Virtual Node Tuning for Few-shot Node Classification

Zhen Tan
Arizona State University
ztan36@asu.edu

Ruocheng Guo
ByteDance Research
ruocheng.guo@bytedance.com

Kaize Ding
Arizona State University
kaize.ding@asu.edu

Huan Liu
Arizona State University
huanliu@asu.edu

ABSTRACT

Few-shot Node Classification (FSNC) is a challenge in graph representation learning where only a few labeled nodes per class are available for training. To tackle this issue, meta-learning has been proposed to transfer structural knowledge from base classes with abundant labels to target novel classes. However, existing solutions become ineffective or inapplicable when base classes have no or limited labeled nodes. To address this challenge, we propose an innovative method dubbed Virtual Node Tuning (VNT). Our approach utilizes a pretrained graph transformer as the encoder and injects virtual nodes as soft prompts in the embedding space, which can be optimized with few-shot labels in novel classes to modulate node embeddings for each specific FSNC task. A unique feature of VNT is that, by incorporating a Graph-based Pseudo Prompt Evolution (GPPE) module, VNT-GPPE can handle scenarios with sparse labels in base classes. Experimental results on four datasets demonstrate the superiority of the proposed approach in addressing FSNC with unlabeled or sparsely labeled base classes, outperforming existing state-of-the-art methods and even fully supervised baselines.

CCS CONCEPTS

• **Computing methodologies** → **Cost-sensitive learning**.

KEYWORDS

graph neural networks, few-shot learning, prompt

ACM Reference Format:
Zhen Tan, Ruocheng Guo, Kaize Ding, and Huan Liu. 2023. Virtual Node Tuning for Few-shot Node Classification. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3580305.3599541

1 INTRODUCTION

With the advent of deep learning, Graph Neural Networks (GNNs) have been proposed for effective graph representation learning with sufficient labeled instances [11, 19, 38, 45]. However, there is a growing interest in learning GNNs with limited labels, which is a prevalent issue in large graphs where manual data collection and labeling is costly [5]. This has led to a proliferation of studies in the field of Few-shot Node Classification (FSNC) [4, 15, 21, 41, 42, 50], which aims to learn fast-adaptable GNNs for unseen tasks with extremely scarce ground-truth labels. Conventionally, FSNC tasks are denoted as N-way K-shot R-query node classification tasks, where N is the number of classes, K is the number of labeled nodes per class, and R is the number of unlabeled nodes per class. The labeled nodes are referred to as the "support set" and the unlabeled nodes are referred to as the "query set" for evaluation.

The current modus operandi, i.e., *meta-learning*, has become a predominate and successful paradigm to tackle the label scarcity issue on graphs [4, 15, 33, 42, 50]. Besides the target node classes (termed as *novel classes*) with few labeled nodes, meta-learning-based methods assume the existence of a set of *base classes*, which is disjoint with the novel classes set and has substantial labeled nodes in each class to sample a number of meta-tasks, or episodes, to train the GNN model while emulating the N-way K-shot R-query task structure. This emulation-based training has been proved helpful for fast adaptation to target FSNC tasks [21, 41]. Despite astonishing breakthroughs having been made, Tan et al. [34] firstly points out that those meta-learning-based methods suffer from the piecemeal graph knowledge issue, which implies that only a small portion of nodes are involved in each episode, thus hindering the generalizability of the learned GNN models regarding unseen novel classes. Additionally, the assumption of the existence of substantially labeled base classes may not be feasible for real-world graphs [35]. In summary, while meta-learning is a successful method for FSNC tasks, it has limitations in terms of effectiveness and applicability.

Considering these limitations in the existing efforts, in this work, we first generalize the traditional definition of FSNC tasks to cover more real-world scenarios where there could be limited or even no labels even in base classes. We first start with the most challenging setting where no available labeled nodes exist in base classes. To facilitate sufficient training, we choose *Graph Transformers* (GTs) [2, 49] as the encoder to learn representative node embeddings. Recently, large transformer-based [37] models have thrived in various domains, such as languages [3], images [6], as well as graphs [14]. The number of parameters of GTs can be much larger than traditional GNNs by orders of magnitude, which has shown unique advantages in modeling graph data and acquiring structural knowledge [2, 49]. Furthermore, pretrained in an unsupervised manner, GTs can learn from a large number of unlabeled nodes by enforcing the model to learn from pre-defined pretext tasks (e.g. masked link restoration, masked node recovery, etc.) [14, 49]. In other words, no node label information from base classes is

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599541

needed for obtaining pretrained GTs enriched with topological and semantic knowledge. However, our experiments show that directly transferring node embedding from GTs and fine-tuning the GT encoder on the support set will lead to unsatisfactory performance. This is because directly transferring node embeddings neglects the inherent gap between the training objective of the pretexts and that of the downstream FSNC tasks. Also, naively fine-tuning with the few labeled nodes will lead to severe overfitting. Both these two factors can render the transferred node embeddings sub-optimal for target FSNC tasks. Accordingly, to elicit the learned substantial prior graph knowledge from GTs with only a few labels from each target task, we propose a method, *Virtual Node Tuning* (VNT), that can efficiently modulate the GTs to customize the pretrained node embeddings for different FSNC tasks.

Recent advancements in natural language processing (NLP) have led to the emergence of a new technique called "prompting" for adapting large-scale transformer-based language models to new few-shot or zero-shot tasks [23]. It refers to prepending language instructions to the input text to guide those language models to better *understand* the new task and give more tailored *replies*. However, such a technique cannot be straightforwardly applied to GTs due to the significant disparity between graphs and texts. Given the symbolic nature of graph data, it is infeasible and counter-intuitive to manually design semantic prompts like human languages for each target FSNC task. Inspired by more recent works [16, 22], instead of manually creating prompts in the raw graph data space (e.g., nodes and edges), we propose to inject a set of continuous vectors as task-specific virtual nodes in the node embedding space to function as *soft prompts* to elicit the substantial knowledge contained in the learned GTs. During the fine-tuning phase, these prompts can be optimized via the few labeled nodes from the support set in each FSNC task. This simple tuning with virtual node prompts can modulate the learned node embeddings according to the context of the FSNC task. Moreover, for scenarios where sparsely labeled nodes exist in base classes, we propose to reformulate the problem by assuming the presence of a few *source* FSNC tasks within the base class label space. Meanwhile, we find that initializing the prompt of an FSNC task as the prompt of a previously learned FSNC task can potentially yield positive transfer. Based on this observation, we design a novel *Graph-based Pseudo Prompt Evolution* (GPPE) module, which performs a prompt-level meta-learning to selectively transfer knowledge learned from source FSNC tasks to target FSNC tasks. This module has demonstrated promising improvement for VNT and scales well to conditions where node labels are highly scarce, i.e., very few source tasks exist.

Notably, the proposed framework is fully automatic and requires no human involvement. By only retraining a small prompt tensor and a simple classifier, and recycling a single GT for all downstream FSNC tasks, our method significantly reduces storage and computation costs per task. Through extensive experimentation, we have demonstrated the effectiveness of the VNT-GPPE method in terms of both accuracy and efficiency. We hope this work can provide a new promising path forward for *few-shot Node Classification* (FSNC) tasks. In summary, our main contributions include:

- **Problem Generalization:** We relax the assumption in conventional FSNC tasks to cover scenarios where there could be none or sparsely labeled nodes in base classes.
- **Framework Proposed:** We propose a simple yet effective framework, *Virtual Node Tuning* (VNT), that does not rely on any label from base classes. We inject virtual nodes in the embedding space which function as soft prompts to customize the pretrained node embeddings for each FSNC task. To extend the framework for scenarios where sparely labeled nodes in base classes are available, we further design a *Graph-based Pseudo Prompt Evolution* (GPPE) module that transfers prompts learned from base classes to target downstream FSNC tasks.
- **Comprehensive Experiment:** We conduct extensive experiments on four widely-used real-world datasets to show the effectiveness and applicability of the proposed framework. We find that VNT achieves competitive performance even no labeled nodes from base classes are utilized. Given sparsely labeled base classes, VNT-GPPE outperforms all the baselines even if they are given fully labeled base classes. Further analysis also indicates that VNT considerably benefits from prompt ensemble.

