Diversity-Aware Forward Prototype Learning of Graph Neural Networks for Node Classification

Enze Zhang
School of Software
South China University of Technology
Guangzhou, China
sefranciszhang@mail.scut.edu.cn

Qian Tao*
School of Software
South China University of Technology
Guangzhou, China
taoqian@scut.edu.cn

***Abstract*—Graph neural networks (GNNs) have achieved great success in many applications. Most existing GNNs rely on backpropagation for end-to-end training. Recently, forward-only learning methods have been considered as promising alternatives, but directly applying them to GNNs faces challenges in efficiency and training stability. In this paper, we propose the Prototype-based Forward algorithm for Graph Neural Networks (ProF-GNN), a forward-style GNN learning framework that introduces class-aware virtual nodes to explicitly model class prototypes. By aligning node representations with their corresponding prototypes at each layer, ProF-GNN enables direct and efficient prediction without exhaustive label-wise inference or the generation of multiple negative samples. Furthermore, we incorporate a diversity-aware objective to prevent prototype collapse, ensuring stable and discriminative layer-wise learning. Extensive experiments on real-world datasets demonstrate the effectiveness of the proposed learning framework for GNNs for node classification.**

***Index Terms*—graph neural networks, node classification, prototype learning, forward learning**

I. Introduction

Graph neural networks (GNNs) are a class of neural networks with advantages in learning useful information from graph-structured data. The success of GNNs can be attributed to backpropagation (BP) [1], which provides an efficient and unified mechanism for learning from graph-structured data. Through BP, GNNs are able to optimize their parameters end-to-end by propagating error signals back through multiple layers of message passing and nonlinear transformations. This allows each layer to adapt how node features are aggregated and transformed based on downstream supervision, enabling the model to learn task-relevant representations. Moreover, BP makes it possible to jointly train all components of a GNN under a single loss function, ensuring coherent representation learning across local neighborhoods and global graph structure.

Although BP plays a central role in turning GNNs into powerful models for different graph learning tasks, it also suffers from several well-known limitations that motivate the search for alternative learning paradigms. The major limitations of BP can be summarized as locality, freezing, update locking, and weight transport. To address these issues, Geoffrey Hinton proposed the forward-forward (FF) algorithm [2], which replaces the backward pass with local learning objectives computed during purely forward computations. By training each layer independently using local goodness measures, the FF algorithm eliminates the need for global error propagation and offers a more biologically plausible alternative to BP.

Inspired by the idea of FF, different variants of FF have been developed over the last few years. Some of these variants focused on relaxing the boundary of positive and negative data by using margin-based [3], contrastive [4], or probabilistic objectives [5], improving the stability and sample efficiency. Other variants explored architectural adaptations, such as convolutional FF for images [6], recurrent FF for temporal data [7], and hybrid schemes that combine FF-style local objectives with limited gradient-based optimization [8]. In the field of graph representation learning, the Graph Forward-Forward (GFF) algorithm [9] and ForwardGNN [10] are two representative approaches designed for graph downstream tasks. These approaches demonstrate how the FF algorithm can be systematically adapted to graph-structured data, highlighting a promising direction for training GNNs in a forward-only way.

Directly applying forward-only techniques to GNNs faces two major challenges. The first challenge concerns efficiency in both training and inference. Existing methods with two forward passes typically train different layers sequentially using both positive and negative data, which doubles (or even multiplies) the time and memory consumption compared to BP. This layer-wise, contrastive training paradigm limits scalability to large graphs. Furthermore, in the inference stage, most FF-based methods require encoding the input multiple times—once for each candidate label—and selecting the label that yields the highest goodness score. This exhaustive label-wise evaluation further increases the inference time. Addressing these efficiency bottlenecks is therefore crucial for applying forward-only learning frameworks for GNNs.

The second challenge is the instability of goodness-based learning and inference. In many FF-based methods, negative training data are constructed using only a single incorrect label, which often provides an insufficient contrast signal and leads to unstable or noisy optimization, especially for multi-classification tasks. Although some existing works attempt to

*Corresponding author.

improve stability by generating multiple negative data per instance, this strategy significantly increases memory consumption and computational overhead. As a result, designing mechanisms that allow each layer to make more direct predictions, without relying on multiple negative data, remains a challenging problem.

