

Meta-Weight Graph Neural Network: Push the Limits Beyond Global Homophily

Xiaojun Ma
mxj@pku.edu.cn
Key Lab. of Machine Perception
(MoE), School of AI, Peking
University
Beijing, China

Qin Chen
chenqink@pku.edu.cn
Key Lab. of Machine Perception
(MoE), School of AI, Peking
University
Beijing, China

Yuanyi Ren
celina@pku.edu.cn
Key Lab. of Machine Perception
(MoE), School of AI, Peking
University
Beijing, China

Guojie Song*
gjsong@pku.edu.cn
Key Lab. of Machine Perception
(MoE), School of AI, Peking
University
Beijing, China

Liang Wang
liangbo.wl@alibaba-inc.com
Alibaba Inc
Beijing, China

ABSTRACT

Graph Neural Networks (GNNs) show strong expressive power on graph data mining, by aggregating information from neighbors and using the integrated representation in the downstream tasks. The same aggregation methods and parameters for each node in a graph are used to enable the GNNs to utilize the homophily relational data. However, not all graphs are homophilic, even in the same graph, the distributions may vary significantly. Using the same convolution over all nodes may lead to the ignorance of various graph patterns. Furthermore, many existing GNNs integrate node features and structure identically, which ignores the distributions of nodes and further limits the expressive power of GNNs. To solve these problems, we propose Meta Weight Graph Neural Network (MWGNN) to adaptively construct graph convolution layers for different nodes. First, we model the Node Local Distribution (NLD) from node feature, topological structure and positional identity aspects with the Meta-Weight. Then, based on the Meta-Weight, we generate the adaptive graph convolutions to perform a node-specific weighted aggregation and boost the node representations. Finally, we design extensive experiments on real-world and synthetic benchmarks to evaluate the effectiveness of MWGNN. These experiments show the excellent expressive power of MWGNN in dealing with graph data with various distributions.

CCS CONCEPTS

• **Mathematics of computing** → **Graph algorithms**; • **Computing methodologies** → *Neural networks*; Supervised learning.

*Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

© 2022 Association for Computing Machinery.

ACM ISBN 978-1-4503-9096-5/22/04...\$15.00

<https://doi.org/10.1145/3485447.3512100>

KEYWORDS

Graph Neural Networks, Representation Power, Graph Theory

ACM Reference Format:

Xiaojun Ma, Qin Chen, Yuanyi Ren, Guojie Song, and Liang Wang. 2022. Meta-Weight Graph Neural Network: Push the Limits Beyond Global Homophily. In *Proceedings of the ACM Web Conference 2022 (WWW '22)*, April 25–29, 2022, Virtual Event, Lyon, France. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3485447.3512100>

1 INTRODUCTION

As a powerful approach to extracting and learning information from relational data, Graph Neural Networks (GNNs) have flourished in many applications, including molecules [18, 31], social networks [6], biological interactions [33], and more. Among various techniques [3, 9, 14, 23], Graph Convolutional Network (GCN) stands out as a powerful and efficient model. Since then, more variants of GNNs, such as GAT [24], SGCN [27], GraphSAGE [10], have been proposed to learn more powerful representations. These methods embrace the assumption of homophily on graphs, which assumes that connected nodes tend to have the same labels. Under such an assumption, the propagation and aggregation of information within graph neighborhoods are efficient in graph data mining.

Recently, some research papers [7, 20, 25, 34, 35] propose to adapt the Graph Convolution to extend GNNs' expressive power beyond the limit of homophily assumption, because there do exist real-world graphs with heterophily settings, where linked nodes are more likely to have different labels. These methods improve the expressive power of GNNs by changing the definition of neighborhood in graph convolutions. For example, H2GCN [35] extend the neighborhood to higher-order neighbors and AM-GCN [25] constructs another neighborhood considering feature similarity. Changing the definition of neighborhood do help GNNs to capture the heterophily in graphs.

However, homophily and heterophily is only a simple measurement of the NLD, because they only consider the labels of the graphs. In real-world graphs, topological structure, node feature and positional identity play more critical role than labels, especially

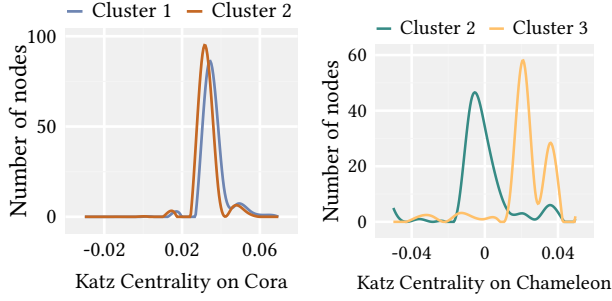


Figure 1: Katz centrality distribution of clustered Cora (in class 1) and Chameleon (in class 4). The nodes are clustered and then the Katz Centrality distribution is plotted for nodes of the same label but belonging in two different clusters.

for graphs with few labels. So the models mentioned above lack the ability to model NLD. We visualize the Katz centrality [12], a measurement of the topological structure of nodes, in Figure 1 to show two situations. We first separate the graph with METIS [11] and show nodes of one label in Cora and Chameleon. In Cora, the Katz centrality reveals high consistency in different parts, while that of Chameleon shows an obvious distinction between different parts.

In addition to the difference in topological structure distributions, there are also variances in node feature distributions. Besides, the correlation between topological structure and node feature distributions is not consistent. Therefore, using one graph convolution to integrate the topological structure and node feature information leads to ignorance of such complex correlation.

In conclusion, there are two challenges limiting the expressive power of GNNs: (1) the complexity of Node Local Distributions (2) the inconsistency correlation between node feature and topological structure. To overcome these challenges, we propose *Meta-Weight Graph Neural Network*. For the first challenge, We model the NLD in topological structure, node feature, positional identity fields with Meta-Weight. In detail, the three types of NLD is captured using Gated Recurrent Unit (GRU) and Multi-Layer Perception (MLP) separately and then combined using an attention mechanism. For the second challenge, based on Meta-Weight, we adaptively conduct graph convolution with two aggregation weights and three channels. The proposed two aggregation weights decouple the correlation between node feature and topological structure. To further model the complex correlation, we boost the node representations with the entangled channel, node feature channel and topological structure channel, respectively. We conduct experiments on semi-supervised node classification. The excellent performance of MWGNN is demonstrated empirically on both real-world and synthetic datasets. Our major contributions are summarized as follows:

- We demonstrate the insufficient modeling of NLD of existing GNNs and propose a novel architecture MWGNN which successfully adapt the convolution learning for different distributions considering topological structure, node feature, and positional identity over one Graph.

- We propose the Meta-Weight mechanism to describe the complicated NLD and the adaptive convolution based on Meta-Weight to boost node embeddings with decoupled aggregation weights and independent convolution channels.
- We conduct experiments on real-world and synthetic datasets to demonstrate the superior performance of MWGNN. Especially, MWGNN gains an accuracy improvement of over 20% on graphs with the complex NLD.

2 PRELIMINARY

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected, unweighted graph with node set \mathcal{V} and edge set \mathcal{E} . Let $|\mathcal{V}| = N$. We use $A \in \{0, 1\}^{N \times N}$ for the adjacency matrix, $X \in \mathbb{R}^{N \times F}$ for the node feature matrix, and $\mathbf{y} \in \mathbb{R}^N$ for the node label matrix. Let \mathcal{N}_i denote the neighborhood surrounding node v_i , and $\mathcal{N}_{i,k} = \{v_j | d(v_i, v_j) \leq k\}$ denote v_i 's neighbors within k hops.

2.1 Graph Neural Networks

Most Graph Neural Networks formulates their propagation mechanisms by two phases, the aggregation phase and the transformation phase. The propagation procedure can be summarized as

$$\mathbf{H}_i^{(l+1)} = \text{TRANS} \left(\text{AGG} \left(\mathbf{H}_i^{(l)}, \left\{ \mathbf{H}_j^{(l)} : v_j \in \mathcal{N}_i \right\} \right) \right), \quad (1)$$

where $\mathbf{H}^{(l)} \in \mathbb{R}^{N \times d^{(l)}}$ stands for the embedding of the l -th layer and $\mathbf{H}^{(0)} = X$, $d^{(l)}$ is the dimension of l -th layer representations. $\text{AGG}(\cdot)$ denotes the function aggregating $\mathbf{H}^{(k)}$, and $\text{TRANS}(\cdot)$ is a layer-wise transformation function including a weight matrix $\mathbf{W}^{(l)}$ and the non-linear activation functions (e.g. ReLU).