2 PROBLEM FORMULATION

In this work, we focus on few-shot node classification (FSNC) tasks on a single graph. Formally, given an attributed network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X}) = (\boldsymbol{A}, \boldsymbol{X})$, where $\boldsymbol{A} = \{0,1\}^{V \times V}$ is the adjacency matrix representing the network structure, $\boldsymbol{X} = [\boldsymbol{x}_1; \boldsymbol{x}_2; ...; \boldsymbol{x}_V]$ represents all the node features, $\mathcal{V}$ denotes the set of vertices $\{v_1, v_2, ..., v_V\}$, and $\mathcal{E}$ denotes the set of edges $\{e_1, e_2, ..., e_E\}$. Specifically, $\boldsymbol{A}_{j,k} = 1$ indicates that there is an edge between node v_j and node v_k; otherwise, $\boldsymbol{A}_{j,k} = 0$. The few-shot node classification problem assumes the existence of a series of node classification tasks, $\mathcal{T} = \{\mathcal{T}_i\}_{i=1}^{I}$, where $\mathcal{T}_i$ denotes the given dataset of a task, I denotes the number of such tasks. Traditional FSNC tasks assume that those tasks are formed from target *novel classes* (i.e. $\mathbb{C}_{novel}$), where only a few labeled nodes are available per class, and there exists a disjoint set of *base classes* (i.e. $\mathbb{C}_{base}$, $\mathbb{C}_{base} \cap \mathbb{C}_{novel} = \emptyset$) on the graph where substantial labeled nodes are accessible during training. Next, we first present the definition of an N-way K-shot R-query node classification task as follows:

Definition 1. ***N-way K-shot R-query Node Classification:*** *Given an attributed graph $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ with a specified node label space $\mathbb{C}$, $|\mathbb{C}| = N$. If for each class $c \in \mathbb{C}$, there are K labeled nodes (i.e. support set $\mathbb{S}$) as references and another R nodes (i.e. query set $\mathbb{Q}$) for prediction, then we term this task as an N-way K-shot R-query Node Classification task.*

Then, the traditional few-shot node classification problem can be defined as follows:

Definition 2. ***Traditional Few-shot Node Classification:*** *Given an attributed graph $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ with a disjoint node label space $\mathbb{C} = \{\mathbb{C}_{base}, \mathbb{C}_{novel}\}$. Substantial labeled nodes from $\mathbb{C}_{base}$ are available for sampling an arbitrary number of N-way K-shot R-query Node Classification tasks for training. The goal is to perform N-way K-shot R-query Node Classification for tasks sampled from $\mathbb{C}_{novel}$.*

However, the assumption of the existence of substantial labeled nodes in the base classes could be untenable for real-world graphs. For example, all the classes on a given graph may only have a few labeled nodes. Considering this limitation, in this paper, we

generalize the definition of FSNC and reformulate it according to the label sparsity within base classes. It is formulated as follows:

Definition 3. ***General Few-shot Node Classification:*** *Given an attributed graph $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ with a disjoint node label space $\mathbb{C} = \{\mathbb{C}_{base}, \mathbb{C}_{novel}\}$. For $\mathbb{C}_{base}$, there are M N-way K-shot R-query Node Classification tasks for training. The goal is to perform N-way K-shot R-query Node Classification for tasks sampled from $\mathbb{C}_{novel}$.*

Note that the key difference of the general FSNC compared to the traditional counterparts lies in the introduced parameter M. Since $M, N, K, R \ll |V|$, the labels in base classes can be very sparse and the value of M determines the label sparsity in the base classes. For example, if $M = 0$, then the training phase actually provides no label from base classes, so the training procedure should be fully unsupervised. If $M \neq 0$, that means we have a collection of tasks with labeled nodes in base classes. In practice, we achieve this by random sampling M N-way K-shot R-query node classification tasks from base classes. We term those M tasks as *source tasks* and the tasks during final evaluation as *target tasks*. Particularly, if M is a very large number, then the general FSNC will scale to the traditional FSNC, which most of the existing works are trying to address. Conversely, if M is a relatively small number (e.g. 48), this signifies the labels provided in the base classes are very sparse, which is the usual scenario for real-world applications.

Our paper is the first to propose this more general problem formulation for FSNC tasks. To tackle this problem, in this work, we propose a novel framework named *Virtual Node Tuning* that achieves promising performance when no source task exists (i.e., $M = 0$). To cover more real-world scenarios (i.e., $M \neq 0$), we further design a *Prompt Transferring* mechanism via *Graph-based Pseudo Prompt Evolution* that performs a prompt-level meta-learning to effectively transfer generalizable knowledge from source tasks to target downstream FSNC tasks.

3 METHODOLOGY

3.1 Preliminary: Graph Transformers

Graph Transformers (GTs) [2, 30, 49] are Graph Neural Networks (GNNs) based on transformer [37] without relying on convolution or aggregation operations. Following BERT [3] for large-scale natural language modeling, a D-layer GT is used to project the node attribute $\boldsymbol{x}_j$ of each node v_j ($\forall j \in \mathbb{N}, 1 \leq j \leq V$) into the corresponding embedding $\boldsymbol{e}_j$. GTs usually have much more parameters than traditional GNNs and are often trained in a self-supervised manner, without the need for substantial gold-labeled nodes. For the sake of generality, we choose two simplest and most universally-used pretext tasks, *node attribute reconstruction* and *structure recovery*, to pretrain the GT encoder [2, 49]. An exhaustive discussion of methods for pretraining GTs is out of the scope of this paper, please see more details for GT pretraining in Appendix A. Then, with a pretrained GT, each node v_j is projected into a F-dimensional embedding space. With both node attribute and topology (or position) structure considered, the embedding matrix of all nodes in the graph $\mathcal{G}$ is:

$$\boldsymbol{E}^0 = [\boldsymbol{e}_1^0; ...; \boldsymbol{e}_j^0; ...; \boldsymbol{e}_V^0] = Embed(\mathcal{G}) = Embed(\boldsymbol{X}, \boldsymbol{A}) \in \mathbb{R}^{V \times F}, \tag{1}$$

where n is the number of nodes in the given graph $\mathcal{G}$ and V is the embedding size. Then, the node embeddings $\boldsymbol{E}^{d-1}$ computed by the $d-1$-th layer are fed into the following transformer layer L^d ($\forall d \in \mathbb{N}, 1 \leq d \leq D$) to get more high-level node representations, which can be formulated as:

$$\boldsymbol{E}^d = [\boldsymbol{e}_1^d; ...; \boldsymbol{e}_j^d; ...; \boldsymbol{e}_V^d] = L^d(\boldsymbol{E}^{d-1}) \in \mathbb{R}^{V \times F}. \tag{2}$$

Conventionally, to adapt the pretrained GT to different downstream tasks, further fine-tuning of the GT on the corresponding datasets [2, 30, 49] is performed. However, according to our experiments in Section 4, this vanilla approach suffers from the following limitations when applied to FSNC tasks: **(1)** The number of labeled nodes for each FSNC task is very limited (usually less than 5 labels), making the fine-tuned GT highly overfit on them and hard to generalize to query set. **(2)** This method neglects the inherent gap between the training objective of the pretext tasks and that of the downstream FSNC tasks, rendering the transferred node embeddings sub-optimal for the target FSNC tasks. **(3)** For every new task, all the parameters of GT models need to be updated, making the model hard to converge and greatly raising the cost to apply GTs to real-world applications. **(4)** GTs are generally pretrained in an unsupervised manner. How to utilize the labels (which can be sparse) in base classes to extract generalizable graph patterns for GTs remains unresolved. This work is the first to propose a simple yet effective and efficient *prompting* method for GTs to tackle the four aforementioned limitations.

3.2 Virtual Node Tuning

Since the GT encoder is pretrained on the given graph in a self-supervised manner, it does not require or utilize any label information from base classes. This characteristic fits the FSNC tasks where no source task exists, i.e., $M = 0$. Then, in this section, we introduce the proposed *Virtual Node Tuning* (VNT) method which effectively utilizes the limited few labeled nodes in the *support set* $\mathbb{S}$ from the target label space $\mathbb{C}$ to customize the node representations from the pretrained GT for each specific FSNC task.