To address these challenges, in this paper, we propose the Prototype-based Forward algorithm for Graph Neural Networks (ProF-GNN), a novel forward-style learning framework for GNNs for node classification. For the first challenge, ProF-GNN requires only a single augmented graph with class-aware virtual nodes for training, eliminating the need for generating multiple negative samples or performing exhaustive label-wise inference. Each virtual node serves as a class prototype, and the learning objective encourages node representations at each layer to align with their corresponding prototypes, allowing GNNs to make more direct predictions. For the second challenge, we introduce a diversity-aware objective that explicitly enforces separation among class prototypes, preventing prototype collapsing and ensuring more discriminative representations during layer-wise training. By combining the prototype alignment with diversity regularization, ProF-GNN achieves both efficient and reliable forward-style learning framework for GNNs.

The main contributions of this paper are summarized as follows:

- We propose ProF-GNN, a prototype-based GNN learning framework that augments graphs with class-aware virtual nodes to explicitly model class-level representations, enabling direct and efficient predictions without exhaustive label-wise evaluation.
- Building upon existing layer-wise learning methods, we further advance prototype-based local training by incorporating a diversity-aware objective, which preserves inter-class discriminability by preventing different prototypes from becoming overly similar.
- We conduct extensive experiments on three benchmark node classification datasets and three representative GNN models to demonstrate the effectiveness of the proposed ProF-GNN and its potential as an alternative to BP for training GNNs for node classification.

II. Related Works

Forward learning is a category of neural network learning technique that reduces or eliminates the dependence on BP. Hinton [2] proposed the Forward-Forward (FF) algorithm to overcome the biological implausibility of BP. FF highlights its biological plausibility by replacing the forward and backward passes in BP with two forward passes using different data. The optimization of the parameters of each layer is guided by goodness, a layer-wise metric affected only by the output of the current layer. The objective is to make the positive (negative) data produce a larger (smaller) goodness on each layer. Compared to BP, FF behaves more similarly to the learning process of the human brain. Since the FF algorithm was proposed, many researchers have tried to introduce the framework of FF to all kinds of neural networks, such as convolutional neural networks [6, 11], recurrent neural networks [7], physical neural networks [12], and graph neural networks [9, 10].

However, most methods require generating positive and negative data with specific methods. For the image data, the augmented data can be generated by simply overlaying some pixels with the encoded labels [2] or fusing the class patterns into the pixels [3, 6]. The negative data are usually created by combining the wrong labels into the original data. Performing two forward passes with different data produces more computational cost, making it hard to reach the goal of reducing the cost of BP. Furthermore, in the inference stage, these methods require one to try encoding every label to the input and compare the accumulative goodnesses across all labels, making it more time-consuming to make the final predictions.

To simplify the process of data augmentation and prediction, Zhao et al. [5] proposed Cascaded Forward (CaFo) algorithm which equips a single-layer predictor on each layer. To avoid performing BP, CaFo only trains the parameters on the single-layer predictors, and the parameters of the layers are frozen or updated with direct feedback alignment. Since CaFo directly outputs the prediction result, it does not require any augmentation on the input data and label-wise evaluation.

In the field of graph representation learning, Paliotta et al. [9] introduced the Graph Forward-Forward (GFF) algorithm which first adapts the FF algorithm for graph classification task. However, GFF still requires the generation of positive and negative data for training, increasing the computational cost of training. Park et al. [10] proposed ForwardGNN which implicitly adopts the idea of prototype learning and performs a single forward pass to train the GNN model. ForwardGNN simplifies the process of data augmentation and training. However, this single-pass approach neglects the critical situation that the similarity of the features of the virtual nodes corresponding to different classes is extremely high, leading to confusion in the inference stage. For example, suppose that a real node with class A is similar to A's virtual node after training. When the representation of A's virtual node is close to that of another class B, the real node may be easy to be mistakenly predicted as class B.

Our method not only aligns the node representations with their corresponding virtual nodes, but also considers the similarity of class prototypes and avoids confusion caused by similar prototypes during prediction. As a result, the representation of each node is less similar to the prototype of other classes and the prediction performance is effectively increased.

III. Preliminaries

We consider a simple undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes, where $\mathcal{V} = \{v_1, v_2, ..., v_N\}$ denotes the set of nodes and $\mathcal{E}$ denotes the set of edges. An edge $(v_i, v_j) \in \mathcal{E}$ indicates that nodes v_i and v_j are connected. Each node is associated with a feature vector, and the node feature matrix is denoted by $\mathbf{X} \in \mathbb{R}^{N \times d}$, where d is the feature dimension. The structure of