2.2 Global and Local Homophily

Here we define global and local homophily ratio to estimate the homogeneity level of a graph.

Definition 2.1 (Global Edge Homophily). We define Global Edge Homophily ratio [35] h as a measure of the graph homophily level:

$$h = \frac{|\{(v_i, v_j) : (v_i, v_j) \in \mathcal{E} \wedge \mathbf{y}_i = \mathbf{y}_j\}|}{|\mathcal{E}|}, \quad (2)$$

h represents the percentage of edges connecting nodes of the same label in the edge set \mathcal{E} , Graphs with strong homophily may have a high global edge homophily ratio up to 1, while those with low homophily embrace a low global edge homophily ratio down to 0.

Definition 2.2 (Local Edge Homophily). For node v_i in a graph, we define the Local Edge Homophily ratio h_i as a measure of the local homophily level surrounding node v_i :

$$h_i = \frac{|\{(v_i, v_j) : v_j \in \mathcal{N}_i \wedge \mathbf{y}_i = \mathbf{y}_j\}|}{|\mathcal{N}_i|}, \quad (3)$$

h_i directly represents the edge homophily in the neighborhood \mathcal{N}_i surrounding node v_i .

3 META-WEIGHT GRAPH NEURAL NETWORK

Overview. In this section, we introduce the proposed method MWGNN. The MWGNN framework consists of two stages: (a) modeling Node Local Distribution by Meta-Weight and (b) adaptive

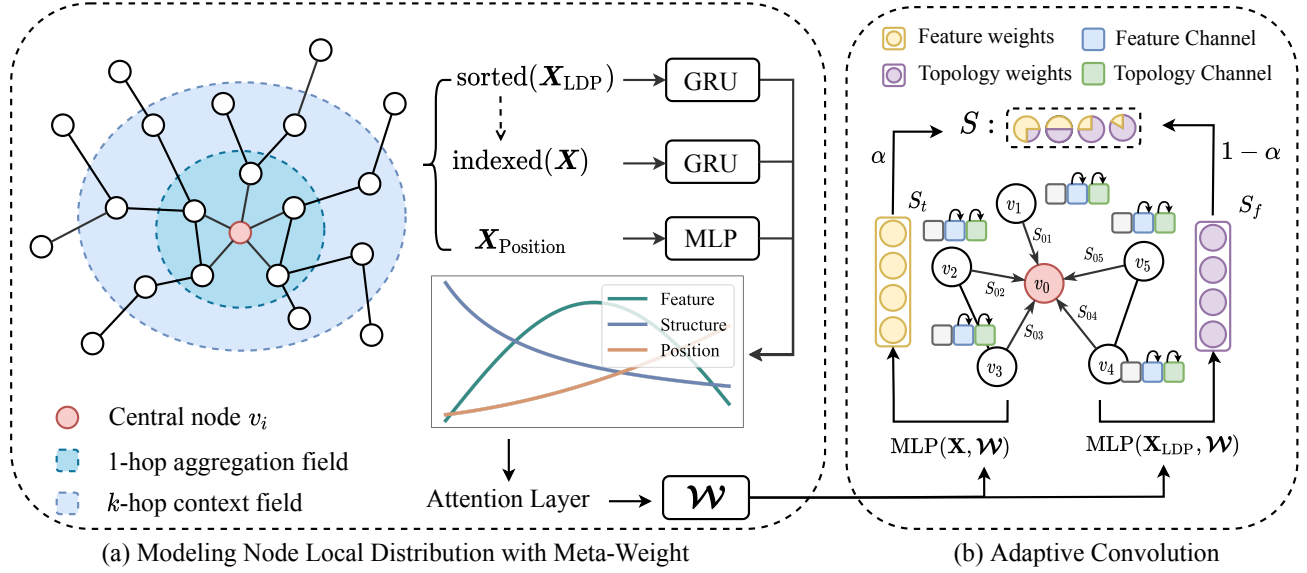


Figure 2: The framework of MWGNN. (a) Generate the Meta-Weight considering k -hop context field for central nodes. First we learn three local distributions in topological structure, node feature, and positional identity fields and integrate them with an attention layer. (b) Based on the Meta-Weight, we propose the Adaptive Convolution. By generating S_t, S_f and adaptively fusing them with a hyper-parameter α , the Adaptive Convolution aggregates the neighbors. Then two additional Independent Convolution Channels are proposed to boost the node representations efficiently.

convolution. The visualization of the framework is shown in Figure 2. First, we generate the Meta-Weight to model the NLD considering topological structure, node feature, and positional identity distributions separately. Then we integrate them via an attention mechanism, as shown in (a). Next, the key contributions in (b) is the Adaptive Convolution consisting of the Decoupled Aggregation Weights and Independent Convolution Channels for node feature and topological structure.

3.1 Modeling Node Local Distribution with Meta-Weight

In this stage, we aim to learn a specific *key* to guide the graph convolution adaptively. As discussed in section 1, the complex Node Local Distribution hinders the expressive power of GNNs. For the node classification task, GNNs essentially map the combination of node features and topological structure from neighborhood to node labels. Using the same convolution over all the nodes and the predefined neighborhood, most existing GNNs can only project a fixed combination to node labels. Therefore, these GNNs achieve satisfactory results on graphs with simple NLD (e.g. homophily) while failing to generalize to graphs with complex NLD (e.g. heterophily).

To push the limit on graphs with complex NLD and conduct adaptive convolution on nodes, first we need to answer the question: *What exactly NLD is and how to explicitly model it?* Node Local Distribution is a complex and general concept, in this paper, when discussing NLD, we refer to the node patterns in topological structure, node feature, and positional identity fields. Topological structure and node feature are widely used in the learning of node representations.

However, only using topological structure and node feature limits the expressive power of GNNs, because some nodes can not be recognized in the computational graphs [30]. IDGNN proposes a solution to improve the expressive power of GNNs than the 1-Weisfeiler-Lehman (1-WL) test [26], by inductively injecting nodes' identities during message passing. It empowers GNNs with the ability to count cycles, differentiate random d -regular graphs, etc. Nevertheless, the task of modeling NLD is much more complex. Thus we need a stronger identification. So we introduce the positional identity to model the NLD along with topological structure and node feature. In general, we learn three representation matrices $\mathcal{D}_t, \mathcal{D}_f, \mathcal{D}_p \in \mathbb{R}^{N \times d_{Meta}}$ to formulate these three node patterns.

Finally, the three kinds of node patterns may contribute unequally to NLD for different nodes, so we adopt an attention mechanism to combine them automatically.

3.1.1 Node Patterns in Topological Structure, Node Feature, and Positional Identity Fields. We elaborate on modeling the local node patterns in topological structure, node feature, and positional identity fields.

Topological Structure Field. Topology information is an essential part of graph data. Most GNNs implicitly capture topology information by aggregating the representations of neighbor nodes, and combining feature information with topology information. Other methods, Struct2Vec [21] for example, explicitly learn the topology information and express it with vectors. Usually, they utilize computationally expensive mechanisms like Random Walk to explicitly extract the topology information. In this paper, we propose a simple but effective method using the degree distribution to capture

node patterns in the topological structure field because the degree distribution can describe the topological structure [2].

For a node v_i , the sub-graph induced by $\mathcal{N}_{i,k}$ contains the related topology information. The degree distribution of $\mathcal{N}_{i,k}$ describes the local topological structure. To make this distribution graph-invariant, we sort the degrees by their value. It is noted that we use a statistical description, the Local Degree Profile (LDP) [4] to provide multi-angle information:

$$X_{\text{LDP},i} = [d_i, \text{MIN}(\text{DN}_i), \text{MAX}(\text{DN}_i), \\ \text{MEAN}(\text{DN}_i), \text{STD}(\text{DN}_i)],$$

where $X_{\text{LDP},i}$ is the LDP for node v_i , d_i is the degree of v_i and $\text{DN}_i = \{d_j | v_j \in \mathcal{N}_i\}$.

