# Propagation is All You Need: A New Framework for Representation Learning and Classifier Training on Graphs

Jiaming Zhuo
Can Cui
jiaming.zhuo@outlook.com
594021820@qq.com
School of Artificial Intelligence
Hebei University of Technology
Tianjin, China

Kun Fu
Bingxin Niu
fukun@hebut.edu.cn
niubingxin666@163.com
School of Artificial Intelligence
Hebei University of Technology
Tianjin, China

Dongxiao He
hedongxiao@tju.edu.cn
College of Intelligence and Computing
Tianjin University
Tianjin, China

Yuanfang Guo
andyguo@buaa.edu.cn
School of Computer Science and Engineering
Beihang University
Beijing, China

Zhen Wang
w-zhen@nwpu.edu.cn
OPtics and ElectroNics (iOPEN), School of Cybersecurity
Northwestern Polytechnical University
Xi'an, China

Chuan Wang*
wangchuan@iie.ac.cn
State Key Laboratory of Information Security
Institute of Information Engineering
Chinese Academy of Sciences
Beijing, China

Xiaochun Cao
caoxiaochun@mail.sysu.edu.cn
School of Cyber Science and Technology, Shenzhen Campus
Sun Yat-sen University
Shenzhen, China

Liang Yang*
yangliang@vip.qq.com
School of Artificial Intelligence
Hebei University of Technology
Tianjin, China

## ABSTRACT

Graph Neural Networks (GNNs) have been the standard toolkit for processing non-euclidean spatial data since their powerful capability in graph representation learning. Unfortunately, their training strategy for network parameters is inefficient since it is directly inherited from classic Neural Networks (NNs), ignoring the characteristic of GNNs. To alleviate this issue, experimental analyses are performed to investigate the knowledge captured in classifier parameters during network training. We conclude that the parameter features, i.e., the column vectors of the classifier parameter matrix, are cluster representations with high discriminability. And after a theoretical analysis, we conclude that the discriminability of these features is obtained from the feature propagation from nodes to parameters. Furthermore, an experiment verifies that compared with cluster centroids, the parameter features are more potential for augmenting the feature propagation between nodes. Accordingly, a novel GNN-specific training framework is proposed by simultaneously updating node representations and classifier parameters via a unified feature propagation scheme. Moreover, two augmentation schemes are implemented for the framework, named Full Propagation Augmentation (FPA) and Simplified Full Propagation Augmentation (SFPA). Specifically, FPA augmentates the feature propagation of each node with the updated classifier parameters. SFPA only augments nodes with the classifier parameters corresponding to their clusters. Theoretically, FPA is equivalent to optimizing a novel graph learning objective, which demonstrates the universality of the proposed framework to existing GNNs. Extensive experiments demonstrate the superior performance and the universality of the proposed framework.

## CCS CONCEPTS

**• Computing methodologies → Machine learning**; **• Networks → Network algorithms**.

## KEYWORDS

Graph neural network, Representation learning, Network training

**ACM Reference Format:**
Jiaming Zhuo, Can Cui, Kun Fu, Bingxin Niu, Dongxiao He, Yuanfang Guo, Zhen Wang, Chuan Wang, Xiaochun Cao, and Liang Yang. 2023. Propagation is All You Need: A New Framework for Representation Learning and Classifier Training on Graphs. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23), October 29–November 3, 2023, Ottawa, ON, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3581783.3612196

*Corresponding authors.

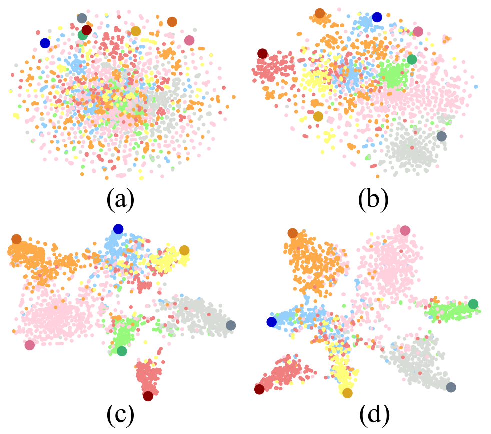


**Figure 1: T-SNE visualization of node and parameter features at four stages. (a) Before training, (b) After 5 iterations, (c) After 20 iterations, (d) After 100 iterations. The embeddings show clear clusters where each parameter embedding (large dot) is further away from the cluster boundary than node embeddings (small dots) in the same cluster.**

## 1 INTRODUCTION

Graph Neural Networks (GNNs), which follow the message-passing mechanism, have shown promising results in various graph-based tasks, including link prediction [30], node classification [10, 22, 25], graph classification [31] and ranking [9, 13, 14]. Compared with classic Neural Networks (NNs), e.g., Multilayer Perceptron (MLP) [18], the superiority of GNNs is that they collectively utilize graph topology and node attributes in a feature propagation[1] manner. Nevertheless, the training scheme for network parameters has been underemphasized in prior research. Specifically, parameter matrices are often treated solely as dimension reduction components, and their training strategy in Graph Neural Networks (GNNs) is directly inherited from Neural Networks (NNs). The lack of a specific understanding and in-depth exploration of these learnable parameters may impose substantial limitations on developing more robust and efficient GNN architectures.

To alleviate this issue, the node features and parameter features collected at several stages are first visualized in Figure 1, where parameter features are the column vectors of the classifier parameter matrix. The result shows that parameter embeddings, i.e., low-dimensional representations of the parameter features, have a significant separation from the cluster boundary compared to node embeddings. Based on this phenomenon, a conjecture is introduced that the parameter features are more distinguishable than the cluster centroids, which are the mean of a class of node features. Subsequently, the distinguishability of pairs of cluster representations is measured and visualized in Figure 2, and the result confirms the above conjecture. Moreover, after theoretical analysis, it is concluded that the discriminability of the parameter features is empowered by a feature propagation[2] on the class-aware matrix, which is composed of node labels and the above two types of features.

[1]It denotes the propagation between nodes.
[2]It denotes the propagation to update the parameter.

Inspired by HP-GMN [26], which utilizes cluster centroids to augment the propagation between nodes, the superiority of parameter features over these centroids under this scheme is verified experimentally, as shown in Figure 3. Accordingly, a full propagation training framework for GNNs is presented by simultaneously updating node and parameter features via a unified feature propagation scheme. Furthermore, two implementations of the proposed framework are introduced, named Full Propagation Augmentation (FPA) and Simplified Full Propagation Augmentation (SFPA), as shown in Figure 4. FPA augments the feature propagation of each node with the updated classifier parameters, ensuring seamless connectivity throughout the graph. In contrast, SFPA only augments nodes with the classifier parameters corresponding to their clusters, promoting an efficient propagation pattern.

Theoretically, by proving the equivalence of FPA and the optimization of a graph learning objective, which consists of Graph Learning object [15, 24, 28, 33] and Cross-Entropy loss, the universality of the proposed framework to existing GNNs is demonstrated. Finally, numerous experiments on nine homophilic and heterophilic benchmarks, including performance evaluation, label efficiency, over-smoothing analysis, and complexity analysis, provide evidence of the effectiveness and universality of this framework.