We introduce an extra set of p randomly initialized continuous trainable vectors with the same embedding size F, *i.e. prompt*, denoted as $\boldsymbol{P} = [\boldsymbol{p}_1; ...; \boldsymbol{p}_p; ...; \boldsymbol{p}_P]$, ($\boldsymbol{p}_p \in \mathbb{R}^F$). Our virtual node tuning strategy is simple to implement. We fix the pretrained weights of the GT encoder during fine-tuning while keeping the prompt parameter $\boldsymbol{P}$ trainable, and we concatenate this prompt with the pretrained node embedding right after the embedding layer and feed it to the first transformer layer of the GT. The injected prompts can be viewed as task-specific *virtual nodes* that help modulate the pretrained node representations and elicit the learned substantial knowledge from the pretrained GT for different target FSNC tasks. In such a manner, our approach allows the frozen large transformer layers to update the intermediate-layer node representations for different tasks, as contextualized by those virtual nodes (more detailed discussion about the effect from those virtual nodes on target FSNC tasks is presented in Section 4.3 and 4.5.1):

$$[\boldsymbol{E}^1 || \boldsymbol{Z}^1] = L^1([\boldsymbol{E}^0 || \boldsymbol{P}]) \in \mathbb{R}^{(V+P) \times F}, \tag{3}$$

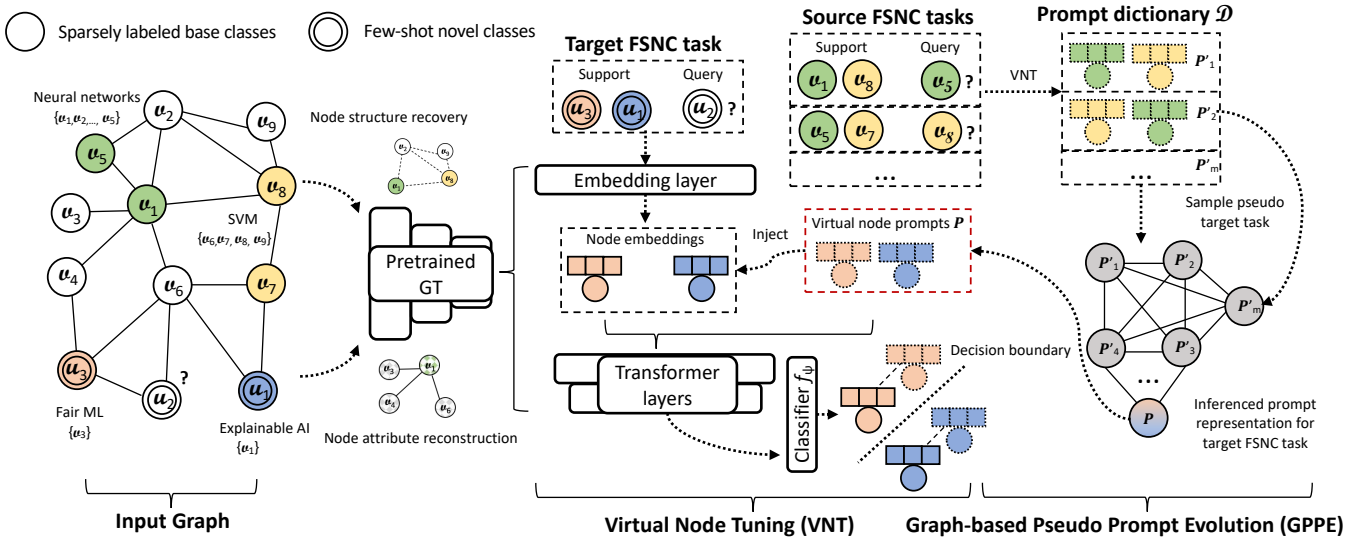


Figure 1: The illustration of the proposed framework, VNT-GPPE. Colors indicate different classes (e.g., *Neural Networks, SVM, Fair ML, Explainable AI*). Especially, white nodes mean labels of those nodes are unavailable. Different types of nodes indicate if nodes are from base classes or novel classes. Note that during VNT and GPPE, the parameters of the GT, including the embedding layer and transformer layers, are fixed.

where || denotes the concatenation operator. Then we feed the learned node representations and the prompt to the following transformer layers L^d ($\forall d \in \mathbb{N}, 2 \leq d \leq D$), which is formulated as:

$$[\boldsymbol{E}^d||\boldsymbol{Z}^d] = L^d([\boldsymbol{E}^{d-1}||\boldsymbol{Z}^{d-1}]) \in \mathbb{R}^{(V+P)\times F}. \tag{4}$$

With the modulated node representations, we can get the predicted label for any node v_j by applying a simple classifier, f_ψ (e.g. SVM, Logistic Regression, shallow MLP, etc.):

$$y = f_\psi(\boldsymbol{e}_j^D). \tag{5}$$

Then, for each target N-way K-shot FSNC task $\mathcal{T}_i = \{\mathbb{S}_i, \mathbb{Q}_i\}$, we can predict labels for all the few labeled nodes in the support set $\mathbb{S}_i$, and calculate the Cross-entropy loss, $\mathcal{L}_{CE}$, to update the prompt parameters $\boldsymbol{P}$ and the simple classifier f_ψ. This optimization procedure can be formulated as:

$$\boldsymbol{P}, \psi = \arg\min_{\boldsymbol{P},\psi} \mathcal{L}_{CE}(\mathbb{S}_i; \boldsymbol{P}, \psi) \tag{6}$$

Finally, following the same procedure, we use the fine-tuned prompt $\boldsymbol{P}$, classifier f_ψ, and node representations from the pretrained GT to predict labels for unlabeled nodes in the query set $\mathbb{Q}_i$. It is notable that the parameters of the pretrained GT are frozen throughout the VNT process and are recycled for all downstream FSNC tasks. In other words, to adapt to a new FSNC task, we only need to train a small prompt tensor $\boldsymbol{P}$ to modulate the intermediate-layer node representations to be customized by the few labeled nodes, which is **computationally similar to training a very shallow MLP**. This signifies that the proposed VNT method requires a low per-task storage and computation cost to retain its effectiveness.

According to our experiments, the proposed VNT method can still achieve competitive performance even if no label in base classes is used. Furthermore, since during each downstream FSNC task, those virtual node prompts are tuned via conditioning on the same frozen GT backbone, we interpret the learned task prompts as task embeddings to form a semantic space of FSNC tasks. We give explanations in Section 4.5.1. Based on this interpretation, we propose a novel *Graph-based Prompt Transfer* method to tackle scenarios where labeled nodes are available in the base classes.

3.3 Prompt Transferring via Graph-based Pseudo Prompt Evolution

For real-world scenarios, there could exist sparsely labeled nodes in base classes, i.e., a few source tasks exist, or $M \neq 0$. On the other hand, we find that first training a prompt on one FSNC task, and then using the resulting prompt as the initialization for the prompt of another task could outperform tuning the virtual node prompt from scratch. We give a motivating example in Section 4.5.1. Inspired by this phenomenon, in this section, we propose a novel *Graph-based Pseudo Prompt Evolution* (GPPE) mechanism that transfers the generalizable knowledge within tasks from base classes to target downstream FSNC tasks. The motivation behind prompt transferring is that, since all the source tasks and target tasks are sampled from the same input graph $\mathcal{G}$, incorporating context from all the individual source tasks will likely yield positive transfer. The details of GPPE are as follows.

To start with, as discussed in Section 2, we assume that, for all the nodes in base classes, there exists a subset of nodes that can form M N-way K-shot R-query node classification tasks, termed as *source tasks*. Note that $M, N, K, R \ll |V|$, so the labels can be very sparse and the value of M determines the label sparsity in the base classes. In practice, we achieve this by random sampling M FSNC tasks from base classes. Without more explanation, we use $M = 48$, $R = 10$ as default for experiments. Next, following the same procedure in Section 3.2, we first pretrain the GT encoder on the whole graph in

an unsupervised manner, then we add virtual node prompts for each source task, and finally, those prompts are tuned on such M source tasks. In this way, we obtain M prompts, each of which can be interpreted as a *task embedding*. Notably, according to equation 6, while performing VNT on the M source tasks, we only use the support set to optimize the prompt. These M prompts are stored as a *prompt dictionary* $\mathcal{D} = [\boldsymbol{P}'_1; ...; \boldsymbol{P}'_m; ...; \boldsymbol{P}'_M] \in \mathbb{R}^{M\times P\times F}$ for transferring knowledge to target FSNC tasks. The required space to store this dictionary is $O(M{\cdot}P{\cdot}F)$, and as M, P, F are small constants, storing this dictionary will not take much extra space, compared to the storage for the weights of the GT or the node embeddings. To further reinforce positive transfer from source tasks to target tasks, we propose a *Prompt Evolution* module to refine those learned representations of virtual node prompts on each target FSNC task based on the task embeddings of all source FSNC tasks. We propose to use a fully connected Graph Attention Network (GAT) [38] to model the relations between these prompts and propagate context knowledge from all source FSNC tasks, where all the task-specific prompts can be regarded as the nodes in the GAT model. We choose GAT as the prompt evolution module for its desired properties: GAT can be inductive and is permutation invariant to the order of learning from source tasks.