As a sequential data, the sorted LDP can be well studied by Gated Recurrent Unit [8] as:

$$\begin{aligned} \mathbf{u}^{(t)} &= \sigma(\mathbf{W}_u [\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}] + \mathbf{b}_u) \\ \mathbf{r}^{(t)} &= \sigma(\mathbf{W}_r [\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}] + \mathbf{b}_r) \\ \hat{\mathbf{h}}^{(t)} &= \tanh(\mathbf{W}_h [\mathbf{r}^{(t)} \odot \mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}] + \mathbf{b}_h) \\ \mathbf{h}^{(t)} &= (1 - \mathbf{u}^{(t)}) \odot \mathbf{h}^{(t-1)} + \mathbf{u}^{(t)} \odot \hat{\mathbf{h}}^{(t)} \end{aligned} \quad (4)$$

where $\mathbf{x}^{(t)} \in \mathbb{R}^5$ is the t -th LDP vector of the sorted degree sequence of $\mathcal{N}_{i,k}$, \odot denotes element-wise multiplication, and $\mathbf{h}^{(t)}$ denotes the hidden states of t -th step of GRU. The output $\mathcal{D}_t = \mathbf{h}^{(T)}$ of GRU is the learnt topological structure distribution, where T is the layer number of GRU.

Node Feature Field. Usually, patterns in node feature field are captured by directly applying neural networks on the input X . On the one hand, this explicit estimation of node feature patterns enables any off-the-shelf neural networks. On the other hand, thanks to the flourish of neural networks, we can accommodate different distributions of node features.

In learning the distributions of topological structure, we sort the degree sequence by the value of the degrees, which enables the estimator to perform the graph-agnostic calculation. To maintain the node-level correspondence between feature and degree sequence, we sort the node feature X along the nodes' dimension to keep them in the same order. For node v_i , the sorted node feature sequence is:

$$X_{\text{Feature}} = \text{SORT}(\{X_j | v_j \in \mathcal{N}_{i,k}\}),$$

where the sorting function $\text{SORT}(\cdot)$ denotes the order by nodes' degrees. We consider two types of neural work to model Feature Pattern:

- GRU. The mechanism to learn the distribution of node feature pattern resembles the one of topological structure pattern. Taking X_{Feature} as input, another GRU as Equation 4 is applied. In this way, GRU reckons the distribution of node feature pattern, and the output \mathcal{D}_f of GRU is the learnt node feature distribution.
- Average. The average operation can be viewed as the simplest neural networks without parameters. This is a graph-agnostic method without the need to sort the node feature. Simply taking the average of $\{X_j | v_j \in \mathcal{N}_{i,k}\}$, we have the summary of the feature distribution. In the experiment, the

most results are carried by MWGNN with the average operation, which illustrates that the specific implementation is not essential to the ability of meta-weight \mathcal{W} . On the contrary, the general design of the local distribution generator exerts a more significant impact on the results.

Positional Identity Field. The position embedding is widely used in Transformer to enhance models' expressive power. In fact, position embedding has been used in graphs in another form [32], by sampling sets of anchor nodes and computing the distance of a given target node to each anchor-set. However, explicitly learning a non-linear distance-weighted aggregation scheme over the anchor-sets [32] is computationally intensive and requires much space for the storage of anchor nodes. Therefore, we use the distance of the shortest path (SPD) between any two nodes as the position embedding [29], which helps the model accurately capture the spatial dependency in a graph. To be concrete, the position embedding of node v_i is:

$$X_{\text{Position},i} = (\phi(v_i, v_1), \phi(v_i, v_2), \dots, \phi(v_i, v_n)), \quad (5)$$

here $\phi(v_i, v_j)$ denotes the SPD between nodes v_i and v_j if the two nodes are connected. If not, we set the output of ϕ to be a special value, i.e., -1. Then, a neural network $\Phi(\cdot)$ is applied on the position embedding X_{Position} to model the distributions of positional identity pattern. Finally, we have $\mathcal{D}_p = \Phi(X_{\text{Position}})$ as the positional identity distribution.

3.1.2 Integration of Three Distributions. The above-mentioned process models three specific local distributions: topological structure distribution, \mathcal{D}_t , node feature distribution \mathcal{D}_f , and positional identity distribution \mathcal{D}_p . The overall local distribution of nodes could be correlated with one of them or their combinations to different extents. Thus we use an attention mechanism to learn the corresponding combination weights $(\mathbf{a}_t, \mathbf{a}_f, \mathbf{a}_p) \in \mathbb{R}^{N \times 3}$ as the attention values of N nodes with the distributions $\mathcal{D}_t, \mathcal{D}_f, \mathcal{D}_p$, respectively. For node v_i , the corresponding topology structure distribution is the i -th row of \mathcal{D}_t . We apply a nonlinear transformation to \mathcal{D}_t , and then use one shared attention vector $\mathbf{q} \in \mathbb{R}^{1 \times d_q}$ to compute the attention value for node v_i as the following:

$$\omega_t^i = \mathbf{q} \cdot \tanh(\mathbf{W}_a \cdot (\mathcal{D}_{t,i})^T + \mathbf{b}), \quad (6)$$

where $\tanh(\cdot)$ is the activation function, $\mathbf{W}_a \in \mathbb{R}^{d_q \times d_{\text{Meta}}}$ is the parameter matrix and $\mathbf{b} \in \mathbb{R}^{d_q \times 1}$ is the bias vector. Similarly, the attention values for node feature and positional identity distributions $\mathcal{D}_f, \mathcal{D}_p$ are obtained by the same procedure. We then normalize the attention values with the softmax function to get the final weight $a_t^i = \frac{\exp(\omega_t^i)}{\exp(\omega_t^i) + \exp(\omega_f^i) + \exp(\omega_p^i)}$. Large a_t^i implies the topological structure distribution dominates the NLD. Similarly, we compute a_f^i and a_p^i . Now for all the N nodes, we have the learned weights $\mathbf{a}_t = [a_t^i], \mathbf{a}_f = [a_f^i], \mathbf{a}_p = [a_p^i] \in \mathbb{R}^{N \times 1}$, and denote $\mathbf{a}_t = \text{diag}(\mathbf{a}_t), \mathbf{a}_f = \text{diag}(\mathbf{a}_f)$ and $\mathbf{a}_p = \text{diag}(\mathbf{a}_p)$. Then we integrate these three distributions to obtain the representation of NLD \mathcal{W} :

$$\mathcal{W} = \mathbf{a}_f \odot \mathcal{D}_f + \mathbf{a}_t \odot \mathcal{D}_t + \mathbf{a}_p \odot \mathcal{D}_p. \quad (7)$$

3.2 Adaptive Convolution

In the second stage, we elaborate the concrete algorithm based on the generated meta-weight \mathcal{W} . To adapt the graph convolution according to the information contained within the NLD, we propose the Decoupled Aggregation Weights and Independent Convolution Channels for node feature and topological structure. On the one hand, we decouple the neighbor embedding aggregation weights based on \mathcal{W} into S_f and S_t and balance them with a hyper-parameter α . The design ensures that the aggregation take the most correlated information into account. On the other hand, two additional Independent Convolution Channels for original topology and feature are introduced to boost the node representations.

3.2.1 Decouple Topology and Feature in Aggregation. Recalling the discussion at the beginning of subsection 3.1, using the same convolution over all the nodes and the pre-defined neighborhood, most existing GNNs can only project a fixed combination of node features and topological structure from neighborhood to node labels. Therefore, these GNNs achieve satisfactory results on graphs with simple NLD. However, when the local distribution varies, the common $\text{MEAN}(\cdot)$ or normalized aggregation can not recognize the difference and loss the distinguishment among nodes. Therefore, we propose decoupling topology and feature in aggregation to adaptively weigh the correlation between neighbor nodes and ego nodes from the local distribution concept. The following is the details of our decouple mechanism:

$$\mathbf{H}^{(l+1)} = \sigma \left(\hat{P} \mathbf{H}^{(l)} \mathbf{W}^{(l)} \right), \hat{P} = \mathbf{A} \odot \mathbf{S} \quad (8)$$

$$\mathbf{S} = \alpha \cdot \mathbf{S}_f + (1 - \alpha) \cdot \mathbf{S}_t \quad (9)$$

$$\mathbf{S}_f = \Psi_f(\mathcal{W}, \mathbf{X}), \mathbf{S}_t = \Psi_t(\mathcal{W}, \mathbf{X}_{\text{LDP}}), \quad (10)$$

where $\mathbf{H}^{(l)} \in \mathbb{R}^{N \times d_l}$ denotes the hidden state of the l -th layer, $\mathbf{W}^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ denotes the parameters of the l -th layer, σ is the activation function, \mathbf{S} is the integrated weight of aggregation, and $\mathbf{S}_f, \mathbf{S}_t$ are decoupled weights generated by two MLPs Ψ_f, Ψ_t , respectively. α is a hyper-parameter to balance the Ψ_f and Ψ_t . Equipped with the external \mathbf{X} and \mathbf{X}_{LDP} , the decouple of topology and feature in aggregation empowers the graph convolution to distinguish the different dependence on the corresponding factors and adjust itself to achieve the best performance.