The main contributions of this paper are summarized as follows:

- We experimentally investigate the knowledge captured in classifier parameters and provide an interpretation for their discriminability from a feature propagation perspective.
- We propose a novel framework for training GNNs and two augmentation schemes for implementing it, where node and parameter features are updated in a propagation manner.
- We provide a novel optimization objective for extending the proposed framework and demonstrate the universality of the proposed framework to existing GNNs.
- We experimentally verify the effectiveness and the universality of the proposed framework on both homophilic and heterophilic benchmarks.

## 2 RELATED WORK

The Graph Convolutional Network (GCN) [10] pioneers message-passing mechanisms for various graph-based tasks. Building on GCN, several variants have been introduced, including Simplified Graph Convolution (SGC) [25] and Graph Attention Networks (GAT) [22], which employ efficient convolution operations and attention mechanisms to selectively aggregate neighboring information. Addressing over-smoothing issues, APPNP [11] combines graph propagation and neural networks for capturing long-range dependencies using Personalized PageRank. Similarly, the GCNII [4] method utilizes identity mapping to enhance information propagation efficiency. For handling heterophily issues, several techniques have been developed, such as FAGCN[1], which incorporate beyond low-pass filters, and Geom-GCN[16], and HP-GMN [26], which supplement information by considering neighborhoods on a latent space and the features in a memory block. Furthermore, GPR-GNN [5] underlines positive or negative messages through learnable aggregation weights, while H2GCN [32] emphasizes the separation of the different orders. Recently, [3] has successfully extended the

AUC optimization framework to deal with classification tasks on Graph.

## 3 PRELIMINARIES

This section presents the concise notations used in this paper, followed by the preliminaries on graph neural networks.

### 3.1 Notations

Given an undirected graph $\mathcal{G} = (\mathcal{E}, \mathcal{V})$, where $\mathcal{E}$ denotes the edge set and $\mathcal{V}$ means the node-set, $N$ is the number of nodes. Graph $\mathcal{G}$ is described by the adjacency matrix $\mathbf{A} \in R^{N \times N}$. $\mathbf{D}$ denotes the diagonal degree matrix, $\mathbf{D}_{i,i} = \sum_j \mathbf{A}_{i,j}$. $\mathbf{L}$ denotes the symmetric positive semidefinite laplacian matrix, $\mathbf{L} = \mathbf{D} - \mathbf{A}$. Generally, one common normalization is $\tilde{\mathbf{A}} = (\mathbf{D} + \mathbf{I}_N)^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I}_N)(\mathbf{D} + \mathbf{I}_N)^{-\frac{1}{2}}$, and hence $\tilde{\mathbf{L}} = \mathbf{I}_N - \tilde{\mathbf{A}}$, where $\mathbf{I}_N$ is an all-one diagonal matrix. $\mathbf{X} \in R^{N \times F}$ represents the node attribute matrix, where $F$ is the dimension of attributes. $\mathbf{Y} \in R^{N \times C}$ denotes the label matrix. $C$ is the number of classes. $N_L$ is the number of the labeled nodes.

### 3.2 Message Passing Graph Neural Networks

Most current Graph Neural Networks (GNNs) follows a propagation-transformation strategy. The propagation operator, supported by the Message-Passing paradigm [6, 21], iteratively updates the features of each node by aggregating these of their neighbors. The transformation operator encodes input features into a low dimensional space, typically a learnable network, e.g., MLP [18]. For the $i$-th node, the update scheme of its feature in the $k$-th layer can be formulated as

$$Propagation \quad \mathbf{H}^{k+1}_{i:} = \mathbf{P}^{k}_{i,i}\mathbf{Z}^{k}_{i:} + \sum_{j \in N_i} \mathbf{P}^{k}_{i,j}\mathbf{Z}^{k}_{j:} \tag{1}$$

$$Transformation \quad \mathbf{Z}^{k+1}_{i:} = \sigma(\mathbf{H}^{k+1}_{i:}\mathbf{W}^{k}) \tag{2}$$

where $\mathbf{H}$ and $\mathbf{Z}$ denote the node feature and the actual output, respectively, the scalar $\mathbf{P}_{i,j}$ denotes the aggregation weight, $\sigma$ terms the nonlinear activation functions, $\mathbf{W}$ terms the network parameters, $N_i$ is the neighbor set. The first and second terms represent the node and neighbors' messages.

The main difference between various GNN models is that it adopts different propagation-transformation schemes. For example, GCN [10] applies propagation and transformation at each layer, whereas SGC only performs transformation at the final layer. Additionally, APPNP [11] combines the initial feature, the output of the transformation layers, at each layer using Skip Connection [8].

$$GCN \quad \mathbf{H}^{k+1} = \tilde{\mathbf{A}}\mathbf{H}^{k}\mathbf{W}^{k} \tag{3}$$

$$SGC \quad \mathbf{H}^{k+1} = \tilde{\mathbf{A}}\mathbf{H}^{k} \tag{4}$$

$$APPNP \quad \mathbf{H}^{k+1} = (1 - \alpha)\tilde{\mathbf{A}}\mathbf{H}^{k} + \alpha\mathbf{H}^{0} \tag{5}$$

where $\alpha$ denotes the balance hyperparameter.

**Objectives.** A particular set of parameters of GNN networks is computed by optimizing objectives, which minimizes an error between the actual and desired output vectors for every node in the training set. Actually, in GNNs [10, 11, 22, 25], the actual vectors $\mathbf{Z}$ is the product of the node feature matrix $\mathbf{H} \in R^{N \times D}$ and the parameter matrix $\mathbf{W} \in R^{D \times C}$, that is $\mathbf{Z} = \mathbf{HW}$. Hence, the total error is presented by Cross-Entropy loss as

$$E = \sum_i \mathbf{Y}_{i,:}\, ln\,(softmax\,(\mathbf{Z}_{i,:})) \tag{6}$$

where $ln$ is the natural logarithm, $softmax$ is an activation function, which is often employed for multi-class classification, $\mathbf{Z}$ is the actual state of an output unit, and $\mathbf{Y}$ is its desired state. To optimize the objective by Gradient Descent, it is necessary to compute the partial derivative of E with respect to the network parameters.

### 3.3 Graph Learning Objective

Several works have demonstrated the equivalence between the existing GNNs and the optimization objectives [15, 24, 28, 33]. One of the most commonly utilized objectives can be formulated as

$$\arg\min_{\mathbf{H}} \; \|\mathbf{H} - \mathbf{H}^{0}\|_F^2 + \lambda\, tr(\,\mathbf{H}^{T}\tilde{\mathbf{L}}\mathbf{H}\,) \tag{7}$$

where $tr$ terms the matrix trace, $\mathbf{H}^0$ denotes the initial feature, $\lambda$ is the balance coefficient between the above two terms. By minimizing the first term, the updated feature $\mathbf{H}$ is guaranteed to be similar to $\mathbf{H}^0$. The second term has the same mathematical form as Spectral Clustering, which follows the smoothness assumption on graph [23]. Therefore, minimizing it forces connected nodes to have similar features. Ultimately, by optimizing Equation 7 with Gradient Descent, a batch of GNNs can be induced[15, 24, 28, 33], including GCN[10], SGC[25], GAT[22], APPNP[11] and JKNet[27].

## 4 ANALYSIS

This section begins by introducing the setups and critical findings of two experiments. Subsequently, a theoretical analysis of the conjecture of the experimental findings is provided.

### 4.1 Experimental Discovery

The node and parameter features are first collected at several stages and visualised to investigate the knowledge captured by classifier parameters during network training.