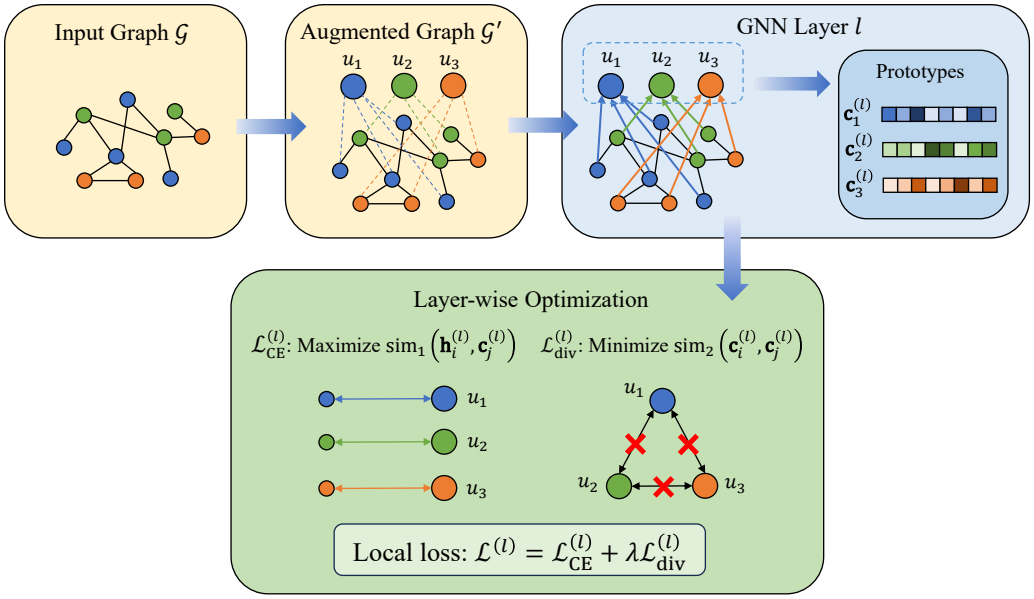


Fig. 1. Overview of the proposed ProF-GNN framework.

$\mathcal{G}$ is represented by the adjacency matrix $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{N \times N}$, where $a_{ij} = a_{ji} = 1$ if $(v_i, v_j) \in \mathcal{E}$, and $a_{ij} = a_{ji} = 0$ otherwise.

The goal of supervised node classification is to learn a function $f(\mathbf{A}, \mathbf{X}) = \mathbf{y}$, where $\mathbf{y} \in \mathbb{R}^N$ denotes the ground-truth labels. For a graph with nodes with C classes, the i-th component $y_i \in \{1, 2, ..., C\}$ of $\mathbf{y}$ is the ground-truth label of node $v_i \in \mathcal{V}$. Let $\mathcal{V}_{\text{train}} \subset \mathcal{V}$ denote the training set in which the nodes are labeled. The objective of training is to minimize the supervised loss over $\mathcal{V}_{\text{train}}$ and the remaining nodes are used for evaluation.

IV. Methodology

The overall framework of the proposed ProF-GNN is illustrated in Figure 1. Before training, ProF-GNN augments the input graph by adding a set of virtual nodes, each of which is connected to real nodes belonging to that class. The features of these virtual nodes are treated as class prototypes, and the labeled nodes are encouraged to become closer to their corresponding prototypes in the representation space during training. Meanwhile, we design a diversity loss that prevents the class prototypes from being too close to each other. For each layer, we linearly combine the two losses and update the parameters locally, while keeping the parameters of the previous layers fixed.

A. Graph Augmentation

Usually, models trained by non-BP algorithms do not compare the final output with the ground-truth labels, so the input data are required to be augmented by encoding the label information as a part of the input. In the previous works focusing on BP-free GNNs [9, 10], two popular methods are applied to augment the graph data. The first method is to concatenate the encoded label to the node feature, and the second method is to add a virtual node corresponding to each possible class label. ProF-GNN adopts the second method to augment the graph data, that is, adding the virtual nodes $u_1, u_2, ..., u_C$ to the input graph and connecting each node $v_i \in \mathcal{V}_{\text{train}}$ with label y_i to the virtual node u_{y_i}. Our main idea is to regard the features of the virtual nodes as the prototypes to be learned and make predictions by comparing the node feature with these prototypes. Given a simple undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we denote the augmented graph by $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$, where

$$\mathcal{V}' = \mathcal{V} \cup \{u_i | i \in \{1, 2, ..., C\}\}, \tag{1}$$

$$\mathcal{E}' = \mathcal{E} \cup \{(v_i, u_c) | v_i \in \mathcal{V}_{\text{train}}, y_i = c\}. \tag{2}$$