3.2.2 Independent Convolution Channels for Topology and Feature. GNNs learn node representations via alternately performing the aggregation and the transformation. In addition to the integrated information, the original node patterns in the feature and topological structure are essential in graph data mining. GNNs lose their advantages when the representations are over-smoothed [17] because the useful original node patterns are smoothed. Recently, to alleviate the problem, some research work [7, 15, 19, 27] proposes that separating the procedure of aggregation and transformation. APPNP [15] first generates predictions for each node based on its own features and then propagates them via a personalized PageRank scheme to generate the final predictions. GPR-GNN [7] first extracts hidden states for each node and then uses Generalized PageRank to propagate them. However, the topological information is still entangled with the features even after separating the projection and propagation. Therefore, we propose two additional Independent Convolution Channels for the topology and feature

information so that the model can maintain the original signals. The detailed computation is:

$$\mathbf{H}^{(l+1)} = \sigma \left((1 - \lambda_1 - \lambda_2) \hat{P} \mathbf{H}^{(l)} + \lambda_1 \mathbf{H}_f^{(0)} + \lambda_2 \mathbf{H}_t^{(0)} \right) \cdot \left((1 - \beta) \cdot \mathbf{I}_n + \beta \cdot \mathbf{W}^{(l)} \right), \quad (11)$$

where $\mathbf{I}_n \in \mathbb{R}^{d_l \times d_{l+1}}$ with with ones as diagonal elements and $\lambda_1, \lambda_2, \beta$ are hyper-parameters. $\mathbf{H}_f^{(0)}$ is the representation of initial node features and $\mathbf{H}_t^{(0)}$ is the representation of initial node topology embedding.

Usually, in the previous GNNs like [16], the $\mathbf{H}_f^{(0)}$ is the original node feature matrix \mathbf{X} . In our implementation, we apply **fully-connected neural networks on the original node feature matrix \mathbf{X} and the adjacency matrix \mathbf{A}** respectively to obtain the lower-dimensional initial representations for $\mathbf{H}_f^{(0)}$ and $\mathbf{H}_t^{(0)}$, so that when the feature dimension F and the N are large, we can guarantee an efficient calculation.

3.3 Complexity

Here we analyse the time complexity of training MWGNN. Let $L_d = \max(|N_{1,k}|, \dots, |N_{N,k}|)$ denote the length of node degree sequence length for structure pattern. Because $d_{\text{LDP}} = 5$, we leave out this term. In subsection 3.1, the GRU costs $O(L_d(d_{\text{Meta}}^2 + d_{\text{Meta}}))$. The MLP generating position embeddings costs $O(N^2 d_{\text{Meta}})$ and we can reduce it to $O(|\mathcal{E}| d_{\text{Meta}})$ with an efficient GPU-based implementation using sparse-dense matrix multiplications. Next, the integration of three distributions costs $O(N d_{\text{Meta}} d_q^2)$. In the implementation, we set all the dimension of all hidden states as d_{hidden} . The computation of Equation 10 costs $O(N(F + d_{\text{Meta}}))$ as $F > \text{LDP}$. The computation for $\mathbf{H}_f^{(0)}$ and $\mathbf{H}_t^{(0)}$ costs $O(d_{\text{hidden}}(FN + |\mathcal{E}|))$. The overall time complexity of MWGNN is

$$O \left(L_d d_{\text{hidden}}^2 + N d_{\text{hidden}}^3 + N F d_{\text{hidden}} + |\mathcal{E}|(F + d_{\text{hidden}}) \right),$$

which matches the time complexity of other GNNs.

4 DEEP ANALYSIS OF MWGNN

4.1 How Node Local Distribution Influence the Expressive Power of GNNs

The discussion and empirical results above illustrate the importance of modeling NLD. Recalling subsection 3.1, local edge homophily can be a relatively plain measurement for NLD. Therefore, without loss of generality, we take the local edge homophily and GCN [14] as instances. We set P as the random variable for local edge homophily with its distribution as \mathcal{D}_P . Thus, the variance of P exhibits how the NLD differs throughout the graph. The larger the variance of P is, the more complex the Local Distribution Pattern will be, and vice versa. To prevent trivial discussion, we make a few assumptions¹ to simplify our analysis without loss of generality. We derive a learning guarantee considering the variance of P as follows.

THEOREM 4.1. Consider $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \{\mathcal{F}_c, c \in \{0, 1\}\}, \{\mathcal{D}_P, P \sim \mathcal{D}_P\}, k\}$, which follows assumptions in Appendix A. For any node

¹The detailed assumptions can be found in Appendix B

$v_i \in V$, the expectation of its pre-activation output of 1-layer GCN model is as follows:

$$\mathbb{E}[\mathbf{h}_i] = \mathbf{W} \left(\frac{1}{k+1} \mu(\mathcal{F}_{y_i}) + \frac{k}{k+1} \mathbb{E}_{P \sim \mathcal{D}_P, c \sim B(y_i, p), \mathbf{x}_j \sim \mathcal{F}_c}[\mathbf{x}_j] \right).$$

For any $t > 0$, the probability that the Euclidean distance between the observation \mathbf{h}_i and its expectation is larger than t is bounded as follows:

$$\mathbb{P}(\|\mathbf{h}_i - \mathbb{E}[\mathbf{h}_i]\|_2 \geq t) \leq 2d \exp \left(- \frac{((k+1)t_2/\rho(\mathbf{W}) + \sqrt{d}C_x + \sqrt{d}C_\mu)^2}{2kd\sigma^2 + 4\sqrt{d}C_x((k+1)t_2/\rho(\mathbf{W}) + \sqrt{d}C_x + \sqrt{d}C_\mu)/3} \right),$$

where $\sigma^2 = 4kC_\mu^2 \text{Var}[P] + kC_\tau$.

From Theorem 4.1 we demonstrate that the Euclidean distance between the output embedding of a node and its expectation is small when the variance of P is relatively small. However, as the complexity of LDP increases, the upper bound of the learning guarantee will rapidly grow, which indicates that the traditional learning algorithm is no longer promising under this circumstance. Therefore, it is necessary to design an adaptive convolution mechanism to adjust the convolution operator based on nodes' various distribution patterns.

4.2 Connection to existing GNNs

MWGNN on identical NLD degenerates into GNNs with three channels. When learning graphs with identical NLD, the \mathbf{W} can not help to distinguish the nodes from the three distributions. However, we can still learn the adaptive weights by Ψ_f and Ψ_t with $\mathbf{X}, \mathbf{X}_{\text{LDP}}$. Moreover, if we remove the independent convolution channels, MWGNN degenerates to advanced GAT with two types of attention mechanisms.

Innovation design of MWGNN. The two stages of MWGNN could be related to two types of methods. First, the form of Distribution-based Meta-Weight is like a kind of attention for aggregation. Unlike GAT measuring the similarity between nodes embeddings along the edges, we consider the local node distributions to generate the weights \mathbf{S} from two aspects. The meta-weights give a description of a sub-graph around central nodes, and the pair-wise correlation is implicitly contained in the \mathbf{S} . In addition, the design of Independent Convolution Channels is related to the residual layer. [5, 16] also introduced feature residual layers into GNNs. In addition to the residual layer of features, we also add a residual layer of topology. In this way, the final output of MWGNN contains three channels of representations, and the topology information is both explicitly and implicitly embedded into the representations.

5 EXPERIMENT

5.1 Datasets

The proposed MWGNN is evaluated on nine real-world datasets and two types of synthetic datasets.

5.1.1 Real-world datasets. The detailed information is in Table 3. We use datasets considering both homophily and heterophily. Cora, Citeseer, and Pubmed [28] are widely adopted citation datasets with strong edge homophily; In contrast, Texas and Cornell [20] are heterophily datasets; The situation of Chameleon and Squirrel

[22] are rather complex, with both homophily and heterophily combined.

5.1.2 Synthetic datasets. For synthetic benchmarks, we randomly generate graphs as below, referring to [1] and [25]: (1) Labels: we randomly assign C classes of labels to N nodes. (2) Node features: for the nodes with the same label, we use one Gaussian distribution to generate d -dimension node features. The Gaussian distributions for the C classes of nodes have the same co-variance matrix, but the mean values of these C Gaussian distributions are distinguishable. (3) Edges: the probability of building edges follows Bernoulli distributions controlled by $\mathbf{B} \in \mathbb{R}^{C \times C}$. In particular, the probability of building an edge between node v_i and v_j follows the Bernoulli(\mathbf{B}_{y_i, y_j}), where y_i and y_j are the node labels of v_i and v_j .

