shot learning (Vinyals et al., 2016; Finn et al., 2017; Snell et al., 2017) aims to classify unseen target classes with only a few labeled samples based on the meta-knowledge obtained from training on abundant samples from base classes.

Graph Meta-learning. There have been various studies to solve FSNC in graph-structured data. Meta-GNN (Zhou et al., 2019) addresses the problem by directly applying MAML (Finn et al., 2017) on GNN, and GPN (Ding et al., 2020) uses ProtoNet (Snell et al., 2017) architecture with adjusted prototype calculation by considering node importance. G-Meta (Huang & Zitnik, 2020) utilizes subgraph-level embeddings of nodes inside training episodes based on both ProtoNet and MAML frameworks to enable scalable and inductive graph meta-learning. TENT (Wang et al., 2022b) tries to reduce variances within training episodes through node-level, class-level, and task-level adaptations. Meta-GPS (Liu et al., 2022) utilizes various components of network encoder, prototype-based parameter initialization, and S^2 (scaling & shifting) transformation to solve FSNC tasks even on heterophilic graphs. GLITTER (Wang et al., 2022a) claims that the given entire graph structure is redundant for learning node embeddings within the meta-task so that it tries to learn task-specific structure for each meta-task. COSMIC (Wang et al., 2023b) applies a contrastive learning scheme on meta-learning to obtain the intra-class generalizability with hard (unseen) node classes generated by similarity-sensitive mix-up to achieve high inter-class

generalizability.

Graph Meta-learning for Label-scarcity Problem. There were a few studies aiming to alleviate the label-scarcity problem of graph meta-learning methods. TEG (Kim et al., 2023) utilizes equivariant neural networks to capture task-patterns shared among training episodes regardless of node labels, enabling the learning of highly transferable task-adaptation strategies even with a limited number of base classes and labeled nodes. Meanwhile, X-FNC (Wang et al., 2023a) obtains pseudo-labeled nodes via label propagation based on Poisson Learning, and optimizes the model based on information bottleneck to discard irrelevant information within the augmented support set. Although these methods extract useful meta-knowledge based on training episodes (i.e., TEG) or from pseudo-labeled nodes (i.e., X-FNC), they still highly depend on a few labeled nodes during the model training, and thus still fall short of utilizing the information of all nodes in the graph. As a result, their FSNC performance degrades as the number of labeled nodes and base classes decreases (Wang et al., 2023a; Kim et al., 2023).

Unsupervised FSNC. As existing graph meta-learning methods suffer from the label-scarcity problem, there were several studies to handle the FSNC problem in an unsupervised manner. TLP (Tan et al., 2022) utilizes GCL methods to solve FSNC, and it has shown superior FSNC performance than graph meta-learning methods without labels. VNT (Tan et al., 2023) applies graph transformer on FSNC and solves downstream FSNC task by only fine-tuning 'virtual' nodes injected as soft prompts and the classifier with given a few-labeled samples in the downstream task. Most recently, (Liu et al., 2024) analyse advantages of applying GCL on FSNC over graph meta-learning in two aspects: 1) utilization of graph augmentation, and 2) explicit usage of all nodes in a graph. Base on this analysis, they present a GCL-based method named COLA that aims to combine GCL and meta-learning by constructing meta-tasks without labels *during* the training phase, which is computationally costly. Although it shares some similarities with our method NAQ, COLA focuses on GCL-based model while our NAQ focuses on enabling unsupervised graph meta-learning.

5.2. Unsupervised Meta-learning

In computer vision, several unsupervised meta-learning methods exist that attempt to address the limitations of requiring abundant labels for constructing training episodes. More precisely, UMTRA (Khodadadeh et al., 2019) and AAL (Antoniou & Storkey, 2019) are similar methods, making queries via image augmentation on randomly sampled support set samples. In addition, AAL focuses on task generation, while UMTRA is mainly applied to MAML. On the other hand, CACTUs (Hsu et al., 2018) aims to make episodes based on pseudo-labels obtained from cluster assignments, which come from features pre-trained in an unsupervised fashion. LASIUM (Khodadadeh et al., 2020) generates synthetic training episodes that can be combined with existing models, such as MAML and ProtoNet, with generative models. Moreover, Meta-GMVAE (Lee et al., 2021) uses VAE (Kingma & Welling, 2013) with Gaussian mixture priors to solve the few-shot learning problem.

6. Limitations & Future Work

Although NAQ has proven its effectiveness for Few-Shot Node Classification (FSNC), it is crucial to acknowledge its limitations, presented below, to stimulate future work.

6.1. Computational Issue of NAQ-DIFF

Due to some technical issues regarding sparse matrix multiplication, we cannot even calculate the truncated approximation of the graph Diffusion for the dataset, which has many edges (e.g., ogbn-products). This problem hinders the applicability of NAQ-DIFF to large real-world datasets. Hence, it will be promising to devise unsupervised episode generation methods that can fully leverage the structural information of graphs while reducing computational costs.

6.2. Problem of Naïve Support Set Generation.

Since NAQ depends on naïve random sampling for support set generation, there is a possibility that nodes having the same label can be assigned to a distinct support set, which is an undesirable case. Although we sample 1-shot support sets to avoid the above problem, developing a more sophisticated support set generation method that mitigates the problem mentioned above and generates a K-shot ($K > 1$) support set will be valuable future work.

7. Conclusion

In this study, we proposed NAQ, a novel unsupervised episode generation algorithm that enables unsupervised graph meta-learning. NAQ generates 1) support sets by random sampling from the entire graph, and 2) query sets by utilizing feature-level similar nodes (i.e., NAQ-FEAT) or structurally similar neighbors from graph diffusion (i.e., NAQ-DIFF). As NAQ generates training episodes out of all nodes in the graph without any label information, it can address the label-scarcity problem of supervised graph meta-learning models. Moreover, generated episodes from NAQ can be used for training any existing graph meta-learning models almost without modifications and even boost their performance on the FSNC task. Extensive experimental studies on various downstream task settings demonstrate the superiority and potential of NAQ.

Acknowledgment

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (RS-2024-00335098), and supported by National Research Foundation of Korea(NRF) funded by Ministry of Science and ICT (NRF-2022M3J6A1063021).

Impact Statement

This paper proposes to advance the field of unsupervised Machine Learning on graph-structured data like social networks. Although our research might have many potential ethical/societal impacts during its application, we believe no specific points should be emphasized here.

References

Antoniou, A. and Storkey, A. Assume, augment and learn: Unsupervised few-shot meta-learning via random labels and data augmentation. *arXiv preprint arXiv:1902.09884*, 2019.

Bojchevski, A. and Günnemann, S. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018.

Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C. F., and Huang, J.-B. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

Ding, K., Wang, J., Li, J., Shu, K., Liu, C., and Liu, H. Graph prototypical networks for few-shot learning on attributed networks. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 295–304, 2020.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 1126–1135. PMLR, 2017.

Gareth, J., Daniela, W., Trevor, H., and Robert, T. *An introduction to statistical learning: with applications in R*. Springer, 2013.

Gasteiger, J., Weißenberger, S., and Günnemann, S. Diffusion improves graph learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. Bootstrap your own latent: A new approach to self-supervised learning. *NeurIPS*, 2020.

Hastie, T., Tibshirani, R., and Friedman, J. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

Hsu, K., Levine, S., and Finn, C. Unsupervised learning via meta-learning. *arXiv preprint arXiv:1810.02334*, 2018.

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*, 2020.

Huang, J., Du, L., Chen, X., Fu, Q., Han, S., and Zhang, D. Robust mid-pass filtering graph convolutional networks. In *Proceedings of the ACM Web Conference 2023*, 2023.

Huang, K. and Zitnik, M. Graph meta learning via local subgraphs. *Advances in Neural Information Processing Systems*, 33, 2020.

Khodadadeh, S., Boloni, L., and Shah, M. Unsupervised meta-learning for few-shot image classification. *Advances in neural information processing systems*, 32, 2019.

Khodadadeh, S., Zehtabian, S., Vahidian, S., Wang, W., Lin, B., and Bölöni, L. Unsupervised meta-learning through latent-space interpolation in generative models. *arXiv preprint arXiv:2006.10236*, 2020.

Kim, S., Lee, J., Lee, N., Kim, W., Choi, S., and Park, C. Task-equivariant graph few-shot learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1120—-1131, 2023.

Kingma, D. P. and Ba, J. L. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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

Lee, D. B., Min, D., Lee, S., and Hwang, S. J. Meta-gmvae: Mixture of gaussian vae for unsupervised meta-learning. In *International Conference on Learning Representations*, 2021.

Lee, N., Hyun, D., Lee, J., and Park, C. Relational self-supervised learning on graphs. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 1054–1063, 2022a.

Lee, N., Lee, J., and Park, C. Augmentation-free self-supervised learning on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7372–7380, 2022b.