**Experimental setup.** Precisely, to match the dimension of the node features, the parameter features are first depicted as the D-dimensional row vectors of $\bar{\mathbf{W}}$, where $\bar{\mathbf{W}} \in R^{D \times C}$ represents the matrix transpose of the classifier parameter matrix $\mathbf{W}$. Next, the node and parameter features in four states are saved during the GCN [10] network training on the Cora dataset. Using the visual tool t-SNE [20], the low-dimensional embeddings of these two types of features are generated and visualized, where the category of nodes is the node label, and the category of parameter features is its row index in the matrix $\bar{\mathbf{W}}$. In Figure 1, each color represents a category, and small light and large dark dots stand for node and parameter embeddings, respectively.

As shown in Figure 1, each cluster consists of small dots and a large dot, all of which are of the same color. From this, it can be inferred that the parameter features may represent a cluster. Meanwhile, Figure 1 shows that each large dot is further away from the cluster boundary than the small dots. As of now, the reason behind this occurrence is not ensured. We conjecture that the farther the embedding is from the cluster boundary, the more discriminative power their features possess. This means parameter features

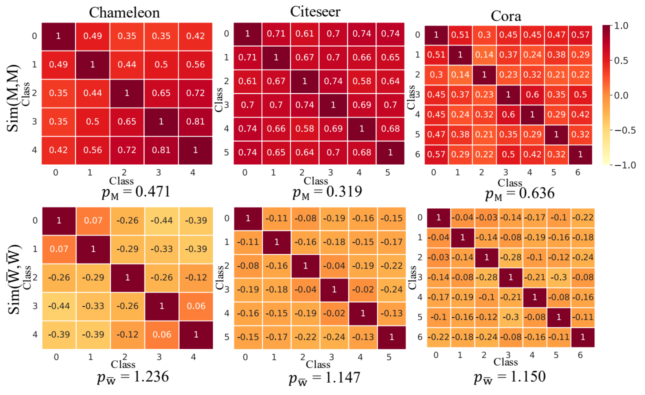


**Figure 2: Comparison of the discriminability $p$ of cluster centroids $\mathbf{M}$ and parameter feature $\bar{\mathbf{W}}$. Each heatmap grid stands for the cosine similarity of a cluster pair. Parameter features have higher discriminability than cluster centroids.**

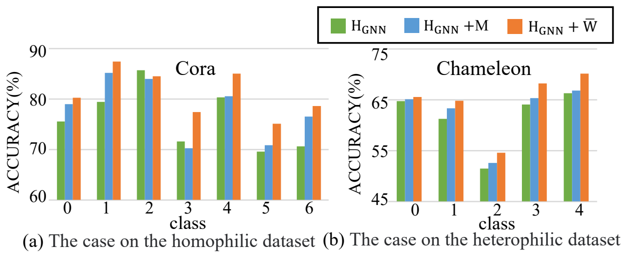


**Figure 3: Performance improvement by using cluster representations to augment feature propagation among nodes. The green, blue, and orange bars stand for SGC, the variant with $\mathbf{M}$, and the variant with $\bar{\mathbf{W}}$. Parameter features have a stronger ability to augment feature propagation among nodes than cluster centroids.**

are more effective in distinguishing between classes than node features.

*Definition 4.1.* (Discriminability of cluster representation) Given a cluster representation matrix $\mathbf{U}$, its discriminability is defined as

$$p_{\mathbf{U}} = 1 - \frac{1}{C(C-1)} \sum_{i=0}^{C-1} \sum_{\substack{j=0,\\ j\neq i}}^{C-1} sim(\mathbf{U}_{i,:}, \mathbf{U}_{j,:}) \tag{8}$$

where $sim$ is the cosine similarity, $C$ is the number of categories.

As per Definition 4.1, an excellent cluster representation ought to possess high discriminability, leading to a precise indication for the node clustering task. Based on this metric, the discriminability of two kinds of cluster representations, i.e., node and parameter features, are measured and visualized.

**Experimental setup.** To simplify the process of comparing parameter features with the node features in the same class, the centroid, a widely-used cluster representation [7, 26], is employed to stand for the node features. Each centroid represents the average of the feature of the nodes in the same cluster. Then, the cosine similarity between all pairs of clusters, e.g., centroids, is computed and represented as $sim(\mathbf{M}_{i,:}, \mathbf{M}_{j,:})$. To visually compare the discriminability of parameter features $\bar{\mathbf{W}}$ and cluster centroids $\mathbf{M}$, heatmaps on various datasets are provided, where each cell represents a cosine similarity value. The node and parameter features are the actual output of the SGC model after 100 training iterations.

When comparing the upper and lower heatmaps shown in Figure 2, it is evident that the cosine similarities between any two parameter features are not lower than those of the cluster centroid counterpart. Here, the comparison only considers different clusters. Therefore, it can be easily deduced that $p_{\bar{\mathbf{W}}} > p_{\mathbf{M}}$. In conclusion, the results confirm the conjecture that parameter features are more effective in discriminating between different classes than node features.

## 4.2 Theoretical Explanation

To further examine the conclusion stated above, a comprehensive analysis is conducted on the updating process of the parameter features. Then, an interpretation of the above phenomenon is presented from a feature propagation perspective.

Proposition 4.2. *The scheme of updating parameter features $\bar{\mathbf{W}}$, which is induced by the optimization of Equation 6, is equivalent to a feature propagation from nodes $\mathbf{H}$ to parameters $\bar{\mathbf{W}}$.*

To prove this proposition, Lemma 4.3 is first introduced.

Lemma 4.3. *Given the $j$-th parameter features, i.e., the column vector with index $j$, its solution with Equation 6 satisfies:*

$$\bar{\mathbf{W}}_{j,:} = \bar{\mathbf{W}}_{j,:} + \epsilon \sum_{i=0}^{N_L-1} \mathbf{O}_{j,i}\, \mathbf{H}_{i,:} \tag{9}$$

*where $\mathbf{O} \in R^{C\times N}$ represents the matrix $(\mathrm{Y}-softmax(\mathbf{H}\bar{\mathbf{W}}^T))^T$, and $\epsilon$ terms the learning rate of the model optimizer.*

Proof. To begin with, the partial derivative of E with respect to the parameter features $\bar{\mathbf{W}}$ is calculated as

$$\frac{\partial E}{\partial \bar{\mathbf{W}}} = -2(\mathrm{Y} - softmax(\mathbf{H}\bar{\mathbf{W}}^T))^T\mathbf{H} \tag{10}$$

Secondly, with the Gradient Descent method, that is $\bar{\mathbf{W}} = \bar{\mathbf{W}} - \epsilon * \frac{\partial E}{\partial \bar{\mathbf{W}}}$, the solution of $\bar{\mathbf{W}}$ can be formulated as

$$\bar{\mathbf{W}} = \bar{\mathbf{W}} + \epsilon\,(\mathrm{Y} - softmax(\mathbf{H}\bar{\mathbf{W}}^T))^T\mathbf{H} \tag{11}$$

Finally, by expanding the Equation 11 row by row and substituting $(\mathrm{Y} - softmax(\mathbf{H}\bar{\mathbf{W}}^T))^T$ with $\mathbf{O}$, the proof of Lemma 4.3 ends. □

Consequently, the first two terms of Equation 9 indicate the combination of the parameter features themself and their neighbor features, respectively. The above operator is the same as the GNN propagation operator, formulated as Equation 1. Thus, the proof of Proposition 4.2 concludes.