In addition, we further generate graphs combining different distributions (i.e. various Local Edge Homophily distributions) to demonstrate the situation where both homophilic and heterophilic data are mixed and tangled. Below are the details: (1) Generate two graphs $G_1(\mathcal{V}_1, \mathcal{E}_1)$ and $G_2(\mathcal{V}_2, \mathcal{E}_2)$ controlled by \mathbf{B}^1 and \mathbf{B}^2 respectively. In details, we set the value of \mathbf{B}_{ij}^1 high when $y_i = y_j$ and low when $y_i \neq y_j$ to build a graph with high homophily. Likewise, we set the value of \mathbf{B}_{ij}^2 low when $y_i = y_j$ and high when $y_i \neq y_j$ to build a graph with low homophily. (2) Combine G_1 and G_2 by randomly assign edges between nodes in G_1 and G_2 with a probability of p .

We generate three combined datasets: C.Homo (short for Combined graphs with Homophily), C.Mixed (short for Combined graphs with mixed Homophily and Heterophily), and C.Heter. Detailed information of synthetic datasets is in Table 4.

5.2 Settings

We evaluate MWGNN on the semi-supervised node classification task compared with state-of-the-art methods. For citation datasets (Cora, Citeseer, and Pubmed), we use the public split recommended by [28], fixed 20 nodes per class for training, 500 nodes for validation, and 1000 nodes for the test. For web page networks (Texas, and Cornell), we adopt the public splits by [20], with an average train/val/test split ratio of 48%/32%/20%. For Wikipedia networks (Chameleon and Squirrel) we use the public splits provided by [22], with an average train/val/test split ratio of 48%/32%/20%.

We use the Adam Stochastic Gradient Descent optimizer [13] with a learning rate $\eta \in \{10^{-2}, 10^{-3}, 10^{-4}\}$, a weight decay of 5×10^{-4} , and a maximum of 200 epochs with early stopping to train all the models. The number of hidden layers is set to 2, and the dimensions of hidden representations are set to 128 for fairness. For GAT-based models, the number of heads is set to 4.

5.3 Evaluation on Real-world Benchmarks

We compare the performance of MWGNN to the state-of-the-art methods³ in Table 1. Compared with all baselines, the proposed MWGNN generally achieves or matches the best performance on all datasets. Especially, MWGNN achieves an improvement of over

²[20] claims that the ratios are 60%/20%/20%, which is different from the actual data splits shared on GitHub.

³Note that on Chameleon and Squirrel, we reuse the results H2GCN reports, as they use public splits by [20]. The results of GPR-GNN and CPGNN are different from their reports because they use their own splits rather than the public splits.

Table 1: The summary of mean and standard deviation of accuracy over all runs. The best results for each dataset is highlighted in gray. "-" stands for Out-Of-Memory.

	Cora	Citeseer	Pubmed	Chameleon	Squirrel	Texas	Cornell
MLP	60.02 ± 0.75	53.36 ± 1.40	63.40 ± 5.03	48.50 ± 2.49	35.38 ± 1.66	75.95 ± 5.06	77.13 ± 5.32
GCN	80.50 ± 0.50	70.80 ± 0.50	79.00 ± 0.30	38.22 ± 2.67	27.12 ± 1.45	58.05 ± 4.81	56.87 ± 5.29
GAT	83.00 ± 0.70	72.50 ± 0.70	79.00 ± 0.30	43.07 ± 2.31	31.70 ± 1.85	57.38 ± 4.95	54.95 ± 5.63
GPR-GNN	83.69 ± 0.47	71.51 ± 0.29	79.77 ± 0.27	49.56 ± 1.71	37.21 ± 1.15	80.81 ± 2.55	78.38 ± 4.01
CPGNN-MLP-1	79.50 ± 0.38	71.76 ± 0.22	77.45 ± 0.24	49.25 ± 2.83	33.17 ± 1.87	80.00 ± 4.22	80.13 ± 6.47
CPGNN-MLP-2	78.21 ± 0.93	71.99 ± 0.39	78.26 ± 0.33	51.24 ± 2.43	28.86 ± 1.78	79.86 ± 4.64	79.05 ± 7.78
CPGNN-Cheby-1	81.13 ± 0.21	69.72 ± 0.59	77.79 ± 1.06	48.29 ± 2.02	36.17 ± 2.87	76.89 ± 4.95	75.00 ± 7.64
CPGNN-Cheby-2	77.68 ± 1.55	69.92 ± 0.46	78.81 ± 0.28	50.95 ± 2.46	31.29 ± 1.26	76.89 ± 5.83	75.27 ± 7.80
AM-GCN	81.70 ± 0.71	71.72 ± 0.55	-	56.70 ± 3.44	-	74.41 ± 4.50	74.11 ± 5.53
H2GCN	81.85 ± 0.38	70.64 ± 0.65	79.78 ± 0.43	59.39 ± 1.58	37.90 ± 2.02	75.13 ± 4.95	78.38 ± 6.62
MWGNN	83.30 ± 0.62	72.90 ± 0.47	82.30 ± 0.64	79.54 ± 1.28	75.41 ± 1.83	81.37 ± 4.27	79.24 ± 5.23

20% on Chameleon and Squirrel, demonstrating the effectiveness of MWGNN while the graph data is not homophily- or heterophily-dominated but a combined situation of the both.

5.4 Evaluation on Synthetic Benchmarks

To better investigate the performance of MWGNN on datasets with different Global/Local Edge Homophily distributions, we conduct experiments on a series of synthetic datasets. On the one hand, we test MWGNN on a set of synthetic graphs whose global edge homophily h evenly ranges from 0 to 1 in Figure 3.

MWGNN outperforms GCN and GAT on all homophily settings. On the other hand, we test our model on three combined graphs C.Homo, C.Mixed, C.Heter. As Figure 3 reveals, MWGNN disentangles the combined data well and achieves good results on all three synthetic graphs. Besides, on C.Mixed, GCN and GAT perform well on the sub-graph where Local Edge Homophily is high, with an accuracy over 95%. On the contrary, up to 31% nodes in the sub-graph where Local Edge Homophily is low are classified correctly. Meanwhile, MWGNN classifies the heterophily and homophily parts' nodes much better, with accuracy of 99% and 69%, separately. This observation suggests the significance of modeling the local distribution of graphs.

5.5 Ablation Study

To estimate the effectiveness of each part in MWGNN, we conduct an ablation study by removing one component at a time on our synthetic datasets, C.Heter, C.Mixed, and C.Homo. The results of the ablation study are in Table 2.

Meta-Weight Generator. We remove the Distribution Based Meta-Weight Generator of MWGNN by removing the \hat{D} in Ψ_f and Ψ_t . From the results, we can see that, in C.Heter and C.Mixed, removing the Distribution Based Meta-Weight Generator explicitly hinders the expressive power of MWGNN. C.Mixed is an example of graphs with different local distributions, homophily and heterophily parts are entangled. Besides, although C.Heter is generated by combining two heterophily graphs, the B^1 and B^2 used to construct them are different. Therefore, the distributions in C.Heter are not the same.

Without the Distribution Based Meta-Weight Generator, MWGNN can no longer capture the distributions and generate convolution adaptively. In addition, the performance of the model only drops a little because the patterns in C.Homo are relatively the same. The removal of the Distribution Based Meta-Weight Generator has little impact on the performance of the model. These results support our claim that Distribution Based Meta-Weight Generator is capable of capturing different patterns, which could be used in the Adaptive Convolution.

Table 2: Ablation Study: Accuracy of MWGNN and its variants on three synthetic combined graph.

	C.Heter	C.Mixed	C.Homo
MWGNN	56.28	76.38	95.98
w/o D	47.24	69.84	94.73
w/o D_f	51.75	74.38	96.48
w/o D_t	50.76	71.32	94.98
w/o D_p	52.26	73.78	95.21
w/o Indep. Channels	53.77	73.87	86.73

We further analyse the impact of the three components of the Distribution Based Meta-Weight Generator by removing the three local distributions D_t , D_f , and D_p one at a time. On C.Mixed, the removal of the local distributions causes the performance of MWGNN to decline in varying extents respectively, which testifies the effect of the proposed Att(\cdot) in subsection 3.1.2.

Independent Channels. To demonstrate the performance improved through the Independent Convolution Channels for Topology and Feature, we test MWGNN after disabling it in Equation 11. The results suggest that our explicit utilization of the independent channel for topology information helps the model to achieve better results. Especially, the shared patterns in C.Homo lead to the improvement.