For training the prompt evolution module, we draw inspiration from meta-learning [10], where a small classification task is constructed episodically to mimic the test scenarios and enable learning on the meta-level beyond a specific task. Similarly, in each episode, we randomly select one task $\mathcal{T}'_m = \{\mathbb{S}'_m, \mathbb{Q}'_m\}$ as a pseudo target FSNC task, and the rest are still regarded as source tasks to extract transferable knowledge. The prompt evolution module will be trained through a number of episodes till its convergence. As the episodes iterate through the prompt dictionary $\mathcal{D}$, the prompt evolution module learns to refine all the prompt representation within $\mathcal{D}$, and simultaneously, learns to adapt to a target FSNC task given a set of source tasks. Here, we illustrate the detailed learning procedure for one episode. We first compute a relation coefficient $c_{m,k}$ between the prompt of the m-th pseudo target FSNC task $\mathcal{T}'_m$ and the k-th prompts for a source task in the dictionary $\mathcal{D}$. The coefficient is calculated based on the following kernel function:

$$c_{m,k} = \langle \boldsymbol{\theta}(\boldsymbol{P}'_m), \boldsymbol{\theta}(\boldsymbol{P}'_k) \rangle, \tag{7}$$

where $\boldsymbol{\theta}$ is a shallow MLP that projects the original prompts to a new metric space. Let $\langle \cdot\ ,\ \cdot \rangle$ denote a similarity function. In this paper, we use cosine similarity for its simplicity and effectiveness. We then normalize all the coefficients with the softmax function to get the final attention weights corresponding to the prompt $\boldsymbol{P}'_m$ of the current pseudo target FSNC task:

$$a_{m,k} = \frac{\exp\left(c_{m,k}\right)}{\exp(\sum_{h=1}^{|\mathcal{D}|} c_{m,h})}. \tag{8}$$

Based on the learned coefficients, the GAT model aggregates information from all the prompts learned from the source tasks in the graph and fuses it with $\boldsymbol{P}'_m$, the original prompt representation from a pseudo target task, to obtain a refined prompt $\tilde{\boldsymbol{P}}'_m$:

$$\tilde{\boldsymbol{P}}'_m = \boldsymbol{P}'_m + (\sum_{k=1}^{|\mathcal{D}|} a_{m,k} \boldsymbol{L} \boldsymbol{P}'_k), \tag{9}$$

where $\boldsymbol{L}$ denotes the weight matrix of a linear transformation. Then, with the refined prompt representation $\tilde{\boldsymbol{P}}'_m$ and the simple classifier f_ψ, we use it to predict the labels of nodes in the corresponding query set $\mathbb{Q}'_m$. Based on the predicted labels, Cross-entropy loss is adopted to update the prompt evolution module:

$$\boldsymbol{\theta}, \boldsymbol{L}, \psi = \arg\min_{\boldsymbol{\theta}, \boldsymbol{L}, \psi} \mathcal{L}_{CE}(\mathbb{Q}'_m, \tilde{\boldsymbol{P}}'_m; \boldsymbol{\theta}, \boldsymbol{L}, \psi). \tag{10}$$

Once the prompt evolution module is learned, we freeze the parameters in the module and deploy it for any target FSNC task $\mathcal{T}_i = \{\mathbb{S}_i, \mathbb{Q}_i\}$ to get the refined prompt representation $\tilde{\boldsymbol{P}}$. Next, we train a separate simple classifier f_ψ for final predictions:

$$\psi = \arg\min_{\psi} \mathcal{L}_{CE}(\mathbb{S}_i, \tilde{\boldsymbol{P}}; \psi). \tag{11}$$

The proposed GPPE module also has a similar parameter number as a shallow MLP, and we find that the proposed GPPE can be learned even if M, the number of source tasks, is very small. We give the analysis on the effect of M in Fig. 5 in Section 4.5.3. The results show that GPPE improves the performance of VNT even with a very small number of source FSNC tasks (e.g., $M = 8$). The process of our framework is illustrated in Fig. 1.

4 EXPERIMENTAL STUDY

4.1 Experimental Settings

We conduct systematic experiments to compare the proposed VNT method with the baselines on the few-shot node classification task. In this work, we consider two categories of baselines, i.e., *meta-learning* based methods, graph contrastive learning (GCL) based *Transductive Linear Probing* (TLP) methods [35], and *prompting methods* on graphs. For meta-learning, we test typical methods (fully-supervised) including: **ProtoNet** [31], **MAML** [10], **Meta-GNN** [51], **G-Meta** [15], **GPN** [4], **AMM-GNN** [41], and **TENT** [42]. For GCL-based TLP methods, we evaluate TLP with self-supervised GCL methods including: **MVGRL** [12], **GraphCL** [48], **GRACE** [52], **BGRL** [36], **MERIT** [18], and **SUGRL** [27]. For prompting methods on graphs, we evaluate **GPPT** [32] and **Graph Prompt** [25]. For those GCL-based methods and the proposed VNT, we choose Logistic Regression as the classifier f_ψ. For comprehensive studies, we report the results of those methods on four prevalent real-world graph datasets: `CoraFull` [1], `ogbn-arxiv` [13], `Cora` [46], `CiteSeer` [46]. Each dataset is a graph that contains a considerable number of nodes. This ensures that the evaluation involves various tasks for a more comprehensive evaluation. A detailed description of those datasets is provided in Appendix B, with their statistics and class splits in Table 5 in Appendix B. For explicit comparison, we compare our method with all the baselines under various N-way K-shot 10-query settings. The default values of the dictionary size M and the query set size R are 48 and 10, respectively.

4.2 Comparable Study

Table 1 presents the performance comparison of all the methods on the few-shot node classification task. Specifically, we present results under four different few-shot settings for a more comprehensive comparison: 5-way 1-shot, 5-way 5-shot, 2-way 1-shot, and 2-way 5-shot. We choose the average classification accuracy and the 95% confidence interval over 5 repetitions with different random seeds

Table 1: The overall comparison between the proposed VNT method and meta-learning or self-supervised GCL-based TLP methods under different settings. Accuracy (↑) and confident interval (↓) are in %. The best results are bold, and the second best results in each category of methods are underlined. OOM denotes the out-of-memory issue.

Dataset	CoraFull		Ogbn-arxiv		CiteSeer		Cora	
Setting	5-way 1-shot	5-way 5-shot	5-way 1-shot	5-way 5-shot	2-way 1-shot	2-way 5-shot	2-way 1-shot	2-way 5-shot
MAML	22.63±1.19	27.21±1.32	27.36±1.48	29.09±1.62	52.39±2.20	54.13±2.18	53.13±2.26	57.39±2.23
ProtoNet	32.43±1.61	51.54±1.68	37.30±2.00	53.31±1.71	52.51±2.44	55.69±2.27	53.04±2.36	57.92±2.34
Meta-GNN	55.33±2.43	70.50±2.02	27.14±1.94	31.52±1.71	56.14±2.62	67.34±2.10	65.27±2.93	72.51±1.91
GPN	52.75±2.32	72.82±1.88	37.81±2.34	50.50±2.13	53.10±2.39	63.09±2.50	62.61±2.71	67.39±2.33
AMM-GNN	58.77±2.32	75.61±1.78	33.92±1.80	48.94±1.87	54.53±2.51	62.93±2.42	65.23±2.67	82.30±2.07
G-Meta	60.44±2.48	75.84±1.70	31.48±1.70	47.16±1.73	55.15±2.68	64.53±2.35	67.03±3.22	80.05±1.98
TENT	55.44±2.08	70.10±1.73	48.26±1.73	61.38±1.72	62.75±3.23	72.95±2.13	53.05±2.78	62.15±2.13
MVGRL	59.91±2.39	76.76±1.63	OOM	OOM	64.45±2.77	80.25±1.82	71.17±3.04	89.91±1.44
GraphCL	64.20±2.56	83.74±1.46	OOM	OOM	73.51±3.09	92.38±1.24	73.50±3.18	92.35±1.30
GRACE	66.69±2.26	84.06±1.43	OOM	OOM	69.85±2.75	85.93±1.57	69.13±2.69	88.68±1.37
BGRL	43.83±2.11	70.44±1.62	36.76±1.74	53.44±0.36	54.32±1.63	70.50±2.11	60.14±2.33	79.86±1.92
MERIT	73.38±2.25	87.66±1.43	OOM	OOM	64.53±2.81	90.32±1.66	67.67±2.99	95.42±1.21
SUGRL	**77.35±2.20**	83.96±1.52	60.04±2.11	77.52±1.45	**77.34±2.83**	86.32±1.57	82.35±2.20	92.22±1.15
GPPT	62.35±2.34	73.68±2.24	40.36±1.68	51.68±1.92	68.93±2.20	82.53±1.86	70.32±1.86	85.58±1.73
Graph Prompt	72.45±2.08	81.29±2.36	44.58±1.84	75.62±1.96	69.85±2.26	85.26±1.78	78.65±1.98	89.38±1.96
VNT (**Ours.**)	68.50±2.13	84.56±2.15	50.40±1.97	74.91±1.87	70.60±2.15	86.23±1.75	84.50±1.94	90.50±1.55
VNT-GPPE (**Ours.**)	76.68±2.25	**88.75±2.07**	**61.34±1.86**	**79.93±1.69**	75.85±2.45	**93.46±1.72**	**88.62±2.12**	**95.65±1.51**

as the evaluation metrics. For each repetition, we sample 100 FSNC tasks for evaluation and calculate the evaluation metrics. From Table 1, we obtain the following observations:

- Generally speaking, for both TLP and the proposed GT, **self-supervised pretraining can outperform the meta-learning-based method**. However, one most recent pretraining method, BGRL, when transferred for downstream FSNC tasks, shows surprisingly frustrating performance. This further validates **the impact from the gap of training objective** between pretexts and target FSNC tasks. The pretext of BGRL minimizes the *Mean Square Error* of the original node representation and its slightly perturbed counterpart but does not enforce the model to discriminate between different nodes as the other GCL baselines do. The objective of this pretext deviates more from the downstream FSNC tasks, thus leading to worse results. We further show the impact of this in ablation studies in Section 4.3.
- Even **without any label information from base classes**, the proposed method, VNT, outperforms meta-learning-based methods and most GCL-based TLP baselines. This demonstrates the superiority of the proposed VNT in terms of accuracy. The pretrained GT has learned substantial prior knowledge and the injected virtual node prompts effectively elicit the knowledge for different downstream FSNC tasks.
- Given **a few source FSNC tasks**, the proposed method, VNT-GPPE, consistently outperforms all the baselines, including existing prompt methods for graphs. This further validates that GPPE effectively generalizes the knowledge learned from the few source tasks to target tasks, yielding positive transfer.
- Compared to all the baselines, the proposed VNT-GPPE method is **more robust to extremely scarce label scenarios**, i.e, the number of labeled nodes in the support set K equals 1. The performance degradation resulting from decreasing the number of shots K is significant for all the methods. Smaller K makes the encoder or the classifier more prone to overfitting, thus leading to worse generalization to query sets. In contrast, the proposed method injects virtual nodes into the model, which have separate learnable embeddings for different FSNC tasks. This implies that our method implicitly performs adaptable data augmentation for the few labeled nodes, which makes our framework more robust to tasks with extremely scarce labeled nodes. Also, the **improvement of involving GPPE is more significant when K is extremely small**. This shows that transferring knowledge from source tasks helps to mitigate the overfitting on novel tasks. Further analysis and explanation are given in Section 4.5.1.

4.3 Ablation Study

In this subsection, we conduct ablation studies to investigate the effectiveness of different components, i.e., VNT and GPPE, of the proposed framework. We consider both cases: when the encoder is frozen and when it is not frozen. We present the results of experiments on the Cora and Ogbn-arxiv datasets, under different N-way K-shot settings (similar results can be observed on the other datasets and settings). For the GCN baselines, following the common practice [4, 51], we pretrain a 2-layer GCN using all the data from base classes with Cross-Entropy Loss. Specifically, Frozen means during fine-tuning, the GNN encoder is fixed, and only the classifier is fine-tuned. Prompt refers to the proposed virtual node tuning method. GPPE is the proposed graph-based pseudo prompt evolution module. Because GPPE is based on the proposed framework of VNT, for the tested variant with GPPE, we freeze the GT encoder and add virtual node prompts. The scores reported are averaged over 5 runs with different random seeds. The results of

Table 2: Ablation study on Cora and Ogbn-arxiv datasets to analyze the effectiveness of different components in our method.

Encoder	Frozen	VNT	GPPE	Cora		Ogbn-arxiv			
				2-way 1-shot	2-way 5-shot	2-way 1-shot	2-way 5-shot	5-way 1-shot	5-way 5-shot
GCN				52.12±2.62	57.93±2.23	57.62±2.31	64.11±2.65	26.68±1.57	27.90±1.45
GCN	✓			68.43±2.94	78.20±2.83	65.21±2.86	77.10±2.46	38.47±1.77	51.46±1.69
GT				67.50±2.24	79.42±1.89	63.00±2.35	79.84±1.98	40.73±2.65	55.35±1.88
GT	✓			75.50±2.54	84.94±1.74	53.64±2.62	73.64±2.33	31.64±2.45	52.36±2.04
GT		✓		77.85±1.99	85.43±1.84	71.82±2.58	82.73±2.14	36.36±2.74	65.45±2.31
GT	✓	✓		84.50±1.94	90.50±1.55	82.00±1.77	87.27±1.64	50.40±1.97	74.91±1.87
GT	✓	✓	✓	**88.62±2.12**	**95.65±1.51**	**85.35±1.72**	**89.98±1.66**	**61.34±1.86**	**79.93±1.69**

the ablation study are presented in Table 2, from which we draw the following conclusions:

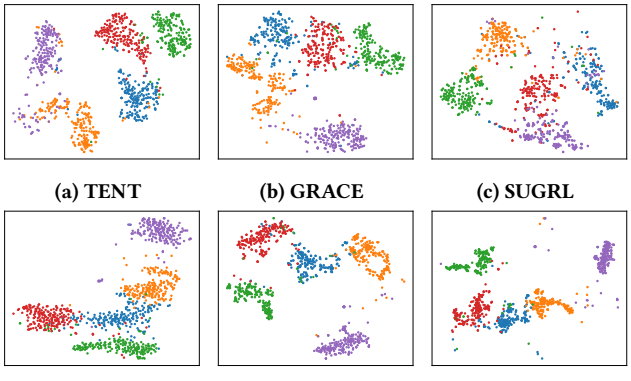
- Our simple implementation of **GT can consistently yield better results** than traditional GNNs, such as GCN. This is because the GT has a much larger number of parameters, making it capable of learning more complex relations among nodes. Besides, pretrained with the two pretext tasks, i.e., *node attribute reconstruction* and *structure recovery*, in a self-supervised manner, GT can learn more transferable graph patterns compared to those GCN-based methods.
- **Freezing the GNN encoder** during fine-tuning on the downstream FSNC tasks consistently leads to better results. This shows that fine-tuning the graph encoder on the few labeled nodes could lead to overfitting and negatively impact the quality of the learned node embeddings.
- The proposed method, VNT, which contains a frozen pretrained GT encoder with virtual node, exhibits **competitive performance** compared with vanilla GTs. This implies that the introduced virtual nodes can help the model modulate the learned substantial graph knowledge for each FSNC task while avoiding impairing the pretrained embeddings.
- The proposed method, VNT-GPPE, which involves the prompt evolution module achieves **the best performance**. This validates that the introduced GPPE module can effectively provide positive transfer from source tasks to target tasks and mitigate the overfitting on novel tasks.

4.4 Node Embedding Analysis

To explicitly illustrate the advantage of the proposed framework, in this subsection, we analyze the quality of the learned node representations from different training strategies. Particularly, we leverage two prevalent clustering evaluation metrics: *Normalized Mutual Information* (NMI) and *adjusted random index* (ARI), on learned node embeddings clustered based on K-Means. Also, we deploy t-SNE to visualize them and compare them with those learned by baseline methods on the CoraFull dataset. We choose nodes from 5 randomly selected novel classes for visualization. The results are presented in Table 3 and Fig. 2. Complete results with all the baselines are included in Appendix F. We observe that the proposed VNT-GPPE method enhances the quality of the node representations of GTs and achieves the best clustering performance on novel classes. Also, we find that a vanilla GT without VNT cannot learn node embeddings that are discriminative enough compared to strong baselines like TENT and SUGRL. However, when equipped with the proposed VNT, a GT can learn highly discriminative node embeddings. Furthermore, GPPE can significantly improve the performance of VNT. This also authenticates that the introduced prompts help to elicit more customized knowledge for each downstream FSNC task, and GPPE can effectively transfer the knowledge learned from source FSNC tasks to target FSNC tasks.

Table 3: The overall NMI (↑) and ARI (↑) scores of baselines and ablated variants of the proposed framework on CoraFull and CiteSeer datasets.