To formulate the adjacency matrix $\mathbf{A}'$ of $\mathcal{G}'$, we define a label matrix $\mathbf{M} = [m_{ij}] \in \mathbb{R}^{N \times C}$, where

$$m_{ij} = \begin{cases} 1, y_i = j \text{ and } v_i \in \mathcal{V}_{\text{train}} \\ 0, \text{otherwise} \end{cases}. \tag{3}$$

Then, the adjacency matrix of $\mathcal{G}'$ is:

$$\mathbf{A}' = \begin{bmatrix} \mathbf{A} & \mathbf{M} \\ \mathbf{M}^\top & \mathbf{O} \end{bmatrix}. \tag{4}$$

For each newly added virtual node u_i, the features are randomly initialized as $\mathbf{c}_i^{(0)} \in \mathbb{R}^d$. Then, the initial feature matrix of $\mathcal{G}'$ is $\mathbf{X}' = [\mathbf{X}; \mathbf{c}_1^{(0)}; \mathbf{c}_2^{(0)}; ...; \mathbf{c}_C^{(0)}]$.

B. Prototype-based Graph Learning

For an L-layer GNN model, the output of the l-th layer $(1 \le l \le L)$ can be formulated as:

$$\mathbf{H}^{(l)} = \alpha(F^{(l)}(\mathbf{A}', \mathbf{H}^{(l-1)})), \tag{5}$$

where α is a non-linear activation function (e.g., rectified linear unit and sigmoid), and $F^{(l)}$ can be realized as any type of GNN layer (e.g., graph convolutional layer or graph attention layer). $\mathbf{H}^{(0)} = \mathbf{X}'$ and $\mathbf{A}'$ are the initial feature matrix and the adjacency matrix (defined in (4)) of the augmented graph $\mathcal{G}'$.

The main goal of our method is to make the representation of each node v_i more similar to the prototype of its label y_i. Since the virtual nodes only receive messages from nodes with their own labels, we treat the representations of these nodes as the class prototypes. The probability distributions of all nodes at the l-th layer are denoted as $\mathbf{P}^{(l)} = [p_{ij}^{(l)}] \in \mathbb{R}^{N \times C}$, where $p_{ij}^{(l)}$ denotes the probability that node v_i belongs to class j:

$$p_{ij}^{(l)} = \frac{\exp(\mathrm{sim}_1(\mathbf{h}_i^{(l)}, \mathbf{c}_j^{(l)})/\tau)}{\sum_{k=1}^{C} \exp(\mathrm{sim}_1(\mathbf{h}_i^{(l)}, \mathbf{c}_k^{(l)})/\tau)}, \tag{6}$$

where $\mathrm{sim}_1(\cdot,\cdot)$ denotes the similarity metric (e.g., dot product or cosine similarity) and τ is the temperature parameter.

After obtaining the probability distribution of each node at the current layer, we apply the mean cross-entropy loss to encourage the representations of each node in the training set to align with the prototype of its corresponding class, thereby making correct predictions:

$$\mathcal{L}_{\text{CE}}^{(l)} = -\frac{1}{|\mathcal{V}_{\text{train}}|} \sum_{v_i \in \mathcal{V}_{\text{train}}} \log p_{i,y_i}^{(l)}. \tag{7}$$

Inspired by [13], we further address the potential issue of prototype collapse, where prototypes of different classes become overly similar. To encourage better separation among prototypes, we design a diversity loss that explicitly pushes the C class prototypes away from each other. The diversity loss is defined as:

$$\mathcal{L}_{\text{div}}^{(l)} = \sum_{i=1}^{C} \sum_{j=i+1}^{C} \mathrm{sim}_2(\mathbf{c}_i^{(l)}, \mathbf{c}_j^{(l)}), \tag{8}$$

where $\mathrm{sim}_2(\cdot,\cdot)$ denotes the similarity metric, which may be different from $\mathrm{sim}_1(\cdot,\cdot)$ in (6).

Finally, for the l-th layer, the total loss we aim to minimize is:

$$\mathcal{L}^{(l)} = \mathcal{L}_{\text{CE}}^{(l)} + \lambda \mathcal{L}_{\text{div}}^{(l)}, \tag{9}$$

where λ is a hyper-parameter that controls the weight of the diversity loss.