Liu, H., Feng, J., Kong, L., Tao, D., Chen, Y., and Zhang, M. Graph contrastive learning meets graph meta learning: A unified method for few-shot node tasks. In *Proceedings of the ACM Web Conference 2024*, 2024.

Liu, X., Ding, J., Jin, W., Xu, H., Ma, Y., Liu, Z., and Tang, J. Graph neural networks with adaptive residual. *Advances in Neural Information Processing Systems*, 2021.

Liu, Y., Li, M., Li, X., Giunchiglia, F., Feng, X., and Guan, R. Few-shot node classification on attributed networks with graph meta-learning. In *Proceedings of the 45th international ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 471–481, 2022.

Lu, Y., Jiang, X., Fang, Y., and Shi, C. Learning to pre-train graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 4276–4284, 2021.

McAuley, J., Pandey, R., and Leskovec, J. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 785–794, 2015.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

Mo, Y., Peng, L., Xu, J., Shi, X., and Zhu, X. Simple unsupervised graph representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 7797–7805, 2022.

Page, L., Brin, S., Motwani, R., and Winograd, T. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

Rong, Y., Huang, W., Xu, T., and Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. *Advances in Neural Information Processing Systems*, 30, 2017.

Tan, Z., Wang, S., Ding, K., Li, J., and Liu, H. Transductive linear probing: A novel framework for few-shot node classification. In *Learning on Graphs Conference*, 2022.

Tan, Z., Guo, R., Ding, K., and Liu, H. Virtual node tuning for few-shot node classification. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2177—2188, 2023.

Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., and Su, Z. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 990–998, 2008.

Thakoor, S., Tallec, C., Azar, M. G., Azabou, M., Dyer, E. L., Munos, R., Veličković, P., and Valko, M. Large-scale representation learning on graphs via bootstrapping. In *International Conference on Learning Representations*, 2022.

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

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

Wang, K., Shen, Z., Huang, C., Wu, C.-H., Dong, Y., and Kanakia, A. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 2020.

Wang, S., Chen, C., and Li, J. Graph few-shot learning with task-specific structures. In *NeurIPS*, 2022a.

Wang, S., Ding, K., Zhang, C., Chen, C., and Li, J. Task-adaptive few-shot node classification. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1910–1919, 2022b.

Wang, S., Dong, Y., Ding, K., Chen, C., and Li, J. Few-shot node classification with extremely weak supervision. In *Proceedings of the 16th International Conference on Web Search and Data Mining*, 2023a.

Wang, S., Tan, Z., Liu, H., and Li, J. Contrastive meta-learning for few-shot node classification. In *SIGKDD*, 2023b.

Wu, H., Wang, C., Tyshetskiy, Y., Docherty, A., Lu, K., and Zhu, L. Adversarial examples on graph data: Deep insights into attack and defense. *IJCAI*, 2019.

Zhang, C., Du, Y., Zhao, X., Han, Q., Chen, R., and Li, L. Hierarchical item inconsistency signal learning for sequence denoising in sequential recommendation. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, 2022.

Zhang, J., Zhang, H., Xia, C., and Sun, L. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*, 2020.

Zhang, L. and Lu, H. A feature-importance-aware and robust aggregator for gcn. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 1813–1822, 2020.

Zhou, F., Cao, C., Zhang, K., Trajcevski, G., Zhong, T., and Geng, J. Meta-gnn: On few-shot node classification in graph meta-learning. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019.

A. Appendix

A.1. Regarding 'Class-level Similarity'

A.1.1. Why is 'class-level similarity' sufficient?

In Section 3.4, we justified our 'similarity' condition presented in Section 3.1 in terms of *'class-level similarity'*. In this section, we provide an explanation on why considering the class-level similarity instead of the exact same class condition, which is in fact impossible because the class information is not given, is sufficient for the query generation process in NAQ, and further justify why our method outperforms supervised meta-learning methods.

Overall, we conjecture that training a model via episodic learning with episodes generated from NAQ can be done successfully not only because our methods enable the utilization of all nodes in a graph, but also because our methods generate sufficiently informative episodes that enable the model to learn the downstream task format. When we take a closer look at the training process of an episodic learning framework, the model only needs to classify a small number (N-way) of classes in a single episode unlike the conventional training scheme requiring the model to classify total $|C|$ classes in a graph. For this reason, we do not have to strive for finding queries whose labels are exactly the same as their corresponding support set sample as in ordinary supervised episode generation. Therefore, finding class-level similar queries is sufficient for generating informative training episodes.

Moreover, if we can generate training episodes that have queries similar enough to the corresponding support set sample while being dissimilar to the remaining $N-1$ support set samples, we further conjecture that the episodes utilizing class-level similar queries from NAQ is even more beneficial than episodes generated in the ordinary supervised manner. This is because the episodes generated by NAQ provide helpful information from different but similar classes while episodes generated in the supervised manner merely provide the information within the same classes as support set sample. To further demonstrate that NAQ has the ability to discover such class-level similar queries, empirical analysis is provided in Section A.1.2.

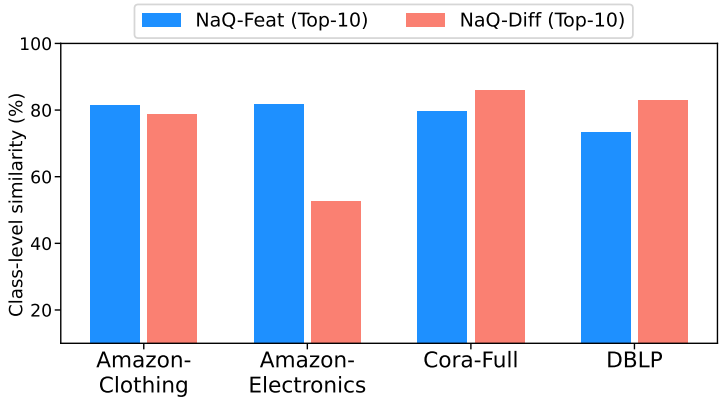


Figure 7. Averaged class-level similarity between each node and top-10 similar nodes found via NAQ-FEAT and NAQ-DIFF

A.1.2. NAQ discovers 'class-level similar' queries

In this section, we provide an empirical evidence that NAQ can find class-level similar neighbors as queries for each support set sample (Figure 7), and we further analyze the experimental results of our methods based on that evidence.

To verify that queries found by NAQ are class-level similar, we measure the averaged class-level similarity between a node and its top-10 similar nodes found by NAQ-FEAT (raw feature similarity) and NAQ-DIFF (graph diffusion) in all four datasets. The class-level similarity between two nodes is computed based on the similarity between their class centroids, where the centroid of class c is computed by $\mathbf{a}_c = \text{MEAN}(\sum x_i \cdot \mathbb{I}\{y_i = c\})$ with x_i denoting the raw feature of node i and y_i denoting the label of node i. The results are presented in Figure 7. In most cases, similar nodes found by NAQ-FEAT and NAQ-DIFF exhibit a high-level ($\sim$80%) average class-level similarity. This result shows that NAQ-FEAT and NAQ-DIFF can discover enough class-level similar nodes as queries for each support set sample.

In addition, we can further justify our arguments in Section 3.2.1 and the experimental results in Section 4.1 based on these results. First, in Figure 7, we observe that we can sample more class-level similar queries by NAQ-DIFF than NAQ-FEAT in citation networks (i.e., Cora-Full and DBLP), implying that considering graph structural information can be more beneficial

in citation networks for the reason described in Section 3.2.1. Second, since NAQ-DIFF can discover class-level similar queries in the DBLP dataset, it shows superior performance in the DBLP dataset even though DBLP has a low homophily ratio. Therefore, we emphasize again that discovering class-level similar queries is essential in generating informative episodes. Third, we observe that the variant of NAQ with higher class-level similarity always performs better in the downstream FSNC task, implying that making queries class-level similar to corresponding support set samples is directly related to the performance of NAQ.

In summary, we quantitatively demonstrated that NAQ indeed discovers class-level similar nodes without using label information, and showed that the experimental results for our methods align well with our motivation regarding the support-query similarity, presented in Section 3.1 and justified in Section 3.4.

A.2. Regarding the Inherent Bias in Graphs

Although in the main paper, we mentioned about the vulnerability of existing GCL methods to class imbalance, there exist other inherent bias that may exist in graphs, i.e., structure noise and feature noise. In this section, we provide further discussions on class imbalance (Section A.2.1) followed by additional results under structure and feature noise (Section A.2.2).

A.2.1. FURTHER DISCUSSION ON CLASS IMBALANCE

In this section, we discuss in detail why existing graph meta-learning methods and our NAQ retains robustness against class imbalance in a graph. Even though we can conclude that task format information learned by episodic learning framework makes the model to be robust against the class imbalance from various experimental results (See Figure 1(c) and Section 4.3), here we delve deeper into which elements of the (ordinary) supervised or unsupervised episode generation (NAQ) contribute to the robustness against the class imbalance. In addition, we present empirical analysis to further support our claim that episodic learning is beneficial to attain robustness against the class imbalance.