Dataset	CoraFull		CiteSeer	
Metrics	NMI	ARI	NMI	ARI
TENT	0.5760	0.4652	0.0930	0.0811
SUGRL	0.7680	0.7049	0.3952	0.4460
GT	0.5225	0.3864	0.3452	0.3189
VNT	0.7768	0.6427	0.5998	0.6331
VNT-GPPE	**0.7927**	**0.7075**	**0.6324**	**0.6762**



(a) TENT (b) GRACE (c) SUGRL

(d) GT (e) VNT (f) VNT-GPPE

Figure 2: The t-SNE results on CoraFull (5-way 5-shot).

4.5 Design Discussion

4.5.1 Interpretation of Virtual Nodes as Prompt. In this study, we explore the interpretability of virtual node prompts in graph data. Unlike previous works in NLP, where prompts are composed of human language that is easily interpretable by humans, virtual nodes in graph data are injected into the node embedding space, making it

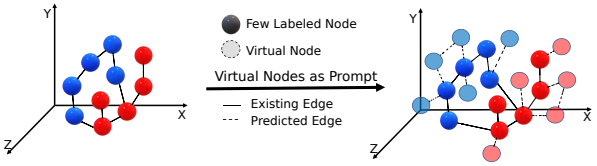


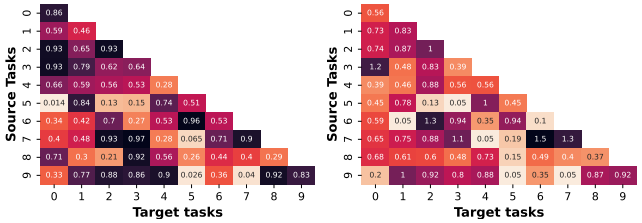
Figure 3: The illustration of effect from introduced virtual nodes under 2-way node classification setting. Different colors indicate different node classes.

Table 4: The L_2 distance and *cosine* similarity scores between prompting virtual nodes and real nodes from two novel classes on Cora dataset. P denotes virtual node prompts, and N denotes existing nodes.

Metrics	L_2	*Cosine*	L_2	*Cosine*
Nodes	P in class-1		P in class-2	
N in class-1	0.0622	0.7625	0.6725	0.2344
N in class-2	0.6520	0.8627	0.0548	0.1165

difficult to understand their effect. To gain insight into the behavior of virtual node prompts, we exploit Walkpooling [28], which is the state-of-the-art for link prediction, to predict the links that are the most likely to exist between the virtual nodes and the few labeled nodes from the support set of each task. We train the model on the whole graph dataset and use it to predict the most possible links between the virtual nodes and the existing ones. Specifically, we only consider potential links with at least one virtual node as their vertex. Under the 2-way few-shot node classification setting, we initialize half of the virtual node prompts as the prototype vector of the first class, and the other half of the virtual node as the prototype vector of the second class. As indicated in Fig. 3, after the convergence of virtual node tuning, we find that the vertices of the most possible links always connect existing nodes with virtual prompt nodes from the same classes. This implies that the introduced virtual node prompts can learn node representations semantically similar to those from the same class, and **thus help push node representations from the same classes closer**. To further validate this, on Cora dataset, we calculate the average *cosine* similarities and L_2 distances (normalized by the longest distance of any pair of nodes) for virtual nodes and existing nodes from the two novel classes. As presented in Table 4, we can see that the virtual nodes and existing nodes from the same classes have smaller L_2 distances and larger *cosine* similarities. We provide further analyses of the effect from different numbers of virtual nodes in Fig. 7 in Appendix E.

4.5.2 Motivation of Virtual Node Prompt Transfer. In this experiment, we empirically study the transferability across randomly sampled 10 source FSNC tasks and 10 target FSNC tasks from CoraFull dataset under the 5-way 5-shot setting. To test this, we first perform VNT on a source task and then directly reuse the learned virtual node prompts for other target tasks. As shown in Fig. 4 (a), we observe that reusing the prompts learned from some source tasks will provide decent performance on corresponding target tasks. Then, we examine a very naive approach to transfer prompts: we use the learned virtual node prompt from a source task as the initializer of prompts for a target task, and then fine-tune it with VNT. As

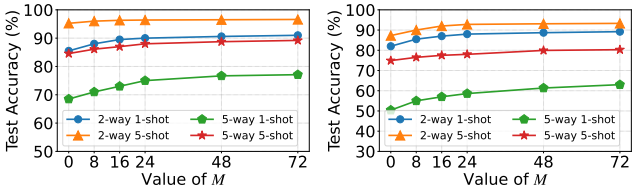


(a) Learned prompts reuse **(b) Learned prompts as initializer**

Figure 4: Relative prompt transfer performance (transfer performance / original VNT performance) on the target tasks of the virtual node prompts trained on the source tasks.

demonstrated in Fig. 4 (b), we can see that through such a simple transfer, VNT can perform better on some target tasks than training from scratch. Both these results imply that selectively transferring learned knowledge from prompts learned in source tasks to target tasks will likely yield positive transfer. This experiment motivates us in designing the GPPE module.

4.5.3 The Effect of Source Task Number M. In this experiment, we evaluate the effect of the source task number M on our framework, VNT-GPPE. A larger value of M signifies more labeled nodes in base classes. $M = 0$ means no source task exists. Thus, the framework will be reduced to VNT. Fig. 5 reports the results of our framework with varying values of M under different few-shot settings. From the results, we observe that generally increasing M will lead to better performance. This is because more labeled nodes in base classes contain more transferable graph knowledge, and the proposed GPPE module can effectively transfer the learned knowledge to target novel classes. We choose $M = 48$ as the default setting. An important observation is that, given a very small number of source FSNC tasks, e.g., $M = 8$, GPPE can still improve VNT by a large margin. This shows that the proposed framework scales well to scenarios with sparsely labeled base classes.



(a) CoraFull **(b) Ogbn-arxiv**

Figure 5: The results of our proposed framework, VNT-GPPE, under different few-shot settings with varying values of the number of source FSNC tasks M on CoraFull and Ogbn-arxiv.

4.6 Sensitivity Analysis of VNT

In this experiment, we aim to study the sensitivity of the VNT framework under various conditions. Specifically, we conduct experiments to evaluate its performance when no source tasks exist in the base classes, i.e., $M = 0$. This setup allows us to observe the general performance of the VNT framework in different scenarios. The results of these experiments will provide valuable insights into the robustness and flexibility of the VNT framework, and help guide the design of future research in this area.

4.6.1 Prompt Initialization. Noticeably, results in subsection 4.5.1 imply that initializing the virtual nodes as prototype representations could benefit the virtual node tuning process. Intuitively, given a test node, an ideal model should produce an output node embedding that is close to the corresponding class prototype representations. To test this, we initialize equal portions of virtual prompt nodes to all novel node classes as their prototype representations. The results, presented in Table 6 in Appendidx D, show that this simple initialization can consistently enhance the performance on downstream few-shot node classification tasks. This suggests that providing the model with hints about the target categories through initialization can help improve the optimization process.

4.6.2 Prompt Ensemble. Previous research by Lester et al. [22] has demonstrated the efficiency of using prompts for ensembling, as the large transformer backbone can be frozen after pretraining, reducing the storage space required and allowing for efficient inference through the use of specially-designed batches [16, 22]. Given such advantages, we investigate the effectiveness of enabling prompt ensembling for VNT. Concretely, to facilitate the ensembling with the prompt initialization strategy, we add independently sampled Gaussian noise tensors to 5 prompts for each FSNC task. Each prompt contains virtual nodes initialized as node class prototypes. We use majority voting to compute final predictions from the ensemble. Table 6 in Appendidx D shows that the ensembled model outperforms the average or even the best single prompt counterpart.

4.6.3 Effectiveness with regard to the Scale of GTs. In Fig. 6 in Appendix D, we present the accuracy of the proposed framework against the scale of the GT encoder as a heat map. We analyze the impact of varying the width (embedding size F) and the depth (number of transformer layers D) of the GT on the performance of our framework. The results shown are from the `Cora` dataset under the 2-way 1-shot setting and similar trends are observed on other datasets under different settings. We have the following findings:

i. Increasing the depth of the GT encoder does not necessarily improve the performance of downstream FSNC tasks. In fact, as the depth increases, the final accuracy may decrease. This is likely because the virtual node prompts are injected only before the first transformer layer, and their effect diminishes as they go deeper. This highlights the importance and effectiveness of the proposed VNT framework for better adaptation in FSNC tasks.

ii. On the other hand, increasing the width of the GT encoder, i.e., the embedding size, leads to improved accuracy. A larger embedding size implies that the input node embeddings contain more detailed semantic and topological information, allowing the injected virtual nodes to modulate the node embeddings more precisely.