Proposition 4.2 demonstrates that when the classifier is considered separately, the updating process of its parameters can be viewed as a feature aggregation over all labeled nodes using a class-aware weight matrix.

Lemma 4.4. *Consider the matrix $\mathbf{O} = (\mathrm{Y} - softmax(\mathbf{H}\bar{\mathbf{W}}^T))^T$, derived from the optimization of Equation 6, if $i$-th node belongs to class $j$, that is $\mathrm{Y}_{i,j} = 1$, there is the weight $\mathbf{O}_{j,i} \geq 0$, or if $\mathrm{Y}_{i,j} = 0$, there is the weight $\mathbf{O}_{j,i} \leq 0$.*

Proof. Let S represents the matrix $softmax(\mathbf{HW})$, where each value satisfies $\mathrm{S}_{i,j} \in [0,1]$. It can be deduced from matrix operations, $\mathbf{O}_{j,i} = \mathbf{O}^T_{i,j} = \mathrm{Y}_{i,j} - \mathrm{S}_{i,j}$. Thus, if $\mathrm{Y}_{i,j} = 1$, there is $\mathbf{O}_{j,i} = 1 - \mathrm{S}_{i,j} \geq 0$, or if $\mathrm{Y}_{i,j} = 0$, there is $\mathbf{O}_{j,i} = 0 - \mathrm{S}_{i,j} \leq 0$. □

Proposition 4.5. *Based on Lemma 4.4, given a parameter features $\bar{\mathbf{W}}_{i:}$, the propagation mechanism, which is described in Proposition 4.2, increases its similarity with the same-class nodes while reduces its similarity with the different-class nodes.*

Considering that the entire propagation process operates on a one-hot node label matrix, it manifests as a piecewise function with two components. The first component involves feature propagation within the same class, while the second component involves feature propagation across different classes, as

$$\bar{\mathbf{W}}_{j,:} = \begin{cases} \bar{\mathbf{W}}_{j,:} - \epsilon \sum_{i=0}^{N_L-1} \mathrm{S}_{i,j}\,\mathbf{H}_{i,:}\,, & \mathrm{Y}_{i,j} = 0 \\ \bar{\mathbf{W}}_{j,:} + \epsilon \sum_{i=0}^{N_L-1} (1-\mathrm{S}_{i,j})\,\mathbf{H}_{i,:}\,, & \mathrm{Y}_{i,j} = 1 \end{cases} \tag{12}$$

Through this propagation mechanism, the features of each parameter are combined with features from nodes of the same class using a positive weight. This results in a significant increase in feature similarity among nodes within the same class. Conversely, it reduces the feature similarities between the parameter and nodes from different classes because of the presence of the negative weight.

Proposition 4.5 demonstrates that the scheme of training parameters, which performs intra-class smoothing (to increase similarity) and inter-class sharpening (to decrease similarity), empowers the parameter features with high discriminability.

## 5 METHODOLOGY

This section first verifies the effectiveness of parameter features in augmenting the feature propagation between nodes. Next, a training framework for GNNs is proposed by simultaneously updating node and parameter features. Moreover, two augmentation schemes are implemented for the framework. Finally, from an optimization perspective, the universality of the framework is demonstrated.

### 5.1 Augmentation of the classifier parameters

As discussed in previous section, parameter features have higher discriminability than the centroids of the same class. And inspired by HP-GMN [26], which utilizes cluster centroids obtained from unsupervised K-Means [7] to augment the feature propagation between nodes, a conjecture is proposed, namely **the parameter features performs better than the cluster centroids in augmenting topological propagation**.

Two variant models $\mathbf{H}_{GNN} + \mathbf{M}$ and $\mathbf{H}_{GNN} + \bar{\mathbf{W}}$, were first proposed to represent the feature propagation between augmenting the cluster centroid and parameter features. Next, these networks, whose backbone model is SGC, are trained on the homophilic Cora dataset, and heterophilic Chameleon dataset, and the node classification performance for each class is reported in Figure 3. In the experiment, each node combines not only the feature of its topology neighbors but also that of two kinds of its non-local neighbors, i.e., cluster centroids $\mathbf{M}$ and classifier parameters $\bar{\mathbf{W}}$, respectively. The combination weights are the inner product of features.

A significant observation in Figure 3, $\mathbf{H}_{GNN} + \mathbf{M}$, namely the augmentation with the cluster centroids, realizes performance improvement on almost all classes, while $\mathbf{H}_{GNN} + \bar{\mathbf{W}}$, namely the augmentation with the classifier parameters, achieve a more remarkable improvement, which confirms the speculation.

The feature propagation mechanism, which is formulated as Equation 1, exhibits several inherent drawbacks. First, the receptive field is seriously restricted by graph topology, making GNNs prone to over-smoothing problems while capturing long-range dependencies. Second, the propagation process lacks sufficient class awareness, rendering GNNs less effective in handling non-homophilic graphs. However, by integrating these parameters during the propagation, nodes can effectively acquire long-range information and enhance their ability to discriminate among local neighbors.

### 5.2 A Full Propagation Training Framework

Based on Proposition 4.2 and experimental findings in Subsection 5.1, we propose a novel GNN training framework. This framework leverages the mutual propagation of classifier parameters and node features, enhancing the quality of node representations. The overview of the framework is in Figure 4. It's worth noting that the node set is extended with classifier parameters as another type of node, called Cluster Anchor nodes (CA nodes).

The feature updating of each node can be described as the combination of the node itself and its neighbors on the augmented graph.

$$Updating\,CA\,nodes\;\; \bar{\mathbf{W}}_{i,:} = \bar{\mathbf{W}}_{i,:} + \sum_{j\in Q_i} \mathcal{A}_{i,j}\,\mathbf{H}_{j,:} \tag{13}$$

$$Updating\,ordinary\,nodes\;\; \mathbf{H}_{i,:} = (\mathbf{H}_{\mathsf{GNN}})_{i,:} + \sum_{j\in \hat{Q}_i} \hat{\mathcal{A}}_{i,j}\,\bar{\mathbf{W}}_{j,:} \tag{14}$$