Supervised Graph Meta-learning. In the training episode generation step of the ordinary supervised meta-learning methods, they first sample N-way classes in base classes C_b, then sample K-shot support set samples and Q queries within each of sampled classes. As a result, all classes *in base classes* are treated equally regardless of the number of samples they contain. Therefore, with an aid of the task format information obtained via episodic learning, supervised graph meta-learning can be robust to class imbalance in a graph.

Table 4. Class-level similarity between each node from Top-p% tail classes in the graph and top-10 similar nodes found via NAQ-FEAT and NAQ-DIFF (Results of 100%: reported in Figure 7)

Datasets	**Amazon-Clothing**		**Amazon-Electronics**		**Cora-Full**		**DBLP**	
top-p% tail classes	NAQ-FEAT	NAQ-DIFF	NAQ-FEAT	NAQ-DIFF	NAQ-FEAT	NAQ-DIFF	NAQ-FEAT	NAQ-DIFF
10%	∼78.7%	∼75.2%	∼72.3%	∼48.2%	∼69.7%	∼77.9%	∼66.6%	∼75.1%
20%	∼81.3%	∼78.2%	∼74.1%	∼51.6%	∼70.7%	∼77.6%	∼68.3%	∼78.0%
50%	∼81.7%	∼80.7%	∼77.8%	∼53.0%	∼74.6%	∼81.8%	∼70.4%	∼80.9%
80%	∼80.8%	∼79.0%	∼78.9%	∼52.5%	∼77.8%	∼84.6%	∼71.9%	∼82.1%
100%	∼81.6%	∼78.8%	∼81.9%	∼52.7%	∼79.8%	∼86.0%	∼73.5%	∼83.0%

Unsupervised Graph Meta-learning with NAQ. Since the class label information is not given to NAQ, addressing class imbalance is not trivial as in the supervised case described above. Instead, NAQ samples 'class-level similar' queries to the support set nodes from tail classes, which can help learning tail-class embeddings. To demonstrate that NAQ still finds 'class-level similar' queries to the tail-class nodes, we measured the averaged class-level similarity between node of the top-p% tail classes and top-10 similar nodes found by NAQ. Results can be found in Table 4. We observe that NAQ still finds class-level similar enough queries even for the nodes from tail classes, especially in the dataset in which each variant of NAQ outperforms (i.e., NAQ-FEAT for product networks (Amazon-Clothing/Electronics), and NAQ-DIFF for citation networks (Cora-Full, DBLP)). For top-10% tail classes, queries found by NAQ-FEAT exhibit 78.67% / 72.29% class-level similarity in Amazon-Clothing / Amazon-Electronics, and queries found by NAQ-DIFF exhibit 77.89% / 75.05% class-level similarity in Cora-Full / DBLP. Therefore, we can conclude that 'class-level similar' queries found by NAQ are beneficial for learning tail-class embeddings from the results of Table 4 and Section 4.3.

Role of the Episodic Learning Framework. To empirically examine whether downstream task format information provided by episodic learning helps attain robustness against the class imbalance in the graph or not, we observed the change in the quality of t-SNE embeddings of the top-10 tail-class nodes produced by NAQ-DIFF when N-way becomes larger (i.e., $N = 5 \rightarrow 20$, more challenging training setting) in the Amazon-Electronics dataset.

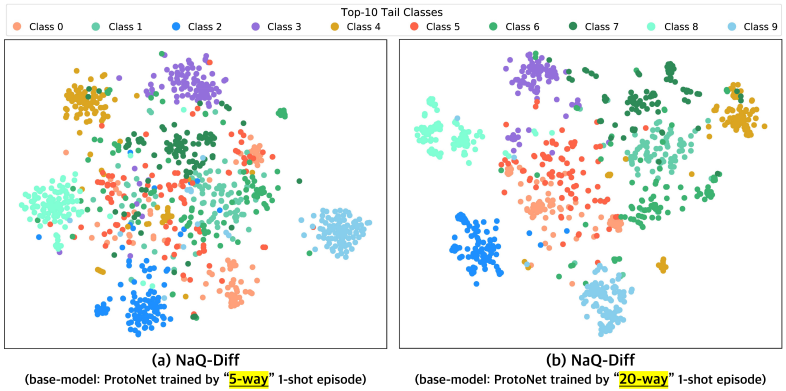
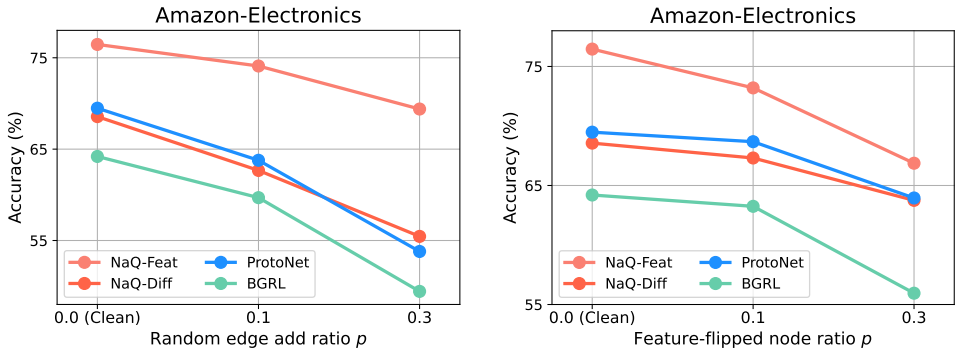


Figure 8. (Left): The t-SNE plot of tail-class embeddings produced by NAQ-DIFF trained with *5-way* training episodes. (Right): The t-SNE plot of tail-class embeddings produced by NAQ-DIFF trained with *20-way* training episodes (base-model: ProtoNet)

As we observed in Figure 5(a), NAQ-DIFF has difficulty in finding class-level similar queries (See Figure 7 and Table 4) due to the low average degree (∼2.06) of the Amazon-Electronics dataset, so that produces inferior tail-class embedding quality compared to NAQ-FEAT in case of Amazon-Electronics. However, by training with more challenging episodes (i.e., 20-way training episodes), NAQ-DIFF *can produce clearly separable tail-class node embeddings even in the Amazon-Electronics dataset.* Therefore, we can conclude that downstream task 'format' information provided by episodic learning benefits learning about minority classes.



(a) Structure noise (b) Feature noise

Figure 9. (a): Impact of the structure noise, (b): Impact of the feature noise (5-way 1-shot, NAQ base-model: ProtoNet)

A.2.2. ADDITIONAL RESULTS ON THE INHERENT BIAS

Structure Noise. Since structure noise in a graph is also a crucial inherent bias that is known to deteriorate the performance of GNNs, we also evaluated the FSNC performance when there are noisy edges in the given graph structure. To perturb graph structure, we considered random edge addition, because adding edges are known to be a more effective attack (Wu et al., 2019). We add random edges as much as $p \in \{0.1, 0.3\}$ of the number of edges in the original graph. By adjusting the random edge adding ratio p, we examined the impact of the structure noise on 5-way 1-shot FSNC performance. Results are presented in Figure 9(a). We observe that meta-learning methods are more robust than a GCL method, BGRL, which we attribute to the task format information learned by episodic learning framework. Moreover, NAQ-FEAT shows significantly better robustness compared with other baselines, as it only utilizes clean raw node feature instead of noisy structure for training episode generation.

Feature Noise. We also examined the impact of the feature noise on the FSNC performance. After random sampling $p \in \{0.1, 0.3\}$ nodes to be corrupted (Liu et al., 2021), we injected feature noise into sampled nodes by randomly flipping 0/1 value on each dimension of the node feature $X_{:,i}$ from Bernoulli distribution with probability $\frac{1}{d}\sum_{i=1}^{d} X_{:,i}$ (Zhang & Lu, 2020; Huang et al., 2023). By adjusting the ratio of noisy nodes, we examined the impact of noisy features on 5-way 1-shot FSNC performance. Results are presented in Figure 9(b). We observe that as more noise is added, BGRL shows a significant performance drop compared to meta-learning methods except for NAQ-FEAT, which we attribute again to the task format information learned by episodic learning framework. As expected, as NAQ-FEAT relies on the node features for the similarity computation, its performance drops as more feature noise is added. Thus, developing a more robust algorithm under feature noise will be a promising direction for future work.

A.3. Ablation Study: Similarity Metric in NAQ-FEAT

As discussed in the Section 3.2, the choice of the similarity metric is an important factor for NAQ-FEAT, since inappropriate choice of the similarity metric can lead to the wrong selection of queries. To examine the impact of the similarity metric, we use the cosine similarity and the negative Euclidean distance to measure the class-level similarity between each node and top-10 similar nodes found by NAQ-FEAT (Table 5), as done in Section A.1.2. Note that Jaccard similarity is excluded when measuring class-level similarity since it cannot be applied to the continuous features. In addition, we evaluated the 5-way 1-shot FSNC performance on each dataset when using the cosine similarity, Jaccard similarity, and the negative Euclidean distance as the similarity metric (Table 6). Note that all hyperparameter settings of NAQ-FEAT other than the similarity metric are identical.