5 RELATED WORK

5.1 Few-shot Node Classification.

Episodic meta-learning [10] has become the most dominant paradigm for Few-shot Node Classification (FSNC) tasks. It trains the GNN encoders by explicitly mimicking the test environment for few-shot learning [4, 51]. Nonetheless, these methodologies depend on the supposition that 'base classes' are available, with ample labeled nodes per class for episode sampling. This leads to a limitation, as these existing techniques do not cater to the more general FSNC problem defined in our study.

5.2 Graph Transformer.

Graph Transformers (GTs) [2, 30, 49] are a new breed of GNNs that leverage the transformer architecture [37]. GTs typically consist of two parts: an embedding network that projects raw graphs to the embedding space, and a transformer-based network that learns the complex relationships among the embeddings. Due to their vast number of parameters, GTs have the ability to capture a greater depth and complexity of knowledge. They are usually trained in a self-supervised way using pre-defined pretext tasks, such as such as node attribute reconstruction [49] and structure recovery [2] to encode both topological and semantic information. Recent advancements in GTs aim to encode increasingly complex topological knowledge, such as adaptive mechanisms for position encoding from graph spectrums [7, 20] and iterative encoding of local sub-structures as auxiliary information [26]. A recent study [29] presents a generalized recipe for developing GTs. Despite the promising results of GTs in general transfer learning tasks, no current studies have successfully applied them to the unique challenge of few-shot learning scenarios with limited labeled data.

5.3 Learning with Prompts on Graphs.

Prompting, as highlighted in [22, 23], is a recent development in Natural Language Processing (NLP) that adapts large language models to various downstream tasks by adding task descriptions to input texts. This technique has inspired several studies [8, 25, 32] to apply such prompting methods to pretrained message-passing GNNs on symbolic graph data. For instance, GPF [8] introduces learnable perturbations as prompts for graph-level tasks. GPPT [32] and Graph Prompt [25] propose a uniform template for pretext tasks and targeted downstream tasks to facilitate prompting. Contrasting with these works, our framework introduces adjustable virtual nodes in the embedding space of Graph Transformers. By incorporating a unique GPPE module, our framework can address scenarios with sparse labels in base classes. Our paper is the first to present a prompt-based method for GTs, specifically designed for few-shot node classification tasks.

6 CONCLUSION

In this paper, we propose a novel approach, dubbed as Virtual Node Tuning (VNT), to tackle the problem of general few-shot node classification (FSNC) on symbolic graphs. This method adjusts pretrained graph transformers (GTs) by incorporating virtual nodes in the embedding space for tailored node embeddings. We also design a Graph-based Pseudo Prompt Evolution (GPPE) module for efficient knowledge transfer in scenarios with sparse labels. Our comprehensive empirical studies showcase our method's effectiveness and its potential for prompt initialization and ensemble. Our research thus pioneers a novel approach for learning on graphs under limited supervision and fine-tuning GTs for a target domain.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation (NSF) under grants IIS-2229461.

REFERENCES

[1] Aleksandar Bojchevski and Stephan Günnemann. 2018. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In *ICLR*.

[2] Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. 2022. Structure-aware transformer for graph representation learning. In *International Conference on Machine Learning*. PMLR, 3469–3489.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[4] Kaize Ding, Jianling Wang, Jundong Li, Kai Shu, Chenghao Liu, and Huan Liu. 2020. Graph prototypical networks for few-shot learning on attributed networks. In *CIKM*.

[5] Kaize Ding, Zhe Xu, Hanghang Tong, and Huan Liu. 2022. Data augmentation for deep graph learning: A survey. *arXiv preprint arXiv:2202.08235* (2022).

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[7] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2021. Graph neural networks with learnable structural and positional representations. *arXiv preprint arXiv:2110.07875* (2021).

[8] Taoran Fang, Yunchao Zhang, Yang Yang, and Chunping Wang. 2022. Prompt Tuning for Graph Neural Networks. *arXiv preprint arXiv:2209.15240* (2022).

[9] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

[11] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*. 1024–1034.

[12] Kaveh Hassani and Amir Hosein Khasahmadi. 2020. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*. PMLR, 4116–4126.

[13] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *NeurIPS*.

[14] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*. 2704–2710.

[15] Kexin Huang and Marinka Zitnik. 2020. Graph meta learning via local subgraphs. In *NeurIPS*.

[16] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. 2022. Visual prompt tuning. *arXiv preprint arXiv:2203.12119* (2022).

[17] Yizhu Jiao, Yun Xiong, Jiawei Zhang, Yao Zhang, Tianqi Zhang, and Yangyong Zhu. 2020. Sub-graph contrast for scalable self-supervised graph representation learning. In *2020 IEEE international conference on data mining (ICDM)*. IEEE, 222–231.

[18] Ming Jin, Yizhen Zheng, Yuan-Fang Li, Chen Gong, Chuan Zhou, and Shirui Pan. 2021. Multi-scale contrastive siamese networks for self-supervised graph representation learning. In *International Joint Conference on Artificial Intelligence 2021*. Association for the Advancement of Artificial Intelligence (AAAI), 1477–1483.

[19] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[20] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. 2021. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems* 34 (2021), 21618–21629.

[21] Lin Lan, Pinghui Wang, Xuefeng Du, Kaikai Song, Jing Tao, and Xiaohong Guan. 2020. Node classification on graphs with few-shot novel labels via meta transformed network embedding. *Advances in Neural Information Processing Systems* 33 (2020), 16520–16531.

[22] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691* (2021).

[23] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586* (2021).

[24] Zemin Liu, Yuan Fang, Chenghao Liu, and Steven CH Hoi. 2021. Relative and absolute location embedding for few-shot node classification on graph. In *AAAI*.

[25] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. 2023. GraphPrompt: Unifying Pre-Training and Downstream Tasks for Graph Neural Networks. In *Proceedings of the ACM Web Conference 2023*. 417–428.

[26] Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. 2021. Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667* (2021).

[27] Yujie Mo, Liang Peng, Jie Xu, Xiaoshuang Shi, and Xiaofeng Zhu. 2022. Simple unsupervised graph representation learning. AAAI.

[28] Liming Pan, Cheng Shi, and Ivan Dokmanić. 2021. Neural Link Prediction with Walk Pooling. In *International Conference on Learning Representations*.

[29] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. 2022. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems* 35 (2022), 14501–14515.

[30] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. 2020. Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems* 33 (2020), 12559–12571.

[31] Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *NeurIPS*.

[32] Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. 2022. Gppt: Graph pre-training and prompt tuning to generalize graph neural networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1717–1727.

[33] Zhen Tan, Kaize Ding, Ruocheng Guo, and Huan Liu. 2022. Graph few-shot class-incremental learning. In *WSDM*.

[34] Zhen Tan, Kaize Ding, Ruocheng Guo, and Huan Liu. 2022. A Simple Yet Effective Pretraining Strategy for Graph Few-shot Learning. *arXiv preprint arXiv:2203.15936* (2022).

[35] Zhen Tan, Song Wang, Kaize Ding, Jundong Li, and Huan Liu. 2022. Transductive Linear Probing: A Novel Framework for Few-Shot Node Classification. *arXiv preprint arXiv:2212.05606* (2022).

[36] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Remi Munos, Petar Veličković, and Michal Valko. 2021. Bootstrapped Representation Learning on Graphs. In *ICLR Workshop on Geometrical and Topological Representation Learning*.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[38] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[39] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. 2020. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies* (2020).

[40] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint arXiv:1909.01315* (2019).

[41] Ning Wang, Minnan Luo, Kaize Ding, Lingling Zhang, Jundong Li, and Qinghua Zheng. 2020. Graph Few-Shot Learning with Attribute Matching. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*.

[42] Song Wang, Kaize Ding, Chuxu Zhang, Chen Chen, and Jundong Li. 2022. Task-Adaptive Few-shot Node Classification. *arXiv preprint arXiv:2206.11972* (2022).

[43] Zhihao Wen, Yuan Fang, and Zemin Liu. 2021. Meta-inductive node classification across graphs. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[44] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771* (2019).

[45] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks?. In *Proceedings of the 2019 International Conference on Learning Representations*.

[46] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*. PMLR, 40–48.

[47] Huaxiu Yao, Chuxu Zhang, Ying Wei, Meng Jiang, Suhang Wang, Junzhou Huang, Nitesh Chawla, and Zhenhui Li. 2020. Graph few-shot learning via knowledge transfer. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*.

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

[49] Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. 2020. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140* (2020).