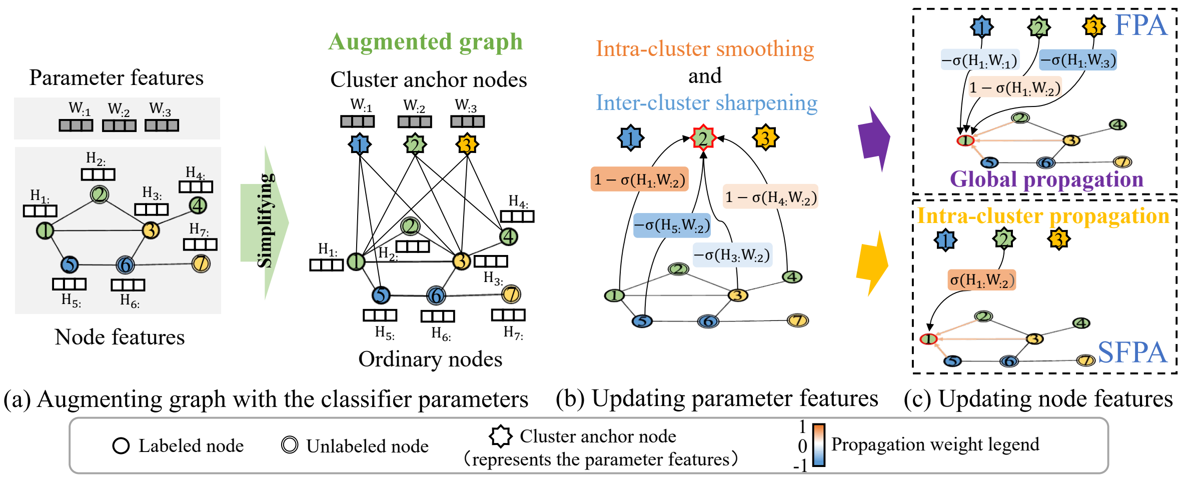
where $\bar{\mathbf{W}}$ and $\mathbf{H}$ denote the features of CA nodes and ordinary nodes, respectively. $Q_i$ and $\hat{Q}_i$ are the neighbor set of $i$-th CA node and $i$-th ordinary node, respectively, $\mathbf{H}_{\mathsf{GNN}}$ stands for the GNN feature, which is obtained by the propagation between ordinary nodes, $\mathcal{A} \in R^{C\times N_L}$ and $\hat{\mathcal{A}} \in R^{N_L\times C}$ denotes two weight matrices, which can be constructed through heuristics or learning.

The proposed framework has several advantages. Firstly, it unifies the mutual propagation of two types of nodes. Secondly, it is compatible with most existing GNNs since the propagation in GNNs can be viewed as an inner-layer update in Equation 14. Lastly, it helps surpass the previous limit of GNNs' expressive power by considering long-range dependencies [17].

### 5.3 Implementations

Two augmentation scheme are implemented for the framework, named Full Propagation Augmention (FPA) and Simplified FPA (SFPA). They correspond to the updating process of the feature of CA nodes, i.e., the classifier parameter, and the features of ordinary nodes, shown in Figure 4 (b) and 4 (c), respectively.

**Updating the features of CA nodes.** As discussed in Section 4 and Subsection 5.1, training with Cross-Entropy loss enhances the classifier parameter's discriminative power, which can augment topological propagation. Thus, FPA and SFPA both follow this scheme as their initial step. For Equation 13, the matrix $\mathcal{A}$ is set to



(a) Augmenting graph with the classifier parameters (b) Updating parameter features (c) Updating node features

**Figure 4: Overview of the Full Propagation Training Framework. (a) Augmenting graph with the classifier parameters. (b) Updating the parameter features with a class-aware matrix, which incorporates label and feature similarity, achieving intra-cluster smoothing and inter-cluster sharpening among the labeled nodes. (c) FPA updates the features of all labeled nodes, while SFPA only performs among the same cluster.**

the following form, which is released from Equation 9.

$$\mathcal{A} = \epsilon(\mathrm{Y}_L - softmax(\mathbf{H}_L \bar{\mathbf{W}}^T))^T \tag{15}$$

where $\epsilon$ denotes a combination coefficient, $\mathrm{Y}_L$ and $\mathbf{H}_L$ term the label and feature of the labeled nodes, respectively. Finally, thanks to the class-awareness of matrix $\mathcal{A}$, the CA node performs intra-cluster smoothing and inter-cluster sharpening in the feature space, as presented in Section 4.

**Full Propagation Augmentation (FPA).** Especially considering the class-aware capabilities of matrix $\mathcal{A}$, FPA set its transpose as the propagation matrix $\hat{\mathcal{A}}$ for updating node features.

$$\hat{\mathcal{A}} = \mathrm{Y}_L - softmax(\mathbf{H}_L \bar{\mathbf{W}}^T) \tag{16}$$

The module is shown in the upper part of Figure 4 (c), where each node receives all CA nodes' features using the class-aware weights, which distinguish between inside and outside the cluster.

**Simplified Full Propagation Augmentation (SFPA).** As shown in the lower part of figure 4(c), the feature propagation of Simplified Full Propagation Augmentation (SFPA) occurs only within the cluster. The matrix $\hat{\mathcal{A}}$ for Equation 14 is represented as:

$$\hat{\mathcal{A}} = \mathrm{Y}_L \odot softmax(\mathbf{H}_L \bar{\mathbf{W}}^T) \tag{17}$$

where $\odot$ represents the Hadamard product, i.e., element-wise multiplication. It is worth noting that extending the scope of the action using pseudo labels is suboptimal for the proposed implementations. Since their low accuracy inevitably leads to an error propagation augmentation.

The advantages of the two schemes are as follows. Firstly, neither of them increases space complexity since no additional parameters are introduced. Secondly, FPA is easy to understand with a straightforward derivation process. Concretely, it is obtained from the joint optimization of graph regularization and classification loss, which have been widely explored. Thirdly, compared to FPA, SFPA is more efficient without losing discriminability. The propagation of SFPA retains the intra-class part of that of FPA but with a sparser propagation matrix.

## 5.4 The Unified Optimization Perspective

To extend the proposed framework, a novel optimization objective is introduced, which collectively targets classifier training and node representation learning. The objective $E_G$ includes initial feature constraint, laplacian graph regularization, and classification error. $E_G$ can be formulated as

$$E_G = \|\mathbf{H} - \mathbf{H}^0\|_F^2 + \lambda\, tr(\mathbf{H}^T \tilde{\mathbf{L}} \mathbf{H}) + \gamma\, \mathcal{D}(\mathrm{Y}, \sigma(\mathbf{H}\bar{\mathbf{W}}^T)) \tag{18}$$

where $\mathcal{D}(,)$ denotes the distance measure, $\sigma$ terms the nonlinear activation function, $\lambda$ and $\gamma$ are hyperparameters.

Theorem 5.1. *The optimization scheme of $E_G$ for node feature* $\mathbf{H}$ *is equivalent to the implementation of FPA with the backbone model APPNP, where $\mathcal{D}$ and $\sigma$ are the same as their settings in Equation 6.*

Proof. Based on these conditions, Equation 18 can be reformulated as combining conventional graph learning object and cross-entropy loss for classification.

$$E_G = \|\mathbf{H} - \mathbf{H}^0\|_F^2 + \lambda\, tr(\mathbf{H}^T \tilde{\mathbf{L}} \mathbf{H}) + \gamma\, \mathrm{Y}\, ln\, (softmax\, (\mathbf{H}\bar{\mathbf{W}}^T)) \tag{19}$$

The partial derivative of $E_G$ with respect to node feature $\mathbf{H}$ is

$$\frac{\partial E_G}{\partial \mathbf{H}} = (\mathbf{H} - \mathbf{H}^0) + \lambda \tilde{\mathbf{L}}\mathbf{H} + \gamma\, (\mathrm{Y} - softmax(\mathbf{H}\bar{\mathbf{W}}^T))\bar{\mathbf{W}} \tag{20}$$

Next, let $\frac{\partial E_G}{\partial \mathbf{H}} = 0$, the iterative solution of $\mathbf{H}$ is

$$\mathbf{H}^k = (1-\alpha)\tilde{\mathbf{A}}\mathbf{H}^{k-1} + \alpha\mathbf{H}^0 + \gamma(\mathrm{Y} - softmax(\mathbf{H}^{k-1}\bar{\mathbf{W}}^T))\bar{\mathbf{W}} \tag{21}$$

where $\alpha = \frac{1}{1+\lambda}$ and $\gamma = \frac{\gamma}{1+\lambda}$ are hyperparameters. It can be observed that the first two terms in Equation 21 correspond to the feature updating process in APPNP. Moreover, the propagation, formulated by the third term in Equation 21, is the same as the function of FPA. □

Illustrated by Theorem 5.1, the universality of the proposed framework is validated from an optimization perspective. The overall performance could gain by replacing the first two terms in Equation 18 with the new objectives for graph learning, e.g., GNN-LF/HF [33] and tsGCN [24].