By adopting a specific optimizer (e.g., Adam or stochastic gradient descent), the parameters of $F^{(l)}$ are updated solely according to the gradient of the local objective of $\mathcal{L}^{(l)}$. In contrast to BP, these gradients are strictly confined within the current layer and are not propagated to any previous layers. Therefore, the parameters of previous layers are completely unaffected by $\mathcal{L}^{(l)}$. This strictly local update mechanism ensures that ProF-GNN follows a forward-only, layer-wise learning paradigm, thereby preserving the biological plausibility that characterizes forward-style algorithms.

In the inference stage, the prediction result of each node v_i is obtained by picking the class with the highest mean probability across all layers:

$$\hat{y}_i = \arg\max_c \frac{1}{L} \sum_{l=1}^{L} p_{ic}^{(l)}. \tag{10}$$

The entire algorithm of ProF-GNN is summarized in Algorithm 1.

Algorithm 1: The algorithm of ProF-GNN

Input: GNN model with layers $\{F^{(1)},F^{(2)},...,F^{(L)}\}$, graph $\mathcal{G}$, training set $\mathcal{V}_{\text{train}}$, node feature $\mathbf{X}$, adjacency matrix $\mathbf{A}$, node label $\mathbf{y}$.
Output: Trained GNN model

```
G' ← augmented graph of G using Eqs.(1) and (2)
A' ← augmented adjacency matrix using Eqs.(3) and (4)
Randomly initialize the representations c_1^(0), c_2^(0), ..., c_C^(0) of the virtual nodes.
X' ← [X; c_1^(0); c_2^(0); ...; c_C^(0)]
Initialize H^(0) as X'
for l ← 1 to L do
    for epoch ← 1 to num_epochs do
        Calculate H^(l) using Eq.(5)
        Calculate the probability distribution P^(l) using Eq.(6)
        Calculate the consistency loss L_CE^(l) using Eq.(7)
        Calculate the diversity loss L_div^(l) using Eq.(8)
        Calculate the total loss L^(l) using Eq.(9)
        Update the parameters of F^(l) with an optimizer
    end
    Detach the gradients of H^(l)
end
```

V. Experiments

We conduct experiments to demonstrate the effectiveness and generality of the proposed ProF-GNN. We evaluate our method on three real-world datasets for node classification in comparison with BP and several existing forward-style training methods. All experiments are implemented in PyTorch 2.4.0 and PyTorch Geometric 2.6.0 on an NVIDIA 3090 GPU.

A. Datasets and Experimental Settings

We evaluate the proposed ProF-GNN on three widely used benchmark citation network datasets: Cora, CiteSeer, and PubMed. In these datasets, nodes represent scientific publications, edges denote citation relationships, and each node is associated with a bag-of-words feature vector. Table I presents the statistics of the datasets we used in our experiments. We split the data randomly into train, validation, and test sets, with a ratio of 60%, 20%, and 20%, respectively.

TABLE I
DATASETS STATISTICS IN EXPERIMENTS.

Dataset	Nodes	Edges	Features	Classes
Cora	2,708	10,556	1,433	7
CiteSeer	4,230	10,674	602	6
PubMed	19,717	88,648	500	3

We apply the proposed ProF-GNN on three representative GNNs, including graph convolutional network (GCN) [14], graph attention network (GAT) [15], and GraphSAGE [16]. For all types of GNNs, we set the size of hidden units to 128 and apply a ReLU activation function on each layer. For GAT, we set the number of attention heads to 4 and concatenate their outputs on each layer. For GraphSAGE, we adopt the mean aggregator on each layer.

We compare ProF-GNN with the following training methods: backpropagation (BP) [1], forward-forward (FF) [2], SymBa [3], single-forward (SF) proposed by ForwardGNN [10], cascaded forward [5] with mean squared error loss (CaFo-MSE) and cross-entropy loss (CaFo-CE). For FF, we set the threshold parameter of goodness to 2.0, and for SymBa, we set the scale factor to 4.0. For the two variants of CaFo, we adopt the non-learnable strategy to the GNN layers, that is, initializing the parameters of the GNN layers with Kaiming uniform and freezing them while training. For CaFo-MSE, we directly use the analytic solution that minimizes the mean squared error as the updated weight of each predictor. For our method, we calculate the probability distribution of each node based on dot product similarity and set the value of the temperature parameter τ in (6) to 1.0. Besides, we adopt the orthogonal projection loss as the diversity loss (i.e. $\mathrm{sim}_2(\cdot,\cdot)$ in (8) is the squared dot product of two vectors) and set the value of the trade-off parameter λ in (9) to 0.05. For our method and other baseline methods (except CaFo-MSE), the number of epochs is 500, and the Adam optimizer [17] with a learning rate of 0.001 is applied.