In Table 5, we observe that using the cosine similarity as the similarity metric discovers more class-level similar nodes than using the negative Euclidean distance. As a result, in Table 6, the FSNC accuracy when using the cosine similarity is superior to when using the negative Euclidean distance. Note that this is mainly due to the fact that the datasets used in this experiment have bag-of-words node features, and thus the cosine similarity serves as a better metric. Therefore, we can confirm that choosing an appropriate similarity metric is important.

Table 5. Impact of the similarity metric on class-level similarity between each node and top-10 similar nodes found via NAQ-FEAT.

Datasets (Feature type: bag-of-words)	Avg. Class-level sim. (Cosine sim.)	Avg. Class-level sim. (Neg. Euclidean dist.)
Amazon-Clothing	$\sim 81.6\%$	$\sim 61.0\%$
Amazon-Electronics	$\sim 81.9\%$	$\sim 64.6\%$
Cora-Full	$\sim 79.8\%$	$\sim 40.4\%$
DBLP	$\sim 73.5\%$	$\sim 19.1\%$

Table 6. Impact of the similarity metric on NAQ-FEAT (5-way 1-shot, base-model: ProtoNet)

Datasets (Feature type: bag-of-words)	FSNC Accuracy (Cosine sim.)	FSNC Accuracy (Jaccard sim.)	FSNC Accuracy (Neg. Euclidean dist.)
Amazon-Clothing	83.77%	83.35%	80.83%
Amazon-Electronics	76.46%	76.63%	70.68%
Cora-Full	64.20%	63.53%	45.60%
DBLP	71.38%	72.68%	67.53%

When comparing cosine similarity and Jaccard similarity, since they are similar metrics when measuring similarities in bag-of-words data, NAQ-FEAT with both similarity metrics shows similar FSNC performance over four datasets as shown in Table 6. Thus, we have the freedom to choose one of those two metrics when using NAQ-FEAT on data with bag-of-words features. However, Jaccard similarity cannot be computed with continuous features, as we mentioned above. Hence, it will be more beneficial to consider cosine similarity as a similarity metric due to its generality.

Lastly, note that we did not consider the learnable similarity metric since it requires node-node similarity calculation process per model update for episode generation, which is computationally burdensome. For this reason, we have not considered the learnable metric since we pursued an episode generation method that can be performed *before the training phase.*

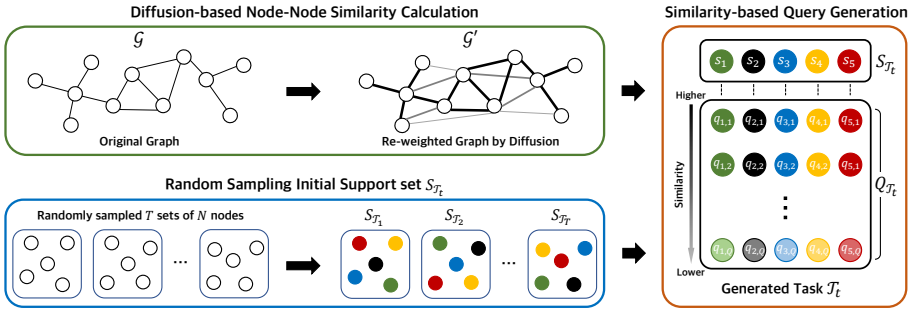


Figure 10. Overview of the NAQ-DIFF. The only difference from NAQ-FEAT is that NAQ-DIFF utilizes graph diffusion instead of raw-feature-based similarity to get node-node similarity.

A.4. Details on NAQ-DIFF

For NAQ-DIFF, we used Personalized PageRank (PPR) (Page et al., 1999)-based diffusion to obtain diffusion matrix $\mathbf{S}$, where $\theta_k^{PPR} = \alpha(1-\alpha)^k$, with teleport probability $\alpha \in (0, 1)$, as the weighting coefficient θ_k. In our experiments, $\alpha = 0.1$ is used to calculate PPR-based diffusion. Also, we used $\tilde{T}_{sym} = (w_{loop} \cdot \mathbf{I}_N + D)^{-1/2}(w_{loop} \cdot \mathbf{I}_N + A)(w_{loop} \cdot \mathbf{I}_N + D)^{-1/2}$, with the self-loop weight $w_{loop} = 1$, as transition matrix, where $A \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is an adjacency matrix of the graph $\mathcal{G}$ and D is a diagonal matrix whose entries $D_{ii} = \sum_j A_{ij}$ are each node's degree.

Last but not least, although there can be other approaches for capturing the graph structural information (e.g., using the adjacency matrix, or using a k-NN graph computed based on node embeddings learned by a GNN encoder *during the training phase*) (Lee et al., 2022a), we choose the graph diffusion as it captures more global information than the adjacency matrix, and computationally more efficient than the k-NN approach.

A.5. Details on Evaluation Datasets

The following is the details on evaluation datasets used in this work.

- **Amazon-Clothing** (McAuley et al., 2015) is a product-product network, whose nodes are products from the category "Clothing, Shoes and Jewelry" in Amazon. Node features are constructed from the product descriptions, and edges were created based on "also-viewed" relationships between products. The node class is a low-level product category.

- **Amazon-Electronics** (McAuley et al., 2015) is a network of products, whose nodes are products from the category "Electronics" in Amazon. Node features are constructed from the product descriptions, and edges represent the "bought-together" relationship between products. The node class is a low-level product category.

- **Cora-Full** (Bojchevski & Günnemann, 2018) is a citation network, whose nodes are papers. The node features are constructed from a bag-of-words representation of each node's title and abstract, and edges represent the citation relationship between papers. The node class is the paper topic.

- **DBLP** (Tang et al., 2008) is a citation network, whose nodes are papers. Node features are constructed from their abstracts, and edges represent the citation relationship between papers. The node class is the venue where the paper is published.

- **ogbn-arxiv** (Hu et al., 2020) is a citation network, whose nodes are CS arXiv papers. Node features are constructed by averaging the embeddings of words in the title and abstract, where the word embeddings are obtained from the skip-gram model (Mikolov et al., 2013) over the MAG (Wang et al., 2020) corpus. Edges are citation relationships between papers, and the node class is 40 subject areas of arXiv CS papers.

The detailed statistics of the datasets can be found in Table 7. "Hom. ratio" denotes the homophily ratio of each dataset, and "Class split" denotes the number of distinct classes used to generate episodes in training (*only for supervised settings*), validation, and testing phase, respectively. For ogbn-arxiv, due to the GPU memory issue, graph diffusion calculation is done as a truncated sum. Moreover, as node features in ogbn-arxiv are word embeddings, we used the negative Euclidean distance as the similarity metric used for sampling query nodes in NAQ.

Table 7. Dataset statistics.

Dataset	# Nodes	# Edges	# Features	# Labels	Class split	Hom. ratio
Amazon-Clothing	24,919	91,680	9,034	77	40/17/20	0.62
Amazon-Electronics	42,318	43,556	8,669	167	90/37/40	0.38
Cora-Full	19,793	65,311	8,710	70	25/20/25	0.59
DBLP	40,672	288,270	7,202	137	80/27/30	0.29
ogbn-arxiv	169,343	1,166,243	128	40	15/10/15	0.43

A.6. Details on Compared Baselines & Experimental Settings

Details on compared baselines are presented as follows.

- **MAML** (Finn et al., 2017) aims to find good initialization for downstream tasks. It optimizes parameters via two-phase optimization. The inner-loop update finds task-specific parameters based on the support set of each task, and the outer-loop update finds a good parameter initialization point based on the query set.
- **ProtoNet** (Snell et al., 2017) trains a model by building N class prototypes by averaging support samples of each class, and making each query sample and corresponding prototype closer.
- **G-Meta** (Huang & Zitnik, 2020) obtains node embeddings based on the subgraph of each node in episodes, which allows scalable and inductive graph meta-learning.
- **TENT** (Wang et al., 2022b) performs graph meta-learning to reduce the task variance among training episodes via node-level, class-level, and task-level adaptations.
- **GLITTER** (Wang et al., 2022a) aims to learn task-specific structures consisting of support set nodes and their relevant nodes, which have high node influence on them for each meta-training/test task since the given original graph structure is redundant when learning node embeddings in each meta-task.
- **COSMIC** (Wang et al., 2023b) adopts contrastive learning scheme on graph meta-learning to enhance the intra-class generalizability and similarity-sensitive mix-up which generates hard (unseen) node classes for the inter-class generalizability.
- **BGRL** (Thakoor et al., 2022) applies BYOL (Grill et al., 2020) on graphs, so it trains the model by maximizing the agreement between an online embedding and a target embedding of each node, where each embedding is obtained from two differently augmented views.
- **SUGRL** (Mo et al., 2022) simplifies architectures for effective and efficient contrastive learning on graphs, and trains the model by concurrently increasing inter-class variation and reducing intra-class variation.
- **AFGRL** (Lee et al., 2022b) applies BYOL architecture without graph augmentations. Instead of augmentations, AFGRL generates another view by mining positive nodes in the graph in terms of both local and global perspectives.
- **VNT** (Tan et al., 2023) utilizes pretrained transformer-based encoder (Graph-Bert (Zhang et al., 2020)) as a backbone, and adapts to the downstream FSNC task by tuning injected 'virtual' nodes and classifier with given a few labeled samples in the downstream task, then makes prediction on queries with such fine-tuned virtual nodes and classifier.