**Table 1: Statistics of nine graph datasets. #Edge Hom is the edge homophily raised in H2GCN [32].**

| **Dataset** | Nodes | Edges | Features | Classes | #Edge Hom |
|---|---|---|---|---|---|
| Cora | 2,708 | 5,278 | 1,433 | 7 | 0.81 |
| Citeseer | 3,327 | 4,552 | 3,703 | 6 | 0.74 |
| Pubmed | 19,717 | 44,324 | 500 | 3 | 0.80 |
| Penn94 | 41,554 | 1,362,229 | 5 | 2 | 0.47 |
| Cornell5 | 18,660 | 790,777 | 5 | 2 | 0.48 |
| Genius | 421,961 | 984,979 | 12 | 2 | 0.62 |
| Chameleon | 2,277 | 36,101 | 2,325 | 5 | 0.23 |
| Squirrel | 5,201 | 217,073 | 2,089 | 5 | 0.22 |
| Actor | 7,600 | 33,544 | 931 | 5 | 0.22 |

## 6 EXPERIMENTS

This section first provides experimental setups, including datasets, baselines, and implementation details. Then, the node classification results are analyzed, followed by hyper-parameter tuning. Finally, the capabilities of preventing over-smoothing and the low computational cost are verified.

### 6.1 Experimental Setup

**Datasets.** To exhaustively evaluate the proposed models, nine widely used benchmark datasets with various homophily are employed. They can be categorized into four types, and their statistics are shown in Table 1. Cora, Pubmed, and Citeseer are citation networks [29]. Penn94, Cornell5, and Genius are social networks [12]. Chameleon and Squirrel are Wikipedia page-page networks [19]. Actor is a Co-occurrence network [16] . In this study, the datasets with edge homophily [32] greater than 0.5 are homophily datasets, and otherwise, heterophilic datasets. For Cora, Citeseer, and Pubmed, 20 nodes per class for training, 500 nodes for validation, and 1000 nodes for testing are randomly sampled. For Penn94, Cornell5, and Genius, the nodes are randomly split into 2.5%, 2.5%, and 95% for training, validation, and testing, respectively. For Chameleon, Squirrel, and Actor, the widely used split provided in Geom-GCN [16] is utilized, where the training, validation, and testing proportions are 60%, 20% and 20%, respectively.

**Baselines.** To verify the superiority of the proposed models, ten baseline models are employed for performance comparison. These methods are divided into two categories. The first category consists of the basic models for homophilic graph data, including GCN [10], SGC [25], APPNP [11], GAT [22], JKNet [27], GCNII [4]. The methods in the second category possess the adaptability of heterophilic graph data, including FAGCN [1], GPRGNN [5], Geom-GCN [16] and H2GCN [32]. It is nothing that both the proposed implementations are based on GPRGNN.

**Implementation Details.** For the network training of each model, the Adam optimizer is selected, the learning rate is among the set $\{0.005, 0.01, 0.05, 0.1\}$, the weight decay is among the set $\{0, 0.0001, 0.001, 0.005, 0.01\}$, the dimensions of hidden layers are 64, and the dropout rate is among the set $\{0, 0.2, 0.5\}$. The remaining hyperparameter is consistent with the original paper. For the unique hyperparameter of the proposed framework, i.e., $\gamma$, its value is among the set $\{0.2, 0.4, 0.6, 0.8\}$. Node classification is the primary task of performance verification of the proposed framework, and accuracy is the key evaluation metric. All experiments are run ten times, and the average value and standard deviation are reported.

**Table 2: Mean classification accuracy and standard deviation on heterophilic datasets (Bold indicates the best.)**

| Model | Cornell5 | Chameleon | Squirrel | Actor |
|---|---|---|---|---|
| SGC | $61.63_{\pm 2.55}$ | $56.97_{\pm 3.77}$ | $41.44_{\pm 2.39}$ | $28.71_{\pm 1.22}$ |
| GCN | $63.80_{\pm 3.21}$ | $59.82_{\pm 2.58}$ | $36.89_{\pm 1.34}$ | $30.64_{\pm 1.49}$ |
| APPNP | $63.41_{\pm 2.59}$ | $52.57_{\pm 1.82}$ | $33.29_{\pm 1.72}$ | $29.94_{\pm 0.70}$ |
| GAT | $62.45_{\pm 2.17}$ | $60.26_{\pm 2.50}$ | $40.72_{\pm 1.55}$ | $28.62_{\pm 0.68}$ |
| JKNet | $59.42_{\pm 2.58}$ | $62.31_{\pm 2.76}$ | $44.24_{\pm 2.11}$ | $36.47_{\pm 0.51}$ |
| GCNII | $57.19_{\pm 4.64}$ | $63.02_{\pm 1.37}$ | $41.17_{\pm 2.80}$ | $36.18_{\pm 0.61}$ |
| FAGCN | $65.39_{\pm 2.33}$ | $61.12_{\pm 1.95}$ | $40.88_{\pm 2.02}$ | $36.81_{\pm 0.26}$ |
| GPRGCN | $64.74_{\pm 1.90}$ | $63.43_{\pm 1.77}$ | $47.29_{\pm 2.43}$ | $36.58_{\pm 1.04}$ |
| Geom-GCN | $63.58_{\pm 1.62}$ | $60.31_{\pm 1.53}$ | $38.32_{\pm 1.59}$ | $31.63_{\pm 0.98}$ |
| H2GCN | $66.16_{\pm 2.24}$ | $58.79_{\pm 1.93}$ | $37.90_{\pm 2.02}$ | $36.45_{\pm 1.16}$ |
| FPA(ours) | $68.02_{\pm 1.33}$ | $65.02_{\pm 1.49}$ | $49.39_{\pm 1.04}$ | $\mathbf{36.81}_{\pm 1.09}$ |
| SFPA(ours) | $\mathbf{68.75}_{\pm 0.94}$ | $\mathbf{66.16}_{\pm 1.52}$ | $\mathbf{50.93}_{\pm 1.16}$ | $36.72_{\pm 0.79}$ |