B. Main Results

Table II presents the average node classification accuracy of GCN, GAT, and GraphSAGE trained by BP, some forward-style baseline methods, and the proposed ProF-GNN. All the reported results are averaged over 10 runs, and the top two results among forward-style methods are highlighted in bold

TABLE II
NODE CLASSIFICATION ACCURACY (%) ON THREE DATASETS OF GCN, GAT, AND GRAPHSAGE, OBTAINED WITH A DIFFERENT NUMBER OF LAYERS, AND TRAINED BY DIFFERENT LEARNING METHODS.

(a) GCN

Methods	Cora			CiteSeer			PubMed		
	2 layers	3 layers	4 layers	2 layers	3 layers	4 layers	2 layers	3 layers	4 layers
BP	87.22±1.18	84.99±0.69	84.51±1.19	72.03±1.69	69.93±1.29	70.81±1.22	86.78±0.64	86.85±0.57	85.97±0.50
FF	79.01±2.28	77.26±2.00	75.54±2.90	71.70±1.11	71.19±2.10	69.4±1.52	84.01±0.45	83.16±0.79	82.06±0.63
SymBa	86.98±0.41	86.48±0.83	86.76±1.61	74.49±1.04	74.49±1.61	75.23±0.98	84.50±0.74	84.29±0.48	84.18±0.74
SF	86.00±1.52	85.16±1.52	85.86±1.82	71.86±1.99	71.38±0.91	70.98±2.47	**87.70±0.41**	**87.60±0.47**	**87.01±0.40**
CaFo-MSE	76.06±1.42	77.74±2.06	78.95±1.73	62.81±1.44	64.11±2.29	63.30±1.90	79.55±0.98	80.01±1.22	80.32±0.98
CaFo-CE	45.97±4.31	43.11±4.23	41.27±3.47	59.37±1.58	56.5±2.33	57.04±2.72	61.16±2.08	59.86±1.51	60.05±1.75
ProF-GNN (ours)	**87.66±1.09**	**87.15±1.08**	**88.38±1.06**	**74.89±1.65**	**75.03±1.60**	**75.36±1.74**	82.93±0.47	82.99±0.57	82.68±0.26

(b) GAT

Methods	Cora			CiteSeer			PubMed		
	2 layers	3 layers	4 layers	2 layers	3 layers	4 layers	2 layers	3 layers	4 layers
BP	83.31±1.06	83.65±1.55	84.53±1.54	70.65±2.21	71.44±1.88	71.89±1.71	84.20±0.38	86.07±0.43	85.76±0.52
FF	31.71±6.99	29.08±6.39	24.16±7.47	23.62±4.35	20.84±3.46	20.95±3.19	84.41±1.07	83.95±1.12	84.24±0.68
SymBa	70.66±4.35	72.67±1.60	74.84±2.61	60.30±2.88	67.75±1.62	71.91±2.51	82.91±1.11	84.21±0.84	83.79±1.36
SF	84.75±1.61	85.36±1.03	84.75±1.15	70.95±1.25	69.93±1.96	69.69±1.81	**87.31±0.56**	**87.05±0.48**	**86.95±0.59**
CaFo-MSE	76.61±2.37	79.36±1.40	79.94±2.01	63.47±1.32	62.48±1.65	64.61±2.29	79.05±0.89	79.88±0.76	80.08±0.60
CaFo-CE	46.61±3.58	44.47±4.10	41.33±3.42	58.26±1.88	57.09±1.94	57.06±2.38	61.19±1.19	60.55±1.89	61.11±1.58
ProF-GNN (ours)	**86.33±0.78**	**87.04±1.63**	**86.89±1.27**	**74.07±2.07**	**73.75±1.85**	**74.28±1.99**	86.68±0.71	86.66±0.48	86.07±0.37