For meta-learning baselines except for GLITTER and COSMIC, we used a 2-layer GCN (Kipf & Welling, 2017) as the GNN encoder with the hidden dimension chosen from $\{64, 256\}$, and this makes MAML to be essentially equivalent to Meta-GNN (Zhou et al., 2019). Such choice of high hidden dimension size is based on (Chen et al., 2019), which demonstrated that a larger encoder capacity leads to a higher performance of meta-learning model. For each baseline, we tune hyperparameters for each episode generation method. In the case of GLITTER[3] and COSMIC[4], we adopted the settings regarding the GNN encoder (e.g., number of layers and GNN model type) and hyperparameter settings reported in their official source code. For GCL baselines, we also used a 2-layer GCN encoder with the hidden dimension of size 256.

[3]https://github.com/SongW-SW/GLITTER
[4]https://github.com/SongW-SW/COSMIC

For VNT, following the original paper, Graph-Bert (Zhang et al., 2020) is used as the backbone transformer model. As the official code of VNT is not available, we tried our best to reproduce VNT with the settings presented in the paper of VNT and Graph-Bert. For all baselines, Adam (Kingma & Ba, 2015) optimizer is used. The tuned parameters and their ranges are summarized in Table 8. Note that training TENT with NAQ was non-trivial, as it utilizes the entire labeled data (X_{C_b}, Y_{C_b}) to compute cross-entropy loss along with episode-specific losses computed with training episodes per each update. Therefore, when we train TENT with NAQ, the cross-entropy loss was calculated over a single episode. For this reason, the superior performance of NAQ with TENT is especially noteworthy (See Figure 3 and 4) as it outperforms vanilla supervised TENT even with much less data involved in each parameter update during the training phase.

Table 8. Tuned hyperparameters and their range by baselines

Baselines	Hyperparameters and Range
MAML-like (MAML, G-Meta)	Inner step learning rate $\in \{0.01, 0.05, 0.1, 0.3, 0.5\}$, # of inner updates $\in \{1, 2, 5, 10, 20\}$, Meta-learning rate $\in \{0.001, 0.003\}$
ProtoNet-like (ProtoNet, TENT)	Learning rate $\in \{5 \cdot 10^{-5}, 10^{-4}, 3 \cdot 10^{-4}, 5 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}, 5 \cdot 10^{-3}\}$
Self-Supervised (TLP)	Learning rate $\in \{10^{-6}, 10^{-5}, 5 \cdot 10^{-5}, 10^{-4}, 5 \cdot 10^{-4}, 10^{-3}\}$

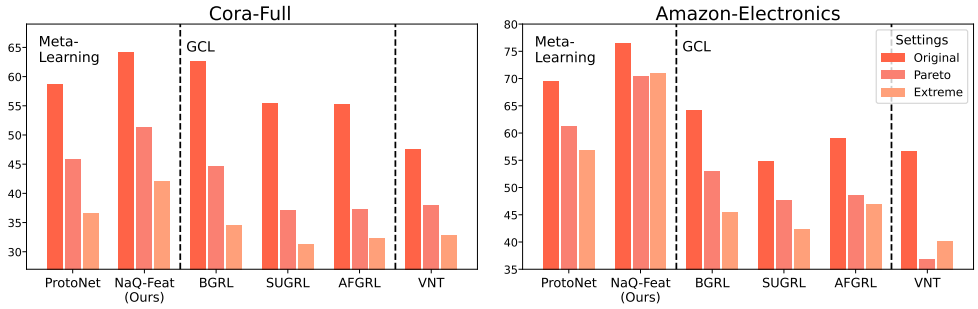


Figure 11. Impact of the class imbalance (5-way 1-shot, NAQ-FEAT base-model: ProtoNet)

Discussion on VNT. Although we tried our best to reproduce VNT, we failed to achieve their reported performance especially on Cora-Full, an evaluation dataset shared by VNT and our paper. This might be due to the random seed, dataset split, or the transformer architecture used in the experiment. However, we conjecture it will also suffer from the inherent bias in data such as class imbalance similar to GCL methods, as graph transformer-based model also learn generic embedding by pretraining on a given graph. As an evidence, in Figure 11, we show the results on Cora-Full and Amazon-Electronics under the same setting used to report results in Figure 1(c). We observe that the performance of VNT deteriorates under class imbalance like GCL methods.

A.7. Model Training with Episodes from NaQ: ProtoNet Example

In this section, we explain how to train ProtoNet (Snell et al., 2017), which is one of the most widely used meta-learning models, with episodes generated by NAQ, as a detailed example of Algorithm 1. Let f_θ be a GNN encoder, $\mathcal{T}_t$ be a generated episode, and $S_{\mathcal{T}_t} = \{(x_{t,i}^{spt}, y_{t,i}^{spt})\}_{i=1}^{N \times K}$ be a randomly sampled support set, then a corresponding query set is generated as $Q_{\mathcal{T}_t} = \{(x_{t,i}^{qry}, y_{t,i}^{qry})\}_{i=1}^{N \times Q} = \text{NAQ}(S_{\mathcal{T}_t})$.

More precisely, we first obtain a prototype $\mathbf{c}_j$ for each class $j \in \{1, \cdots, N\}$ based on the support set $S_{\mathcal{T}}$ as follows[5]:

$$\mathbf{c}_j = \frac{1}{K} \sum\nolimits_{i=1}^{K} f_\theta(x_i^{spt}) \cdot \mathbb{I}\{y_i^{spt} = j\} \tag{4}$$

where $\mathbb{I}\{y_i^{spt} = j\}$ is an indicator function that is equal to 1 only if the label y_i of x_i is j, otherwise 0. Then, the probability of each query $(x^{qry}, y^{qry}) \in Q_{\mathcal{T}}$ belonging to class j is computed as follows:

$$P(y^{qry} = j; x^{qry}) = \frac{\exp(-d(f_\theta(x^{qry}),\ \mathbf{c}_j))}{\sum_{j'} \exp(-d(f_\theta(x^{qry}),\ \mathbf{c}_{j'}))} \tag{5}$$

[5]To remove clutter, we drop the task subscript t from all notations from now on.

where $d(\cdot, \cdot)$ is a distance function. We use Euclidean distance in this work.

Then, the parameter is updated as: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta; qry)$, where η is the learning rate and $\mathcal{L}(\theta; qry)$ is a loss given as:

$$\mathcal{L}(\theta; qry) = -\frac{1}{N \times Q} \sum\nolimits_{(x^{qry}, y^{qry}) \in Q_\mathcal{T}} \log(P(y^{qry} = j; x^{qry})). \tag{6}$$

A.8. Discussion on the Time Complexity of NAQ

In this section, we provide the time analysis of NAQ for generating training episodes. We measured the time spent for the similarity calculation in each dataset, and the time taken to generate all training episodes (i.e., 16,000 in total). The results can be found in Table 9 and 10. Even though the datasets we used are not small, NaQ does not require significant time costs. Moreover, when we use NAQ, the time cost required for similarity calculation and episode generation is at least three times faster than for ordinary supervised methods' training episode generation.

Table 9. Averaged elapsed time over 5 runs in seconds for node-node similarity calculation.

Dataset	NAQ-FEAT	NAQ-DIFF
Amazon-Clothing	1.7769	2.7194
Amazon-Electronics	0.6538	11.0443
Cora-Full	0.0014	1.3207
DBLP	0.0194	9.8653

Table 10. Averaged elapsed time over 5 runs in seconds for generating 16,000 training episodes.

Dataset	NAQ-FEAT	NAQ-DIFF	Supervised
Amazon-Clothing	5.2117	5.0242	64.0850
Amazon-Electronics	6.1301	5.9622	64.4830
Cora-Full	4.9786	4.7484	57.2931
DBLP	5.9689	6.5407	65.8119

However, there are cases where it is challenging to perform similarity calculations at once due to GPU memory problems, if the size of the graph is too large. In such cases, we can calculate the node-node similarity by performing node-wise calculation (NAQ-FEAT) or calculating graph diffusion as a truncated sum (NAQ-DIFF), where this process is required only once for each dataset. Then, a list of Top-k ($k << $ # of nodes) similar nodes for each node can be stored and used by loading them during the episode generation process. For example, in the case of the ogbn-arxiv dataset, which contains about 160,000 nodes, we can calculate the Top-k similar nodes list with a capacity of $\sim$129.20 MiB in a short time of about 150 seconds for NAQ-FEAT and 740 seconds for NAQ-DIFF, for $k = 100$. By using this Top-k similar nodes list, only 2.2713 for NAQ-FEAT and 2.2266 for NAQ-DIFF seconds are spent on average (5 times) for generating total 16,000 training episodes, which is faster than the supervised models' average of 55.9989 seconds in the ogbn-arxiv dataset.