[50] Shengzhong Zhang, Ziang Zhou, Zengfeng Huang, and Zhongyu Wei. 2018. Few-shot Classification on Graphs with Structural Regularized GCNs. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

[51] Fan Zhou, Chengtai Cao, Kunpeng Zhang, Goce Trajcevski, Ting Zhong, and Ji Geng. 2019. Meta-gnn: On few-shot node classification in graph meta-learning. In *CIKM*.

[52] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131* (2020).

A IMPLEMENTATION DETAIL

A.1 General Settings

All experiments are implemented using PyTorch. We run all experiments on a single 80GB Nvidia A100 GPU.

A.2 Implementation of the Simplified GT

For generality, we try to keep the used GT encoder very simple and easy to transfer to other complicated architectures. Specifically, we use a 1-layer MLP to project the raw graph, including node attributes and structural positions into the embedding space. We use simple summation to merge those embeddings together and feed them into the following transformer layers. We use the transformer module released by huggingface [44]. We perform a grid search like Section 4.6.3 to get the width and depth of the GT.

For pretraining, we utilize two prevailing pretext tasks, *node attribute reconstruction* and *structure recovery*, to train the GT encoder in a self-supervised manner. Concretely, for *node attribute reconstruction* pretext, given a node, we minimize the *Mean Square Error* (MSE) between the original node attributes and the reconstructed version via a fully connected layer and the learned node embedding from the GT. For *structure recovery* pretext, given any pair of nodes, we try to predict if there is a link between them and compare the result with the ground truth by an MSE loss. To accommodate a larger graph, similar to Jiao et al. [17], Mo et al. [27], we adopt the mini-batch strategy to sample a portion of nodes with their subgraphs (based on PPR) in each epoch for pretraining.

B DETAILS OF BENCHMARK DATASETS

Table 5: Statistics of benchmark node classification datasets. $\mathbb{C}_{train}$ denotes the base classes for training, $\mathbb{C}_{dev}$ and $\mathbb{C}_{test}$ denote novel classes for validation and test respectively.

Dataset	# Nodes	# Edges	# Features	$\lvert\mathbb{C}\rvert$	$\lvert\mathbb{C}_{train}\rvert$	$\lvert\mathbb{C}_{dev}\rvert$	$\lvert\mathbb{C}_{test}\rvert$
CoraFull	19,793	63,421	8,710	70	40	15	15
Ogbn-arxiv	169,343	1,166,243	128	40	20	10	10
Cora	2,708	5,278	1,433	7	3	2	2
CiteSeer	3,327	4,552	3,703	6	2	2	2

In this section, we provide detailed descriptions of the benchmark datasets used in our experiments. All the datasets are public and available on both PyTorch-Geometric [9] and DGL [40].

- **CoraFull** [1] is a citation network that extends the prevalent small Cora network. Specifically, it is achieved from the entire citation network, where nodes are papers, and edges denote the citation relations. The classes of nodes are obtained based on the paper topic.
- **Ogbn-arxiv** [13] is a directed citation network that consists of CS papers from MAG [39]. Here nodes represent CS arXiv papers, and edges denote the citation relations. The classes of nodes are assigned based on the 40 subject areas of CS papers in arXiv.
- **Cora** [46] is a citation network dataset where nodes mean paper and edges mean citation relationships. Each node has a predefined feature with 1433 dimensions. The dataset is designed for the node classification task. The task is to predict the category of a certain paper.
- **CiteSeer** [46] is also a citation network dataset where nodes mean scientific publications and edges mean citation relationships. Each node has a predefined feature with 3703 dimensions. The dataset is designed for the node classification task. The task is to predict the category of a certain publication.

C A MORE DETAILED REVIEW FOR FEW-SHOT NODE CLASSIFICATION

The task of few-shot node classification (FSNC) aims to train models that can assign labels to unlabeled nodes in graphs, using only a few labeled nodes per class for training. Recently, episodic meta-learning [10] has become a popular paradigm for addressing label scarcity in FSNC tasks. This approach trains GNN encoders by emulating the test environment for few-shot learning. For example, Meta-GNN [51] uses MAML [10] to learn optimization directions with limited labels. GPN [4] employs Prototypical Networks [31] to perform classification based on the distance between node features and prototypes. MetaTNE [21] and RALE [24] also use episodic meta-learning to improve the adaptability of learned GNN encoders and achieve similar results. Additionally, G-Meta [15], GFL-KT [47], and MI-GNN [43] use meta-learning to transfer knowledge when other auxiliary graphs are available. TNT [42] further takes into account the variance among different meta-tasks.

D EXPERIMENT RESULTS OF DESIGN DISCUSSION

D.1 Prompt Initilization and Ensemble

Table 6: The accuracy scores of VNT on Cora and Ogbn-arxiv datasets. Init. indicates the prototype-based prompt initialization strategy described in Section 4.6.1. Method without Init. means the prompts are randomly initialized. The best results are bold. MV refers to majority voting.

VNT			Cora		Ogbn-arxiv			
Init.	Ensemble		2-way 1-shot	2-way 5-shot	2-way 1-shot	2-way 5-shot	5-way 1-shot	5-way 5-shot
			84.50	90.50	82.00	87.27	50.40	74.91
✓			85.25	91.30	83.05	88.34	51.06	75.86
✓	✓	Avg.	85.50	89.64	82.50	89.65	48.82	75.65
		Best	86.24	92.38	**85.45**	90.63	**53.68**	78.82
		MV	**87.63**	**92.52**	84.72	**91.50**	52.15	**79.60**

D.2 Effectiveness with regard to the Scale of GTs

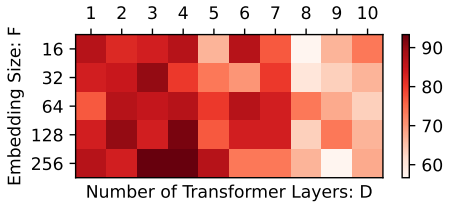


Figure 6: The 2-way 1-shot accuracy (%) of the proposed VNT according to the scale of the GT encode on the Cora dataset.

Table 7: The overall NMI (↑) and ARI (↑) scores of baselines and ablated variants of the proposed framework on `CoraFull` and `CiteSeer` datasets. The best results among the variants and baselines are bold and underlined, respectively.

Dataset	CoraFull		CiteSeer	
Metrics	**NMI**	**ARI**	**NMI**	**ARI**
Meta-learning				
MAML	0.1622	0.0597	0.0754	0.0602
ProtoNet	0.2669	0.1263	0.0915	0.0765
AMM-GNN	0.6247	0.5087	0.2090	0.1781
G-Meta	0.5003	0.3702	0.1913	0.1502
Meta-GNN	0.5534	0.4196	0.1317	0.1171
GPN	0.6001	0.4599	0.2119	0.2087
TENT	0.5760	0.4652	0.0930	0.0811
GCL-based TLP				
MVGRL	0.6227	0.4788	0.2554	0.2232
GraphCL	0.7023	0.5628	0.5579	0.5890
GRACE	0.6781	0.5856	0.2663	0.2778
BGRL	0.5137	0.4382	0.2051	0.1875
MERIT	0.7419	0.6590	0.3923	0.4014
SUGRL	0.7680	0.7049	0.3952	0.4460
Ablated variants of VNT				
GT	0.5225	0.3864	0.3452	0.3189
VNT	0.7768	0.6427	0.5998	0.6331
VNT-GPPE	**0.7927**	**0.7075**	**0.6324**	**0.6762**

E NUMBER OF VIRTUAL NODES

The number of virtual nodes P is an important hyper-parameter to tune. On the four benchmark datasets, we give the results under the 2-way 5-shot setting. Specifically, we use a parameter α to control the number of virtual nodes. For a N-way K-shot FSNC task, we define $\alpha = \frac{P}{N \cdot K}$. Larger α means more virtual nodes are introduced. From the results shown in Fig. 7, we find that when $\alpha = 1$, the proposed VNT can give the best performance. Therefore, we choose $P = \alpha \times (N \cdot K) = N \cdot K$ as the default number of virtual nodes per prompt.

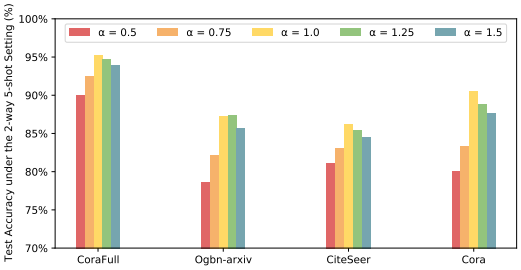


Figure 7: The test accuracy (%) under different α values on four benchmark datasets.

F MORE RESULTS ON NODE CLUSTERING

In table 7, we present the complete results on Node Clustering in terms of NMI and ARI.