**Table 3: Mean classification accuracy and standard deviation on homophilic datasets (Bold indicates the best.)**

| Model | Cora | Citeseer | Pubmed | Genius |
|---|---|---|---|---|
| SGC | $81.81_{\pm 0.27}$ | $71.04_{\pm 0.45}$ | $78.41_{\pm 0.24}$ | $79.77_{\pm 1.35}$ |
| GCN | $81.71_{\pm 0.74}$ | $72.10_{\pm 0.58}$ | $79.80_{\pm 0.11}$ | $77.29_{\pm 3.59}$ |
| APPNP | $83.21_{\pm 0.23}$ | $71.78_{\pm 0.38}$ | $80.14_{\pm 0.22}$ | $80.89_{\pm 0.43}$ |
| GAT | $82.88_{\pm 0.44}$ | $71.03_{\pm 0.88}$ | $78.61_{\pm 0.38}$ | $78.61_{\pm 0.38}$ |
| JKNet | $81.10_{\pm 0.13}$ | $69.80_{\pm 0.36}$ | $78.10_{\pm 0.24}$ | $79.54_{\pm 0.21}$ |
| GCNII | $\mathbf{84.71}_{\pm 0.59}$ | $72.57_{\pm 0.74}$ | $79.96_{\pm 0.32}$ | $80.28_{\pm 1.68}$ |
| FAGCN | $83.18_{\pm 0.70}$ | $71.67_{\pm 1.01}$ | $80.08_{\pm 0.18}$ | $80.92_{\pm 0.98}$ |
| GPRGCN | $83.61_{\pm 0.31}$ | $72.59_{\pm 0.52}$ | $80.10_{\pm 0.29}$ | $80.94_{\pm 0.31}$ |
| Geom-GCN | $82.31_{\pm 1.13}$ | $72.44_{\pm 1.50}$ | $80.03_{\pm 0.91}$ | $79.92_{\pm 0.85}$ |
| H2GCN | $82.08_{\pm 0.60}$ | $70.70_{\pm 0.37}$ | $80.26_{\pm 0.22}$ | $80.02_{\pm 0.26}$ |
| FPA(ours) | $84.20_{\pm 0.34}$ | $72.67_{\pm 0.70}$ | $\mathbf{80.46}_{\pm 0.24}$ | $81.34_{\pm 0.29}$ |
| SFPA(ours) | $84.33_{\pm 0.32}$ | $\mathbf{72.91}_{\pm 0.53}$ | $80.25_{\pm 0.36}$ | $\mathbf{81.42}_{\pm 0.33}$ |

**Table 4: Comparison of mean classification accuracy (%) and standard deviation for GNNs equipped with FPA and SFPA. *None* denotes the unprocessed backbone models. The bold indicates the best result for each model on each dataset.**

| Model | | Dataset | | | |
|---|---|---|---|---|---|
| Backbone | Strategy | Cora | Penn94 | Chameleon | Squirrel |
| GCN | *None* | $81.71_{\pm 0.74}$ | $63.55_{\pm 1.30}$ | $59.82_{\pm 2.58}$ | $36.89_{\pm 1.34}$ |
| | FPA | $\mathbf{81.93}_{\pm 0.33}$ | $68.69_{\pm 1.07}$ | $\mathbf{62.73}_{\pm 1.82}$ | $40.80_{\pm 1.12}$ |
| | SFPA | $81.89_{\pm 0.69}$ | $\mathbf{69.38}_{\pm 0.89}$ | $62.43_{\pm 1.78}$ | $\mathbf{41.31}_{\pm 1.44}$ |
| APPNP | *None* | $83.21_{\pm 0.23}$ | $63.56_{\pm 3.51}$ | $52.57_{\pm 1.82}$ | $33.29_{\pm 1.72}$ |
| | FPA | $\mathbf{83.79}_{\pm 0.32}$ | $\mathbf{69.61}_{\pm 1.75}$ | $\mathbf{54.21}_{\pm 1.75}$ | $35.77_{\pm 1.20}$ |
| | SFPA | $83.49_{\pm 0.43}$ | $69.09_{\pm 1.24}$ | $53.95_{\pm 1.98}$ | $\mathbf{36.20}_{\pm 1.53}$ |
| FAGCN | *None* | $83.18_{\pm 0.70}$ | $69.03_{\pm 1.32}$ | $61.12_{\pm 1.95}$ | $40.88_{\pm 2.02}$ |
| | FPA | $\mathbf{83.53}_{\pm 0.46}$ | $\mathbf{70.62}_{\pm 1.61}$ | $\mathbf{61.74}_{\pm 1.56}$ | $47.06_{\pm 1.31}$ |
| | SFPA | $83.42_{\pm 0.48}$ | $69.67_{\pm 1.28}$ | $61.58_{\pm 2.15}$ | $\mathbf{47.87}_{\pm 1.16}$ |

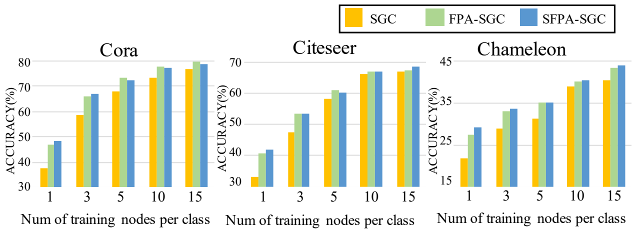


**Figure 5: Impact of the number of nodes per class.**

## 6.2 Performance Evaluation

**Heterophilic Datasets.** As shown in Table 2, the proposed models performs better than others on heterophilic datasets. In line with the results from previous studies, GNNs broadly received structural inductive bias using feature aggregating operation. However, their node embeddings are poorly performed on heterophilic graphs since their class distribution is inconsistent with homophilic graph-based inductive bias. By considering global classifier parameters in aggregation operation, the node representations obtained by the proposed models can receive task-related positive influences, improving the expressibility of nodes.

**Homophilic Datasets.** Table 3 shows that the proposed models demonstrate superior performance compared to the basic models for homophilic graph data on homophilic datasets. The results suggest that the proposed augmentation models may provide more accurate information for absorbing the features of different-class neighbors than traditional aggregations with positive weights.

**FPA and SFPA with Different Backbones.** Table 4 mainly shows the performance improvement of backbone models after being equipped with FPA and SFPA. It is worth noting that almost all GNN models directly profited by equipping the proposed global augmentation strategies, whether applied to graphs with higher or lower edge homophily. This is due to the fact that nodes aggregate the information of CA nodes which perform intra-cluster smoothing and inter-cluster sharpening on the feature space.

## 6.3 Model Analysis

**Limited labeled Training Data.** The label efficiency experiment is conducted to evaluate the impact of the number of training nodes on the performance, where SGC is chosen as the backbone model, and the training nodes per class are selected among $\{1, 3, 5, 10, 15\}$. As illustrated in Figure 5, the accuracy of the SGC model decreases with a reduction in the number of usable labels while the accuracy variances increase on each dataset. However, our proposed framework leverages the classifier parameter matrix to function both as a classifier and a feature propagation corrector using the label matrix, leading to a more stable outcome.

**Over-smoothing Analysis.** The experiment analyzes the ability of the proposed models to alleviate the over-smoothing problem, a drawback found in GCN and SGC models presented in [2]. As shown in Figure 6, the proposed implementation models consistently outperform the backbone models across all datasets and layers. Moreover, the performance of the proposed models for each dataset does not drop significantly as some baselines. The reason is that the messages of nodes are complemented by propagating the parameter features, and the false propagation coefficient between nodes is corrected.