Moreover, in the case of ogbn-products having 2,449,029 nodes, which is a very large-scale dataset, we can calculate such Top-100 similar nodes list in a short time of about 705 seconds by using batched node-node similarity calculation. Thus, our NAQ-FEAT can be scalable to very large graphs having million scale nodes. The following results presented in Table 11 demonstrate the effectiveness of our NAQ-FEAT in a very large-scale dataset.

Table 11. Overall averaged FSNC accuracy (%) with 95% confidence intervals on very large-scale dataset (ogbn-products: having 2,449,029 nodes, 61,859,140 edges, 47 classes (class split: 15/15/17), # of features: 100 (obtained by PCA on bag-of-words features), NAQ-FEAT base-model: ProtoNet)

Dataset	**ogbn-products**	
Baselines	5-way 1-shot	10-way 1-shot
ProtoNet (Sup.)	43.50±1.20	34.19±0.69
COSMIC (Sup.)	OOM	OOM
TLP-BGRL	OOM	OOM
TLP-SUGRL	27.81±0.78	18.72±0.52
NAQ-FEAT (Ours)	**53.82±1.26**	**43.84±0.77**

It is worth noting that *we only need one similarity calculation per dataset*, which makes NAQ practical in reality.

A.9. Regarding Overlapping Queries in NAQ

In this section, we discuss the query overlapping problem of NAQ, where sampled query sets corresponding to each distinct support set have an intersection, which might hurt the FSNC performance of NAQ. Although we tried to prevent this problem by generating only a 1-shot support set as we mentioned in 'Support set generation' process in Section 3.2 (In other words, as each class contains only a 1-shot support node, the number of overlapping queries among classes can be minimized.), such query overlapping problem can happen and might be problematic for NAQ. To assess the severity of this problem, we measured the average query overlap ratio within training episodes generated by NaQ for each dataset. As shown in Table 12 below, query overlap is generally very rare case.

Table 12. Averaged query overlap ratio within 16,000 training episodes generated by NAQ

Datasets	**Amazon-Clothing**		**Amazon-Electronics**		**Cora-Full**		**DBLP**	
N-way	NAQ-FEAT	NAQ-DIFF	NAQ-FEAT	NAQ-DIFF	NAQ-FEAT	NAQ-DIFF	NAQ-FEAT	NAQ-DIFF
5	0.1573%	0.9978%	0.0871%	11.1715%	0.2206%	0.4743%	0.1826%	0.0605%
10	0.3855%	2.0769%	0.2118%	16.9618%	0.5101%	1.0138%	0.4108%	0.1389%
20	0.7834%	4.0358%	0.4457%	21.4706%	1.0221%	2.0151%	0.8559%	0.3054%

However, in the Amazon-Electronics dataset, which has a very low average degree ($\sim$2.06), we observe non-negligible overlap ratio in the case of NAQ-DIFF, which uses graph Diffusion to find queries. To address this issue, we intentionally dropped overlapping queries in training episodes. Table 13 and 14 below show results of the effect of dropping overlapping queries. 'Overlap drop ver.' means that we dropped overlapping queries after the episode generation process of NAQ.

Table 13. Impact of dropping overlapping queries on FSNC performance (%) of NAQ-DIFF in Amazon-Electronics (base-model: ProtoNet)

Amazon-Electronics		
Setting	NAQ-DIFF (Original ver.)	NAQ-DIFF (Overlap drop ver.)
5-way 1-shot	68.56±1.18	**69.77±1.17**
10-way 1-shot	59.46±0.86	**61.98±0.86**
20-way 1-shot	49.24±0.59	**52.15±0.60**

Table 14. Impact of dropping overlapping queries on FSNC performance (%) of NAQ-FEAT in Cora-Full (base-model: ProtoNet)

Cora-Full		
Setting	NAQ-FEAT (Original ver.)	NAQ-FEAT (Overlap drop ver.)
5-way 1-shot	64.20±1.11	63.37±1.08
10-way 1-shot	51.78±0.75	52.32±0.75
20-way 1-shot	40.11±0.45	40.27±0.48

From above results, we can conclude that removing query overlaps is a promising solution when query overlap is not negligible like the case of NAQ-DIFF in Amazon-Electronics (see Table 13). However, when query overlap is negligible, dropping overlapping queries does not bring remarkable improvements (see Table 14).

In summary, the results in Table 12 and Table 14 demonstrate that the query overlapping problem of NAQ is generally negligible in real-world datasets, and the results in Table 13 imply that dropping overlapping queries can be a promising solution for some of the exceptional cases like NAQ-DIFF in Amazon-Electronics dataset.

A.10. g-UMTRA: Augmentation-based Query Generation Method

In this section, we introduce our investigation method named g-UMTRA, utilizes graph augmentation to generate queries. In computer vision, UMTRA (Khodadadeh et al., 2019) tried to apply MAML in an unsupervised manner by generating episodes with image augmentations. With randomly sampled N support set nodes, UMTRA makes a corresponding query set through augmentation on the support set. Inspired by UMTRA (Khodadadeh et al., 2019), we devised an augmentation-based query generation method called g-UMTRA. as an investigation method. g-UMTRA generates query set by applying graph augmentation on the support set. The method overview an be found in Figure 12.

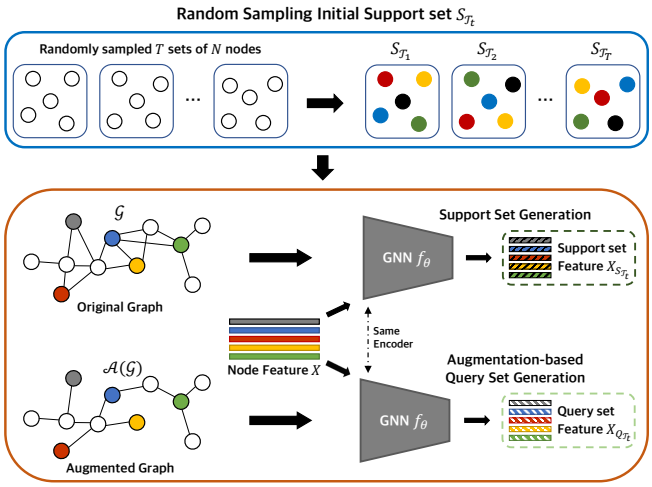


Figure 12. Overview of g-UMTRA. First, we randomly sample T sets of N nodes from the entire graph and assign them distinct labels. Then, we generate support set features by using a GNN encoder with original structures and query set features with augmented structures.

Specifically, we first randomly sample T sets of N nodes from entire graph to generate $\{\mathcal{T}_t\}_{t=1}^{T}$. Then for each task $\mathcal{T}_t$, we generate a N-way support set $S_{\mathcal{T}_t} = \{(x_{t,i}, y_{t,i}) \mid x_{t,i} \in \mathcal{V}\}_{i=1}^{N\times 1}$ with distinct pseudo-labels $y_{t,i}$ for each $x_{t,i}$, and their corresponding query set $Q_{\mathcal{T}_t}$ in the embedding space by applying graph augmentation.

By notating GNN encoder f_θ as $f_\theta(\mathcal{V}; \mathcal{G})$, we can formally describe the query generation process of g-UMTRA as follows:

$$
\begin{aligned}
X_{S_{\mathcal{T}_t}} &= \{\big(f_\theta(x_{t,i}; \mathcal{G}), y_{t,i}\big) \mid (x_{t,i}, y_{t,i}) \in S_{\mathcal{T}_t}\}, \\
X_{Q_{\mathcal{T}_t}} &= \{\big(f_\theta(x_{t,i}; \mathcal{A}(\mathcal{G})), y_{t,i}\big) \mid (x_{t,i}, y_{t,i}) \in S_{\mathcal{T}_t}\},
\end{aligned} \tag{7}
$$

where $f_\theta(x_{t,i}; \mathcal{G})$ is an embedding of node $x_{t,i}$ with the given graph $\mathcal{G}$ and a GNN encoder f_θ, and $\mathcal{A}(\cdot)$ is a graph augmentation function. For $\mathcal{A}(\cdot)$, we can consider various strategies such as node feature masking (DropFeature) or DropEdge (Rong et al., 2020).