5.6 Parameter Analysis

We investigate the sensitivity of hyper-parameters of MWGNN on Cora and Squirrel datasets. To be specific, the hyper-parameter

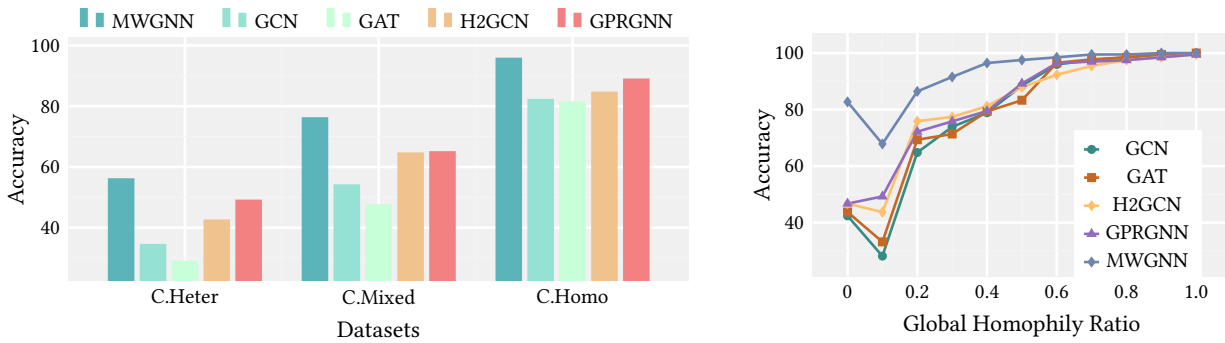


Figure 3: MWGNN and other baselines on synthetic datasets.

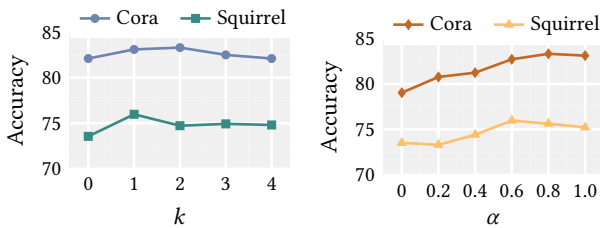


Figure 4: Parameter analysis over Cora and Squirrel on hop number k and combine alpha α .

α controlling the ratio of convolution weight S_f and S_d and the number of hops we consider when modeling the local distribution.

Analysis of k . We test MWGNN with different hop numbers k , varying it from 0 to 4. As Figure 4 shows, as k increases, the performance is generally stable. Besides, a small k is fair enough for MWGNN to reach satisfactory results.

Analysis of α . To analyse the impact of α in Equation 10, we study the performance of MWGNN with α ranging evenly from 0 to 1. Cora reaches its maximum at $\alpha = 0.8$, while squirrel reaches its maximum at $\alpha = 0.6$, which indicates that different graphs vary in the dependency on feature and topology. In addition, MWGNN is relatively stable when α changes around the maximum point.

6 RELATED WORK

6.1 Graph Neural Networks

The Graph Neural Networks (GNNs) aim to map the graph and the nodes (sometimes edges) to a low-dimension space. Scarselli et al. [23] first propose a GNN model utilizing recursive neural networks to update topologically connected nodes' information recursively. Then, Bruna et al. expand GNNs to spectral space [3]. Defferrard, Bresson, and Vandergheynst [9] generalize a simpler model, ChebyNet. Kipf and Welling [14] propose Graph Convolution Networks (GCNs) which further simplify the graph convolution operation. GAT [24] introduces an attention mechanism in feature aggregation to refine convolution on graphs. GraphSAGE [10] presents a general inductive framework, which samples and aggregates feature from local neighborhoods of nodes with various pooling methods such as mean, max, and LSTM.

6.2 GNNs on Heterophily Graphs

Many models [7, 20, 25, 34, 35] design aggregation and transformation functions elaborately to obtain better compatibility for heterophily graph data and retain their efficiency on homophily data. Geom-GCN [20] formulates graph convolution by geometric relationships in the resulting latent space. H2GCN [35] uses techniques of embedding separation, higher-order neighborhoods aggregation, and intermediate representation combination. CPGNN [34] incorporates a label compatibility matrix for modeling the heterophily or homophily level in the graph to go beyond the assumption of strong homophily. GPR-GNN [7] adopts a generalized PageRank method to optimize node feature and topological information extraction, regardless of the extent to which the node labels are homophilic or heterophilic. AM-GCN [25] fuses node features and topology information better and extracts the correlated information from both node features and topology information substantially, which may help in heterophily graph data.

7 CONCLUSION

In this paper, we focus on improving the expressive power of GNNs on graphs with different LDPs. We first show empirically that different LDPs do exist in some real-world datasets. With theoretical analysis, we show that this variety of LDPs has an impact on the performance of traditional GNNs. To tackle this problem, we propose Meta-Weight Graph Neural Network, consisting of two key stages. First, to model the NLD of each node, we construct the Meta-Weight generator with multi-scale information, including structure, feature and position. Second, to decouple the correlation between node feature and topological structure, we conduct adaptive convolution with two aggregation weights and three channels. Accordingly, we can filter the most instructive information for each node and efficiently boost the node representations. Overall, MWGNN outperforms corresponding GNNs on real-world benchmarks, while maintaining the attractive proprieties of GNNs. We hope MWGNN can shed light on the influence of distribution patterns on graphs, and inspire further development in the field of graph learning.

ACKNOWLEDGMENTS

We would like to thank Zijie Fu and Yuyang Shi for their help on language polishing. This work was supported by the National Natural Science Foundation of China (Grant No.61876006).

REFERENCES

- [1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. 2019. MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 21–29.
- [2] Reka Zsuzsanna Albert and Albert-Laszlo Barabasi. 2001. Statistical mechanics of complex networks. *Reviews of Modern Physics* 74, 1 (2001), 47–97.
- [3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. 2014. Spectral networks and locally connected networks on graphs. In *Proceedings of the 2nd International Conference on Learning Representations*.
- [4] Chen Cai and Yusu Wang. 2018. A simple yet effective baseline for non-attributed graph classification. *arXiv preprint arXiv:1811.03508* (2018).
- [5] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and Deep Graph Convolutional Networks. In *ICML (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 1725–1735.
- [6] Justin Cheng, Jon M. Kleinberg, Jure Leskovec, David Liben-Nowell, Bogdan State, Karthik Subbian, and Lada A. Adamic. 2018. Do Diffusion Protocols Govern Cascade Growth?. In *ICWSM*. AAAI Press, 32–41.
- [7] Eli Chien, Jianhao Peng, Pan Li, and Olga Milenkovic. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.
- [8] Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, Alessandro Moschitti, Bo Pang, and Walter Daelemans (Eds.). ACL, 1724–1734. <https://doi.org/10.3115/v1/d14-1179>
- [9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*. 3844–3852.
- [10] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proceedings of the 31st Conference on Neural Information Processing Systems*.
- [11] George Karypis and Vipin Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing* 20, 1 (1998), 359–392.
- [12] Leo Katz. 1953. A new status index derived from sociometric analysis. *Psychometrika* 18, 1 (1953), 39–43.
- [13] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR (Poster)*.
- [14] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*.
- [15] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR (Poster)*. OpenReview.net.
- [16] Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem. 2019. DeepGCNs: Can GCNs Go As Deep As CNNs?. In *ICCV*. IEEE, 9266–9275.
- [17] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). AAAI Press, 3538–3545.
- [18] Renjie Liao, Zhizhen Zhao, Raquel Urtasun, and Richard S. Zemel. 2019. LanczosNet: Multi-Scale Deep Graph Convolutional Networks. In *ICLR (Poster)*. OpenReview.net.
- [19] Meng Liu, Hongyang Gao, and Shuiwang Ji. 2020. Towards Deeper Graph Neural Networks. In *KDD*. ACM, 338–348.
- [20] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-GCN: Geometric Graph Convolutional Networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.
- [21] Leonardo F.R. Ribeiro, Pedro H.P. Saverese, and Daniel R. Figueiredo. 2017. struc2vec: Learning Node Representations from Structural Identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 385–394.
- [22] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2021. Multi-Scale attributed node embedding. *J. Complex Networks* 9, 2 (2021). <https://doi.org/10.1093/comnet/cnab014>
- [23] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE Trans. Neural Networks* 20, 1 (2009), 61–80. <https://doi.org/10.1109/TNN.2008.2005605>
- [24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the 6th International Conference on Learning Representations*.
- [25] Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, and Jian Pei. 2020. AM-GCN: Adaptive Multi-channel Graph Convolutional Networks. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 1243–1253. <https://doi.org/10.1145/3394486.3403177>
- [26] Boris Weisfeiler and A. A. Lehman. 1968. A Reduction of a Graph to a Canonical Form and an Algebra Arising During This Reduction. *Nauchno-Tekhnicheskaya Informatsia Ser. 2*, N9 (1968), 12–16.
- [27] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. In *ICML (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 6861–6871.
- [28] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016 (JMLR Workshop and Conference Proceedings, Vol. 48)*, Maria-Florina Balcan and Kilian Q. Weinberger (Eds.). JMLR.org, 40–48.
- [29] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do Transformers Really Perform Bad for Graph Representation? *CoRR* abs/2106.05234 (2021). [arXiv:2106.05234](https://arxiv.org/abs/2106.05234)
- [30] Jiaxuan You, Jonathan M. Gomes-Selman, Rex Ying, and Jure Leskovec. 2021. Identity-aware Graph Neural Networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 10737–10745.
- [31] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. 2018. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. In *Proceedings of the 32nd Conference on Neural Information Processing Systems*.
- [32] Jiaxuan You, Rex Ying, and Jure Leskovec. 2019. Position-aware Graph Neural Networks. In *ICML (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 7134–7143.
- [33] Yue Yu, Jie Chen, Tian Gao, and Mo Yu. 2019. DAG-GNN: DAG Structure Learning with Graph Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 7154–7163.
- [34] Jiong Zhu, Ryan A. Rossi, Anup Rao, Tung Mai, Nedom Lipka, Nesreen K. Ahmed, and Danai Koutra. 2021. Graph Neural Networks with Heterophily. In *AAAI*. AAAI Press, 11168–11176.
- [35] 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.).