(c) GraphSAGE

Methods	Cora			CiteSeer			PubMed		
	2 layers	3 layers	4 layers	2 layers	3 layers	4 layers	2 layers	3 layers	4 layers
BP	87.90±0.89	86.94±0.85	86.50±1.85	76.01±1.25	74.41±2.33	74.14±1.72	87.45±0.57	88.49±0.61	87.73±0.48
FF	78.73±1.28	75.55±2.00	74.44±3.25	70.81±1.86	71.07±1.16	68.98±1.52	84.89±0.61	84.08±0.74	83.94±0.83
SymBa	**88.27±1.69**	**87.00±1.12**	**86.78±1.99**	**76.91±1.17**	**74.70±1.49**	**74.79±1.51**	84.43±0.40	84.31±0.53	84.37±0.44
SF	84.66±1.58	85.75±1.18	85.08±0.87	71.43±1.69	70.96±0.93	69.94±1.94	**85.91±0.69**	**86.38±0.59**	**85.98±0.72**
CaFo-MSE	62.06±1.89	64.83±2.68	66.39±2.03	53.92±3.34	55.80±3.81	55.92±1.83	74.03±1.56	74.51±1.10	75.06±0.96
CaFo-CE	53.22±3.46	56.17±2.90	53.46±4.41	52.25±1.76	53.38±1.30	53.06±1.58	58.75±1.08	59.51±1.14	59.66±1.28
ProF-GNN (ours)	86.26±1.69	86.34±1.35	86.46±1.59	72.07±1.47	71.76±2.28	72.28±2.22	82.94±0.64	83.31±0.43	83.34±0.56

and underlined, respectively. From the results in Table II, we can make a few observations as follows:

- ProF-GNN outperforms most baseline methods and even conventional BP when applied to GCN and GAT on Cora and CiteSeer datasets, demonstrating that the introduction of prototype learning effectively enhances the learning process. However, when applied to the GraphSAGE model, the performance of ProF-GNN is slightly weaker than SymBa. This can be attributed to the neighborhood sampling and aggregation mechanism of GraphSAGE, which only aggregates a smaller size of feature information of local neighborhoods during message passing. As a result, the virtual nodes in ProF-GNN may receive incomplete or biased neighborhood representations, limiting the quality and representativeness of the learned prototypes.
- For all GNN models, ProF-GNN performs stronger than SF on the Cora and CiteSeer datasets, but weaker on the PubMed dataset. Unlike SF, which only considers the similarity between node representations and class-specific virtual nodes, our ProF-GNN additionally tries to separate different prototypes to enhance discriminability. Although this explicit prototype separation is beneficial on noisy graphs with overlapping class manifolds (e.g., Cora and CiteSeer), it provides limited or even negative gains on PubMed, where class prototypes are already well-separated due to strong homophily and smooth label distributions.

C. Memory Efficiency Analysis

To evaluate the memory efficiency of ProF-GNN, we train a GCN model using BP, FF, and the proposed ProF-GNN, and compare their GPU memory consumption during training. Specifically, we record the allocated GPU memory in PyTorch immediately before the backward pass at each epoch and report the average over all epochs. For clarity, all experiments are conducted on the PubMed dataset. The mean GPU memory consumption of the three methods is summarized in Table III.

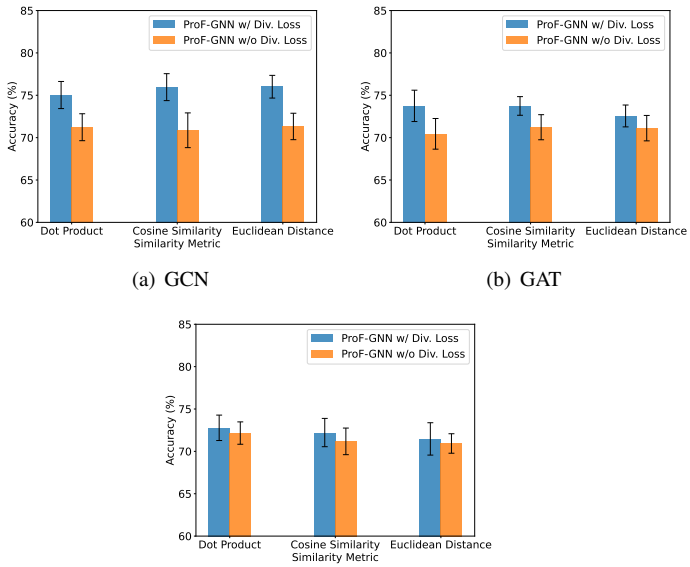
TABLE III
MEAN GPU MEMORY USAGE (MB) OF BP, FF, AND PROF-GNN DURING TRAINING

Methods	2 layers	3 layers	4 layers
BP	183.92	196.64	208.58
FF	225.65	217.62	213.05
ProF-GNN	123.18	125.59	126.29

From Table III, we observe that the FF algorithm incurs higher memory consumption than BP, as it requires additional storage for the hidden representations of negative samples. In contrast, ProF-GNN demonstrates superior memory efficiency compared to both BP and FF, since it only needs to store the output of a single layer at a time during training. It is worth noting that, when an L-layer GCN is trained, the memory consumption of ProF-GNN is not reduced to approximately $1/L$ of that of BP. This is because the calculation of two losses requires creating additional temporary variables to store intermediate results (e.g., similarity values), which partially offset the memory savings.