Note that g-UMTRA is distinguished from UMTRA in the following two aspects: (1) g-UMTRA can be applied to any existing graph meta-learning methods as it only focuses on episode generation, while UMTRA is mainly applied on MAML. (2) As described in Equation 7, g-UMTRA generates episodes as pair of sets $(X_{S_{\mathcal{T}_j}}, X_{Q_{\mathcal{T}_j}})$ that consist of embeddings. Hence, its query generation process should take place in the training process, since augmentation and embedding calculation of GNNs depend on the graph structure. However, in UMTRA, image augmentation and ordinary convolutional neural networks are applied in instance-level, implying that the episode generation process of UMTRA can be done before training.

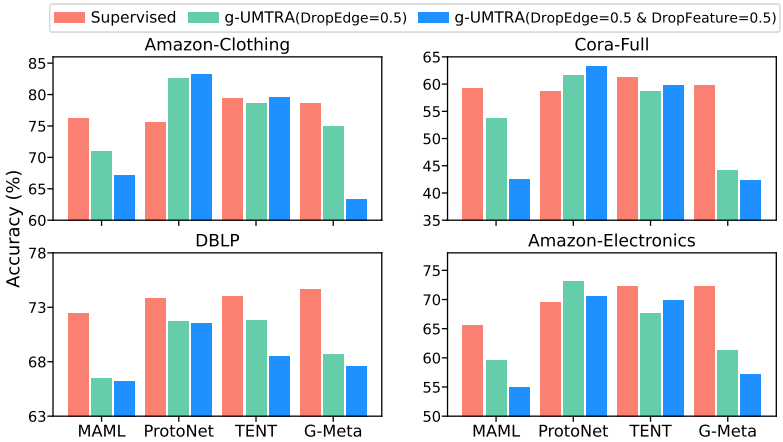


Figure 13. Performance comparison between supervised, g-UMTRA with DropEdge, and g-UMTRA with DropEdge and DropFeature on existing graph meta-learning models (5-way 1-shot).

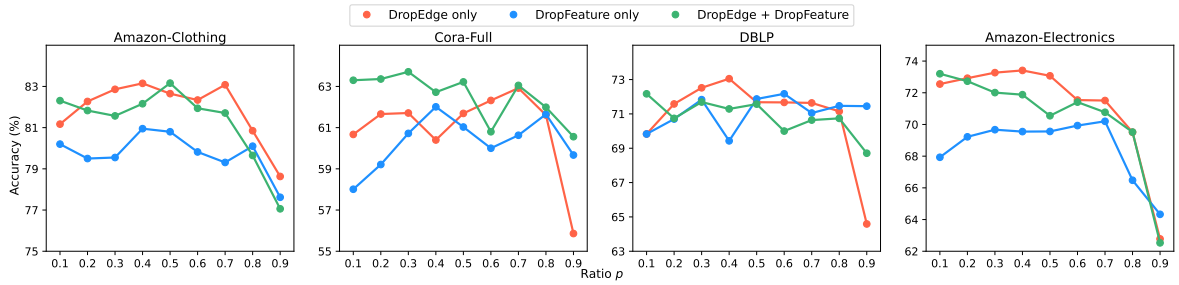


Figure 14. Effect of augmentation function and its strength on g-UMTRA. (5-way 1-shot, base-model: ProtoNet)

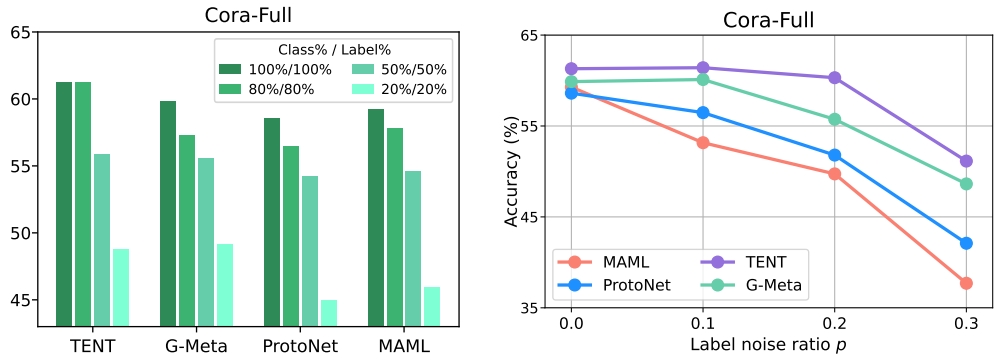
A.10.1. Drawbacks of g-UMTRA

Although g-UMTRA can show remarkable performance with some of base-models like ProtoNet (Snell et al., 2017) (See Figure 13), there are several drawbacks of g-UMTRA that limit its applicability in the real-world settings. First, g-UMTRA requires additional computation of augmented embedding by each update to make query set embeddings, which is time-consuming. Next, g-UMTRA cannot be model-agnostic, since it makes episode within the training phase due to the graph augmentation for the query set generation. Thus, g-UMTRA requires inevitable modification on the training process of some existing models like G-Meta (Huang & Zitnik, 2020) and TENT (Wang et al., 2022b), up-to-date graph meta-learning methods which are developed under the premise of utilizing supervised episodes, having mutually exclusive support set and query set. Lastly, g-UMTRA is also highly sensitive to the augmentation function choice and its strength (See Figure 14), similar to the original UMTRA.

A.11. Additional Experimental Results

Impact of the label-scarcity in Cora-Full. We additionally conducted the experiment about the label-scarcity problem presented in Figure 1(a) in the Cora-Full dataset. Similar to the result shown in Figure 1(a), supervised graph meta-learning methods' FSNC performance decreases as available labeled data and diversity of base classes decrease (See Figure 15(a)).

Impact of the label noise in Cora-Full. We also conducted the experiment regarding the label noise presented in Figure 1(b) in the Cora-Full dataset. Note that as Cora-Full has smaller size than Amazon-Electronics, we selected label noise p ratio from $\{0, 0.1, 0.2, 0.3\}$. As shown in Figure 15(b), similar to the result in Figure 1(b), supervised meta-learning methods' FSNC performance is highly degraded as noise level increases.



(a) Impact of the label-scarcity in Cora-Full (b) Impact of the label noise in Cora-Full

Figure 15. (a): Impact of the label-scarcity on supervised graph meta-learning methods, (b): Impact of the (randomly injected) label noise p on supervised graph meta-learning methods. (5-way 1-shot)

Table 15. Overall averaged FSNC accuracy (%) with 95% confidence intervals on product networks (Full Version)