A DETAILED INFORMATION FOR DATASETS

Table 3: Statistics for real-world datasets.

Dataset	$ \mathcal{V} $	$ \mathcal{E} $	$ \mathcal{Y} $	F	h
Corra	2708	10556	7	1433	0.81
Citeseer	3327	9104	6	3703	0.74
Pubmed	19717	88648	3	500	0.8
Texas	183	325	5	1703	0.11
Cornell	183	298	5	1703	0.31
Chameleon	2277	36101	5	1703	0.2
Squirrel	5201	217073	5	2089	0.22

Table 4: Statistics for combined synthetic datasets. h_1, h_2 denotes h for separate graphs that are used to combine.

Dataset	$ \mathcal{V} $	$ \mathcal{E} $	$ \mathcal{Y} $	F	h	h_1	h_2
C.Homo	1000	22937	5	100	0.51	0.50	0.50
C.Mixed	1000	22705	5	100	0.39	0.10	0.75
C.Heter	1000	15201	5	100	0.20	0.20	0.20

B PROOFS OF THEOREM 4.1

Notation. $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denotes a graph with d -dimension node feature vector \mathbf{x}_i for $v_i \in \mathcal{V}$. Features of all dimensions are bounded by a positive scalar C_x . y_i denotes the label for node v_i . P is a random variable for local edge homophily with its distribution as \mathcal{D}_P . \mathbf{h}_i denotes the embedding of node v_i . $\mathbf{W} \in \mathbb{R}^{d \times d}$ denotes the parameter matrix of 1-layer GCN model. $\rho(\mathbf{W})$ denotes the largest singular value of \mathbf{W} .

Assumptions on Graphs. (1) \mathcal{G} is k -regular. It can prevent us from trivial discussion on the expression of GCN and help us focus on the mechanism of message passing. (2) The features of node v_i are sampled from feature distribution \mathcal{F}_{y_i} , i.e. $\mathbf{x}_i \sim \mathcal{F}_{y_i}$, with $\mu(\mathcal{F}_{y_i}) = \mathbb{E}[\mathbf{x}_i | y_i]$ and $\tau(\mathcal{F}_{y_i}) = \mathbb{E}[\mathbf{x}_i \circ \mathbf{x}_i | y_i]$. Similarly, $\mathcal{F}_{\bar{y}_i}$ denotes the feature distribution of nodes having labels other than y_i . (3) Dimensions of \mathbf{x}_i are independent to each other and they are all bounded by a positive scalar C_x . (4) Dimensions of $\mu(\mathcal{F}_{y_i})$ and $\tau(\mathcal{F}_{y_i})$ are all bounded by positive scalars C_μ and C_τ , respectively. (5) For node v_i , its local edge homophily p is sampled from pattern distribution \mathcal{D}_P . If $P = p$, node v_i 's neighbors' labels are independently sampled from Bernoulli distribution $B(y_i, p)$.

LEMMA B.1 (BERNSTEIN'S INEQUALITY). *Let X_1, \dots, X_n be independent bounded random variables with $X_i \in [a, b]$ for any i , where $-\infty < a \leq b < +\infty$. Denote that $\bar{X} = \frac{1}{n}(X_1 + \dots + X_n)$ and $\sigma^2 = \sum_{i=1}^n \text{Var}[X_i]$. Then for any $t > 0$, the following inequalities hold:*

$$\mathbb{P}(\bar{X} - E[\bar{X}] \geq t) \leq \exp\left(-\frac{nt^2}{2\sigma^2 + 2t(b-a)/3}\right),$$

$$\mathbb{P}(\bar{X} - E[\bar{X}] \leq -t) \leq \exp\left(-\frac{nt^2}{2\sigma^2 + 2t(b-a)/3}\right).$$

LEMMA B.2 (HOEFFDING'S INEQUALITY). *Let X_1, \dots, X_n be independent bounded random variables with $X_i \in [a, b]$ for any i , where $-\infty < a \leq b < +\infty$. Denote that $\bar{X} = \frac{1}{n}(X_1 + \dots + X_n)$. Then for any $t > 0$, the following inequalities hold:*

$$\mathbb{P}(\bar{X} - E[\bar{X}] \geq t) \leq \exp\left(-\frac{2nt^2}{(b-a)^2}\right),$$

$$\mathbb{P}(\bar{X} - E[\bar{X}] \leq -t) \leq \exp\left(-\frac{2nt^2}{(b-a)^2}\right).$$

LEMMA B.3 (THE UNION BOUND). *For any events A_1, A_2, \dots, A_n , we have*

$$\mathbb{P}\left(\bigcup_{i=1}^n A_i\right) \leq \sum_{i=1}^n \mathbb{P}(A_i).$$

THEOREM B.4. *Consider $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \{\mathcal{F}_c, c \in \{0, 1\}\}, \{\mathcal{D}_P, P \sim \mathcal{D}_P\}, k\}$, which follows assumptions (1) - (5). For any node $v_i \in \mathcal{V}$, the expectation of its pre-activation output of 1-layer GCN model is as follows:*

$$\mathbb{E}[\mathbf{h}_i] = \mathbf{W} \left(\frac{1}{k+1} \mu(\mathcal{F}_{y_i}) + \frac{k}{k+1} \mathbb{E}_{P \sim \mathcal{D}_P, c \sim B(y_i, p), \mathbf{x}_j \sim \mathcal{F}_c}[\mathbf{x}_j] \right).$$

For any $t > 0$, the probability that the Euclidean distance between the observation \mathbf{h}_i and its expectation is larger than t is bounded as follows:

$$\mathbb{P}(\|\mathbf{h}_i - \mathbb{E}[\mathbf{h}_i]\|_2 \geq t) \leq 2d \exp\left(-\frac{((k+1)t_2/\rho(\mathbf{W}) + \sqrt{d}C_x + \sqrt{d}C_\mu)^2}{2kd\sigma^2 + 4\sqrt{d}C_x((k+1)t_2/\rho(\mathbf{W}) + \sqrt{d}C_x + \sqrt{d}C_\mu)/3}\right),$$

where $\sigma^2 = 4kC_\mu^2 \text{Var}[P] + kC_\tau$.