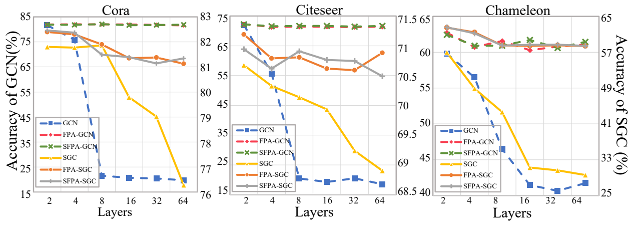


**Figure 6: Influence of model depth (number of layers).**

**Table 5: Time of model training and quantity of network parameters on Cora, where time is the total time of 500 epochs.**

| | SGC | | | GCNII | | |
|---|---|---|---|---|---|---|
| | *None* | FPA | SFPA | *None* | FPA | SFPA |
| Time (sec) | 1.13 | 1.42 | 1.41 | 2.7 | 3.38 | 2.95 |
| Space (byte) | 92,231 | 92,231 | 92,231 | 133,191 | 133,191 | 133,191 |

**Complexity Analysis.** The experiment analyzes the time and space complexity of the proposed framework, whose results are shown in Table 5. The complexities of the proposed augmentation scheme, FPA, and SFPA, are $O(N_L DC)$. Compared to the complexity of the simplest baseline SGC, i.e., $O(|E|D + ND^2)$, where $|E|$is the number of edges, the complexity of the proposed schemes is lower since the number of training nodes is small. Equipping with FPA and SFPA slightly increases the total times for run times since a few matrix operations are employed to compute the propagation weights.

## 7 CONCLUSION

This study explores the training scheme for network parameters in Graph Neural Networks (GNNs), an area that has been underemphasized in previous research. By visualizing node and parameter features, the study observes the distinguishability of parameter embeddings compared to node embeddings. Inspired by this finding, a full propagation training framework for GNNs is proposed, updating both node and parameter features through a unified feature propagation scheme. The universality of the proposed framework is demonstrated theoretically, showcasing its potential to enhance the efficiency and robustness of GNN architectures. The following research is to investigate the training scheme for GNN network parameters in unsupervised scenarios.

## 8 ACKNOWLEDGMENTS

This work was supported in part by the National Science Fund for Distinguished Young Scholarship of China (No. 62025602), in part by the National Natural Science Foundation of China (No. 61972442, 62102413, 62276187, 62272020, U1936210, U1936208, U22B2036, 11931915), in part by the Natural Science Foundation of Hebei Province of China under Grant F2020202040, in part by the Fok Ying-Tong Education Foundation China (No. 171105), in part by the Tencent Foundation and XPLORER PRIZE, and in part by the Fundamental Research Funds for the Central Universities.

## REFERENCES

[1] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 2021. Beyond Low-frequency Information in Graph Convolutional Networks. (2021), 3950–3957.

[2] Chen Cai and Yusu Wang. 2020. A Note on Over-Smoothing for Graph Neural Networks. *CoRR* abs/2006.13318 (2020). arXiv:2006.13318

[3] Junyu Chen, Qianqian Xu, Zhiyong Yang, Xiaochun Cao, and Qingming Huang. 2022. A Unified Framework against Topology and Class Imbalance. In *ACM International Conference on Multimedia*. 180–188.

[4] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and Deep Graph Convolutional Networks. In *ICML*, Vol. 119. PMLR, 1725–1735.

[5] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *ICLR*.

[6] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *ICML*, Doina Precup and Yee Whye Teh (Eds.), Vol. 70. PMLR, 1263–1272.

[7] Greg Hamerly and Charles Elkan. 2003. Learning the k in k-means. In *NIPS*, Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf (Eds.). MIT Press, 281–288.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. IEEE Computer Society, 770–778. https://doi.org/10.1109/CVPR.2016.90

[9] Yixuan He, Quan Gan, David Wipf, Gesine D. Reinert, Junchi Yan, and Mihai Cucuringu. 2022. GNNRank: Learning Global Rankings from Pairwise Comparisons via Directed Graph Neural Networks. In *ICML*. 8581–8612.

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

[11] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*. OpenReview.net.

[12] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser-Nam Lim. 2021. Large Scale Learning on Non-Homophilous Graphs: New Benchmarks and Strong Simple Methods. In *NeurIPS*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 20887–20902.

[13] Ke Ma, Qianqian Xu, Jinshan Zeng, Xiaochun Cao, and Qingming Huang. 2022. Poisoning Attack Against Estimating From Pairwise Comparisons. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 10 (2022), 6393–6408.

[14] Ke Ma, Qianqian Xu, Jinshan Zeng, Guorong Li, Xiaochun Cao, and Qingming Huang. 2023. A Tale of HodgeRank and Spectral Method: Target Attack Against Rank Aggregation is the Fixed Point of Adversarial Game. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 4 (2023), 4090–4108.

[15] Yao Ma, Xiaorui Liu, Tong Zhao, Yozen Liu, Jiliang Tang, and Neil Shah. 2021. A Unified View on Graph Neural Networks as Graph Signal Denoising. , 1202–1211 pages. https://doi.org/10.1145/3459637.3482225

[16] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-GCN: Geometric Graph Convolutional Networks. In *ICLR*. OpenReview.net.

[17] Trang Pham, Truyen Tran, Khanh Hoa Dam, and Svetha Venkatesh. 2017. Graph Classification via Deep Learning with Virtual Nodes. *CoRR* abs/1708.04357 (2017). arXiv:1708.04357

[18] Hassan Ramchoun, Mohammed Amine Janati Idrissi, Youssef Ghanou, and Mohamed Ettaouil. 2017. Multilayer Perceptron: Architecture Optimization and training with mixed activation functions. In *BDCA*, Mohamed Lazaar, Youness Tabii, Mohamed Chrayah, and Mohammed Al Achhab (Eds.). ACM, 71:1–71:6. https://doi.org/10.1145/3090354.3090427

[19] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2021. Multi-scale attributed node embedding. *Journal of Complex Networks* 9, 2 (2021), cnab014.

[20] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[21] Petar Velickovic. 2022. Message passing all the way up. *CoRR* abs/2202.11097 (2022). arXiv:2202.11097

[22] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*. OpenReview.net.

[23] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17, 4 (2007), 395–416.

[24] Shiping Wang, Zhihao Wu, Yuhong Chen, and Yong Chen. 2023. Beyond Graph Convolutional Network: An Interpretable Regularizer-Centered Optimization Framework. (2023), 4693–4701.

[25] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. In *ICML*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. PMLR, 6861–6871.

[26] Junjie Xu, Enyan Dai, Xiang Zhang, and Suhang Wang. 2022. HP-GMN: Graph Memory Networks for Heterophilous Graphs. In *ICDM*, Xingquan Zhu, Sanjay Ranka, My T. Thai, Takashi Washio, and Xindong Wu (Eds.). IEEE, 1263–1268. https://doi.org/10.1109/ICDM54844.2022.00165

[27] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *ICML*, Jennifer G. Dy and Andreas Krause (Eds.), Vol. 80. PMLR, 5449–5458.

[28] Liang Yang, Chuan Wang, Junhua Gu, Xiaochun Cao, and Bingxin Niu. 2021. Why Do Attributes Propagate in Graph Convolutional Neural Networks?. In *AAAI*. AAAI Press, 4590–4598.

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

[30] Muhan Zhang and Yixin Chen. 2018. Link Prediction Based on Graph Neural Networks. In *NeurIPS*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 5171–5181.

[31] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An End-to-End Deep Learning Architecture for Graph Classification. In *AAAI*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). AAAI Press, 4438–4445.

[32] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. In *NeurIPS*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.).

[33] Meiqi Zhu, Xiao Wang, Chuan Shi, Houye Ji, and Peng Cui. 2021. Interpreting and Unifying Graph Neural Networks with An Optimization Framework. In *WWW*, Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia (Eds.). ACM / IW3C2, 1215–1226. https://doi.org/10.1145/3442381.3449953