Dataset		**Amazon Clothing**				**Amazon Electronics**					
Setting		5 way		10 way		5 way		10 way		20 way	
Base Model	Episode Generation	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot
MAML	Supervised	76.13±1.17	84.28±0.87	63.77±0.83	76.95±0.65	65.58±1.26	78.55±0.96	57.31±0.87	67.56±0.73	46.37±0.61	60.04±0.52
	NAQ-FEAT (Ours)	74.07±1.07	86.49±0.86	59.44±0.91	75.99±0.70	59.56±1.17	74.85±1.03	49.03±0.88	70.47±0.73	45.27±0.60	62.36±0.51
	NAQ-DIFF (Ours)	79.30±1.17	86.81±0.82	69.97±0.86	79.74±0.68	62.90±1.18	78.37±0.90	52.23±0.84	68.77±0.76	43.28±0.62	59.88±0.51
ProtoNet	Supervised	75.52±1.12	89.76±0.70	65.50±0.82	82.23±0.62	69.48±1.22	84.81±0.82	57.67±0.85	75.79±0.67	48.41±0.57	67.31±0.47
	NAQ-FEAT (Ours)	83.77±0.96	**92.27±0.67**	76.08±0.81	85.60±0.60	**76.46±1.11**	**88.72±0.73**	68.42±0.86	81.36±0.64	58.80±0.60	**74.60±0.47**
	NAQ-DIFF (Ours)	78.64±1.05	90.82±0.68	71.75±0.81	83.81±0.60	68.56±1.18	84.88±0.83	59.46±0.86	76.73±0.67	49.24±0.59	67.99±0.48
TENT	Supervised	79.46±1.10	89.61±0.70	69.72±0.80	84.74±0.59	72.31±1.14	85.25±0.81	62.13±0.83	77.32±0.67	52.45±0.60	69.39±0.50
	NAQ-FEAT (Ours)	**86.58±0.96**	91.98±0.67	**79.55±0.78**	**86.10±0.60**	76.26±1.11	87.27±0.81	**69.59±0.86**	**81.44±0.61**	59.65±0.60	74.09±0.46
	NAQ-DIFF (Ours)	80.87±1.08	90.53±0.71	72.67±0.82	84.54±0.61	68.14±1.13	83.64±0.80	60.44±0.79	76.03±0.67	51.44±0.58	68.37±0.49
G-Meta	Supervised	78.67±1.05	88.79±0.76	65.30±0.79	80.97±0.59	72.26±1.16	84.44±0.83	61.32±0.86	74.92±0.71	50.39±0.59	65.73±0.48
	NAQ-FEAT (Ours)	85.83±1.03	90.70±0.73	73.45±0.84	82.61±0.66	74.49±1.15	84.68±0.86	61.18±0.83	77.36±0.67	55.35±0.60	69.16±0.51
	NAQ-DIFF (Ours)	82.27±1.10	89.88±0.77	71.48±0.86	82.07±0.63	69.62±1.20	80.87±0.94	58.71±0.80	75.55±0.67	49.06±0.58	67.41±0.47
GLITTER	Supervised	75.73±1.10	89.18±0.74	64.30±0.79	77.73±0.68	66.91±1.22	82.59±0.83	57.12±0.88	76.26±0.67	49.23±0.57	61.77±0.52
	NAQ-FEAT (Ours)	68.24±1.27	76.91±1.00	59.15±0.81	77.19±0.65	64.06±1.16	80.25±0.86	59.31±0.79	74.65±0.67	49.75±0.59	65.30±0.51
	NAQ-DIFF (Ours)	70.24±1.21	82.48±0.83	63.36±1.14	80.41±0.62	65.45±1.22	80.33±0.87	54.96±0.84	71.10±0.72	44.26±0.57	60.20±0.50
COSMIC	Supervised	82.24±0.99	91.22±0.73	74.44±0.75	81.58±0.63	72.61±1.05	86.92±0.76	65.24±0.82	78.00±0.64	58.71±0.57	70.29±0.44
	NAQ-FEAT (Ours)	84.42±1.01	91.73±0.69	73.15±0.78	84.74±0.58	73.98±1.09	87.08±0.75	65.96±0.82	79.11±0.60	**61.05±0.59**	73.73±0.42
	NAQ-DIFF (Ours)	84.40±1.01	91.72±0.69	73.39±0.79	84.82±0.58	74.16±1.08	87.09±0.75	65.95±0.81	79.13±0.60	60.40±0.59	73.75±0.42
TLP	BGRL	81.42±1.05	90.53±0.71	72.05±0.86	83.64±0.63	64.20±1.10	81.72±0.85	53.16±0.82	73.70±0.66	44.57±0.54	65.13±0.47
	SUGRL	63.32±1.19	86.35±0.78	54.81±0.77	73.10±0.63	54.76±1.06	78.12±0.92	46.51±0.80	68.41±0.71	36.08±0.52	57.78±0.49
	AFGRL	78.12±1.13	89.82±0.73	71.12±0.81	83.88±0.63	59.07±1.07	81.15±0.85	50.71±0.85	73.87±0.66	43.10±0.56	65.44±0.48
VNT		65.09±1.23	85.86±0.76	62.43±0.81	80.87±0.63	56.69±1.22	78.02±0.97	49.98±0.83	70.51±0.73	42.10±0.53	60.99±0.50

Table 16. Overall averaged FSNC accuracy (%) with 95% confidence intervals on product networks (Full Version, OOT: Out Of Time, which means that the training was not finished in 24 hours, OOM: Out Of Memory on NVIDIA RTX A6000)

Dataset		**Cora-full**						**DBLP**					
Setting		5 way		10 way		20 way		5 way		10 way		20 way	
Base Model	Episode Generation	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot
MAML	Supervised	59.28±1.21	70.30±0.99	44.15±0.81	57.59±0.66	30.99±0.43	46.80±0.38	72.48±1.22	80.30±1.03	60.08±0.90	69.85±0.76	46.12±0.53	57.30±0.48
	NAQ-FEAT (Ours)	64.64±1.16	74.31±0.94	49.86±0.78	64.88±0.64	38.90±0.46	53.87±0.43	68.49±1.23	77.31±1.08	55.70±0.88	67.94±0.82	44.18±0.53	56.50±0.48
	NAQ-DIFF (Ours)	62.93±1.17	76.48±0.92	50.10±0.83	63.50±0.66	38.09±0.45	54.08±0.41	71.14±1.15	79.47±1.01	59.18±0.91	70.19±0.78	44.94±0.57	58.68±0.47
ProtoNet	Supervised	58.61±1.21	73.91±0.93	44.54±0.79	62.15±0.64	32.10±0.42	50.87±0.40	73.80±1.20	81.33±1.00	61.88±0.86	73.02±0.74	48.70±0.52	62.42±0.45
	NAQ-FEAT (Ours)	64.20±1.11	79.42±0.80	51.78±0.75	68.87±0.60	40.11±0.45	58.48±0.40	71.38±1.17	82.34±0.94	58.41±0.86	72.36±0.73	47.30±0.53	61.61±0.46
	NAQ-DIFF (Ours)	65.30±1.08	79.66±0.79	51.80±0.78	**69.34±0.63**	40.76±0.49	59.35±0.40	73.89±1.15	82.24±0.98	59.43±0.79	72.85±0.76	48.17±0.52	61.66±0.48
TENT	Supervised	61.30±1.18	77.32±0.81	47.30±0.80	66.40±0.62	36.40±0.45	55.77±0.39	74.01±1.20	82.54±1.00	62.95±0.85	73.26±0.77	49.67±0.53	61.87±0.47
	NAQ-FEAT (Ours)	64.04±1.14	78.48±0.79	51.31±0.77	67.09±0.62	40.04±0.48	56.15±0.40	72.85±1.20	80.91±1.00	60.70±0.87	71.98±0.79	47.29±0.53	61.01±0.46
	NAQ-DIFF (Ours)	61.85±1.12	77.26±0.84	49.80±0.76	67.65±0.63	37.78±0.45	56.55±0.41	**76.58±1.18**	**82.86±0.95**	**64.31±0.87**	**74.06±0.75**	**51.62±0.54**	63.05±0.45
G-Meta	Supervised	59.88±1.26	75.36±0.86	44.34±0.80	59.59±0.66	33.25±0.42	49.00±0.39	74.64±1.20	79.96±1.08	61.50±0.88	70.33±0.77	46.07±0.52	58.38±0.47
	NAQ-FEAT (Ours)	65.79±1.21	79.21±0.82	48.90±0.80	63.96±0.61	40.36±0.46	55.17±0.43	70.08±1.24	80.79±0.97	57.98±0.87	71.18±0.75	45.65±0.52	59.38±0.46
	NAQ-DIFF (Ours)	62.96±1.14	77.31±0.87	47.93±0.79	63.18±0.61	37.55±0.46	54.23±0.41	70.39±1.20	80.47±1.03	57.55±0.85	69.59±0.78	44.56±0.52	58.66±0.45
GLITTER	Supervised	55.17±1.18	69.33±0.96	42.81±0.81	52.76±0.68	30.70±0.41	40.82±0.41	73.50±1.25	75.90±1.19	OOT	OOT	OOM	OOM
	NAQ-FEAT (Ours)	62.66±1.12	76.40±0.87	50.05±0.79	67.66±0.61	40.16±0.47	57.13±0.42	64.55±1.18	78.54±1.10	OOT	OOT	OOM	OOM
	NAQ-DIFF (Ours)	54.58±1.14	70.59±0.93	47.62±0.74	64.58±0.65	38.91±0.46	52.70±0.41	63.44±1.21	75.79±1.06	OOT	OOT	OOM	OOM
COSMIC	Supervised	62.24±1.15	73.85±0.83	47.85±0.77	59.11±0.60	42.25±0.43	47.28±0.38	72.34±1.18	80.83±1.03	59.21±0.80	70.67±0.71	49.52±0.51	59.01±0.42
	NAQ-FEAT (Ours)	**66.30±1.15**	**80.09±0.79**	**52.23±0.73**	68.63±0.61	**44.13±0.47**	60.94±0.36	73.55±1.16	82.36±0.94	58.81±0.80	71.14±0.70	50.42±0.52	**64.90±0.43**
	NAQ-DIFF (Ours)	66.26±1.15	80.07±0.79	52.17±0.74	68.95±0.60	44.12±0.47	**60.97±0.37**	73.82±1.16	82.29±0.94	58.81±0.80	71.10±0.70	50.47±0.52	64.78±0.44
TLP	BGRL	62.59±1.13	78.80±0.80	49.43±0.79	67.18±0.61	37.63±0.44	56.26±0.39	73.92±1.19	82.42±0.95	60.16±0.87	72.13±0.74	47.00±0.53	60.57±0.45
	SUGRL	55.42±1.08	76.01±0.84	44.66±0.74	63.69±0.62	34.23±0.41	52.76±0.40	71.27±1.15	81.36±1.02	58.85±0.81	71.02±0.78	45.71±0.49	59.77±0.45
	AFGRL	55.24±1.02	75.92±0.83	44.08±0.70	64.42±0.62	33.88±0.41	53.83±0.39	71.18±1.17	82.03±0.94	58.70±0.86	71.14±0.75	45.99±0.53	60.31±0.45
VNT		47.53±1.14	69.94±0.89	37.79±0.69	57.71±0.65	28.78±0.40	46.86±0.40	58.21±1.16	76.25±1.05	48.75±0.81	66.37±0.77	40.10±0.49	55.15±0.46