D. Ablation Study

To further verify the effectiveness of the proposed diversity-aware objective, we compare our ProF-GNN with a variant that removes the diversity loss (i.e. set $\lambda = 0$) while keeping all other conditions unchanged. Furthermore, we adopt different similarity metrics to measure the node-prototype similarity, including dot product, cosine similarity, and Euclidean distance (i.e. $\mathrm{sim}_1(\alpha, \beta) = 1/(1 + \|\alpha - \beta\|_2)$. In this section, the proposed method is denoted as *ProF-GNN w/ Div. Loss* and the variant without the diversity loss is denoted as *ProF-GNN w/o Div. Loss*. For ease of presentation, we only show the results of 3-layer GNNs on CiteSeer dataset in Figure 2.



(a) GCN (b) GAT

(c) GraphSAGE

Fig. 2. The average accuracy of 3-layer GCN, GAT, and GraphSAGE trained by ProF-GNN with and without diversity loss on the CiteSeer dataset.

From the results, we can conclude that removing the diversity loss in ProF-GNN consistently degrades the performance of node classification.

VI. CONCLUSION

In this paper, we investigate the challenges of applying forward-only learning paradigms to GNNs for node classification, particularly in terms of efficiency and training stability. To address these issues, we propose ProF-GNN, a forward-style GNN framework that introduces class-aware virtual nodes as prototypes and enables layer-wise learning without relying on exhaustive label-wise inference or multiple negative samples. By incorporating a diversity-aware objective, ProF-GNN effectively prevents prototype collapse and ensures stable and discriminative representations. Extensive experiments on benchmark datasets demonstrate that ProF-GNN has strong potential as an alternative to BP for training GNNs for node classification.

Our future work will explore extending the proposed framework to other graph learning tasks, such as link prediction and graph classification, as well as investigating more scalable implementations for large-scale graphs.

Acknowledgment

This work was supported by the Natural Science Foundation of Guangdong Province, China (Grant No. 2024A1515510014).

References

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct 1986.

[2] G. E. Hinton, "The forward-forward algorithm: Some preliminary investigations," *CoRR*, vol. abs/2212.13345, 2022.

[3] H. Lee and J. Song, "Symba: Symmetric backpropagation-free contrastive learning with forward-forward algorithm for optimizing convergence," *CoRR*, vol. abs/2303.08418, 2023.

[4] H. Aghagolzadeh and M. Ezoji, "Marginal contrastive loss: A step forward for forward-forward," in *2024 13th Iranian/3rd International Machine Vision and Image Processing Conference (MVIP)*, pp. 1–6, 2024.

[5] G. Zhao, T. Wang, Y. Jin, C. Lang, Y. Li, and H. Ling, "The cascaded forward algorithm for neural network training," *Pattern Recognition*, vol. 161, p. 111292, 2025.

[6] R. Scodellaro, A. Kulkarni, F. Alves, and M. Schröter, "Training convolutional neural networks with the forward–forward algorithm," *Scientific Reports*, vol. 15, p. 38461, Nov 2025.

[7] A. Ororbia and A. A. Mali, "The predictive forward-forward algorithm," *CoRR*, vol. abs/2301.01452, 2023.

[8] F. Giampaolo, S. Izzo, E. Prezioso, and F. Piccialli, "Investigating random variations of the forward-forward algorithm for training neural networks," in *2023 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, 2023.

[9] D. Paliotta, M. Alain, B. Máté, and F. Fleuret, "Graph neural networks go forward-forward," in *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*, 2023.

[10] N. Park, X. Wang, A. Simoulin, S. Yang, G. Yang, R. A. Rossi, P. Trivedi, and N. K. Ahmed, "Forward learning of graph neural networks," in *The Twelfth International Conference on Learning Representations*, 2024.

[11] G. Lee, J. Shin, and H. Kim, "Vff-net: Evolving forward–forward algorithms into convolutional neural networks for enhanced computational insights," *Neural Networks*, vol. 190, p. 107697, 2025.

[12] A. Momeni, B. Rahmani, M. Malléjac, P. del Hougne, and R. Fleury, "Backpropagation-free training of deep physical neural networks," *Science*, vol. 382, no. 6676, pp. 1297–1303, 2023.

[13] Z. Zhang, Q. Liu, H. Wang, C. Lu, and C. Lee, "Protgnn: Towards self-explaining graph neural networks," in *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pp. 9127–9135, AAAI Press, 2022.

[14] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[15] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[16] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.