PROOF. For a single layer GCN model, the process can be written in the following form for node v_i

$$\mathbf{h}_i = \sum_{j \in \{i\} \cup \mathcal{N}(i)} \frac{1}{k+1} \mathbf{W} \mathbf{x}_j,$$

so that the expectation of \mathbf{h}_i can be derived as follows:

$$\begin{aligned} \mathbb{E}[\mathbf{h}_i] &= \mathbb{E}\left[\sum_{j \in \{i\} \cup \mathcal{N}(i)} \frac{1}{k+1} \mathbf{W} \mathbf{x}_j\right] \\ &= \frac{1}{k+1} \mathbf{W} \mathbb{E}[\mathbf{x}_i] + \sum_{j \in \mathcal{N}(i)} \frac{1}{k+1} \mathbf{W} \mathbb{E}[\mathbf{x}_j] \\ &= \frac{1}{k+1} \mathbf{W} \mu(\mathcal{F}_{y_i}) + \frac{k}{k+1} \mathbf{W} \mathbb{E}_{P \sim \mathcal{D}_P, c \sim B(y_i, p), \mathbf{x}_j \sim \mathcal{F}_c}[\mathbf{x}_j] \\ &= \mathbf{W} \left(\frac{1}{k+1} \mu(\mathcal{F}_{y_i}) + \frac{k}{k+1} \mathbb{E}_{P \sim \mathcal{D}_P, c \sim B(y_i, p), \mathbf{x}_j \sim \mathcal{F}_c}[\mathbf{x}_j] \right). \end{aligned}$$

When P is given, the conditional distribution of \mathbf{x}_j follows

$$(\mathbf{x}_j | P = p) \sim \begin{cases} p\mathcal{F}_{y_i} + (1-p)\mathcal{F}_{\bar{y}_i}, & j \in \mathcal{N}(i), \\ \mathcal{F}_{y_i}, & j = i. \end{cases}$$

Let $\mathbf{x}_j^l, l = 1, \dots, d$ denote the l -th element of \mathbf{x}_j . Then, for any dimension l , $\{\mathbf{x}_j^l, j \in \mathcal{N}(i)\}$ is a set of independent random variables.

When $j \in \mathcal{N}(i)$

$$\begin{aligned}
\text{Var} \left[\mathbb{E} \left[\mathbf{x}'_j | P \right] \right] &= \text{Var} \left[P \mu^l(\mathcal{F}_{y_i}) + (1-P) \mu^l(\mathcal{F}_{\bar{y}_i}) \right] \\
&= (\mu^l(\mathcal{F}_{y_i}) - \mu^l(\mathcal{F}_{\bar{y}_i}))^2 \text{Var} [P]. \\
&\leq 4C_\mu^2 \text{Var} [P]. \\
\mathbb{E} \left[\text{Var} \left[\mathbf{x}'_j | P \right] \right] &= \mathbb{E} \left[\left(\mathbf{x}'_j \right)^2 | P \right] - \mathbb{E} \left[\mathbf{x}'_j | P \right]^2 \\
&= \mathbb{E} \left[P \tau^l(\mathcal{F}_{y_i}) + (1-P) \tau^l(\mathcal{F}_{\bar{y}_i}) \right] \\
&\quad - \mathbb{E} \left[\left(P \mu^l(\mathcal{F}_{y_i}) + (1-P) \mu^l(\mathcal{F}_{\bar{y}_i}) \right)^2 \right] \\
&= -\mathbb{E} [P^2] (\mu^l(\mathcal{F}_{y_i}) - \mu^l(\mathcal{F}_{\bar{y}_i}))^2 \\
&\quad + \mathbb{E} [P] \left(-2(\mu^l(\mathcal{F}_{y_i}) - \mu^l(\mathcal{F}_{\bar{y}_i})) \mu^l(\mathcal{F}_{\bar{y}_i}) + \tau^l(\mathcal{F}_{y_i}) - \tau^l(\mathcal{F}_{\bar{y}_i}) \right) \\
&\quad + \tau^l(\mathcal{F}_{\bar{y}_i}) \\
&\leq C_\tau.
\end{aligned}$$

By the law of total variance,

$$\text{Var} \left[\mathbf{x}'_j \right] = \text{Var} \left[\mathbb{E} \left[\mathbf{x}'_j | P \right] \right] + \mathbb{E} \left[\text{Var} \left[\mathbf{x}'_j | P \right] \right] \leq 4C_\mu^2 \text{Var} [P] + C_\tau.$$

Then for any $t_1 > 0$, we have the following bound by applying lemma Theorem B.1 and lemma Theorem B.3:

$$\begin{aligned}
&\mathbb{P} \left[\frac{1}{k+1} \left\| \sum_{j \in \{i\} \cup \mathcal{N}(i)} (\mathbf{x}'_j - \mathbb{E} [\mathbf{x}'_j]) \right\| \geq t_1 \right] \\
&\leq \mathbb{P} \left[\frac{1}{k} \left\| \sum_{j \in \mathcal{N}(i)} (\mathbf{x}'_j - \mathbb{E} [\mathbf{x}'_j]) \right\| \geq \frac{(k+1)t_1 + C_x + C_\mu}{k} \right] \\
&\quad + \mathbb{P} \left[\left\| \mathbf{x}'_i - \mathbb{E} [\mathbf{x}'_i] \right\| \geq C_x + C_\mu \right] \\
&\leq 2 \exp \left(-\frac{((k+1)t_1 + C_x + C_\mu)^2}{2k\sigma^2 + 4C_x((k+1)t_1 + C_x + C_\mu)/3} \right).
\end{aligned}$$

where $\sigma^2 = 4kC_\mu^2 \text{Var} [P] + kC_\tau$. By applying lemma Theorem B.3 to Theorem B.1, the following holds:

$$\begin{aligned}
&\mathbb{P} \left(\frac{1}{k+1} \left\| \sum_{j \in \{i\} \cup \mathcal{N}(i)} (\mathbf{x}_j - \mathbb{E} [\mathbf{x}_j]) \right\|_2 \geq t_2 \right) \\
&\leq \sum_{l=1}^d \mathbb{P} \left[\frac{1}{k+1} \left\| \sum_{j \in \{i\} \cup \mathcal{N}(i)} (\mathbf{x}'_j - \mathbb{E} [\mathbf{x}'_j]) \right\| \geq \frac{t_2}{\sqrt{d}} \right] \\
&\leq 2d \exp \left(-\frac{((k+1)t_2 + \sqrt{d}C_x + \sqrt{d}C_\mu)^2}{2kd\sigma^2 + 4\sqrt{d}C_x((k+1)t_2 + \sqrt{d}C_x + \sqrt{d}C_\mu)/3} \right).
\end{aligned}$$

Furthermore, we have

$$\begin{aligned}
\|\mathbf{h}_i - \mathbb{E} [\mathbf{h}_i]\|_2 &\leq \frac{1}{k+1} \left\| \mathbf{W} \left(\sum_{j \in \{i\} \cup \mathcal{N}(i)} \mathbf{x}_j - \mathbb{E} [\mathbf{x}_j] \right) \right\|_2 \\
&\leq \frac{\|\mathbf{W}\|_2}{k+1} \left\| \sum_{j \in \{i\} \cup \mathcal{N}(i)} \mathbf{x}_j - \mathbb{E} [\mathbf{x}_j] \right\|_2 \\
&\leq \frac{\rho(\mathbf{W})}{k+1} \left\| \sum_{j \in \{i\} \cup \mathcal{N}(i)} \mathbf{x}_j - \mathbb{E} [\mathbf{x}_j] \right\|_2,
\end{aligned}$$

where $\|\mathbf{W}\|_2$ denotes the matrix 2-norm of \mathbf{W} and $\rho(\mathbf{W})$ denotes the largest singular value of \mathbf{W} . Then, for any $t > 0$, we have

$$\begin{aligned}
\mathbb{P} (\|\mathbf{h}_i - \mathbb{E} [\mathbf{h}_i]\|_2 \geq t) &\leq \mathbb{P} \left(\frac{\rho(\mathbf{W})}{k+1} \left\| \sum_{j \in \{i\} \cup \mathcal{N}(i)} \mathbf{x}_j - \mathbb{E} [\mathbf{x}_j] \right\|_2 \geq t \right) \\
&= \mathbb{P} \left(\frac{1}{k+1} \left\| \sum_{j \in \{i\} \cup \mathcal{N}(i)} \mathbf{x}_j - \mathbb{E} [\mathbf{x}_j] \right\|_2 \geq \frac{t}{\rho(\mathbf{W})} \right) \\
&\leq 2d \exp \left(-\frac{((k+1)t_2/\rho(\mathbf{W}) + \sqrt{d}C_x + \sqrt{d}C_\mu)^2}{2kd\sigma^2 + 4\sqrt{d}C_x((k+1)t_2/\rho(\mathbf{W}) + \sqrt{d}C_x + \sqrt{d}C_\mu)/3} \right),
\end{aligned}$$

which concludes the proof. \square