ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240820

Laner-GNN: Adapting to Long-Tail Degree Distribution with Latent Neighborhood Restorers

Lingjun Xu[a,*], Haowen Li[a], Guojie Song[a,], Liang Wang[b] and Bo Zheng[b]**

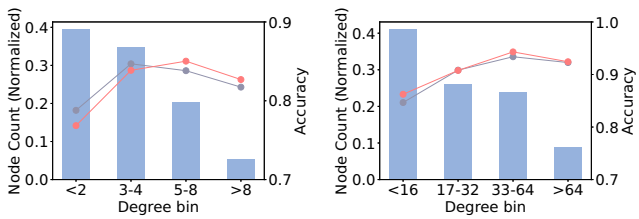
[a]National Key Laboratory of General Artificial Intelligence, School of Intelligence Science and Technology, Peking University, Beijing, China
[b]Alibaba Group, Beijing, China

Abstract. Graph Neural Networks (GNNs) learn the correlation between local graph neighborhoods and node properties, with the requirement of adequate complete and clean neighborhoods. However, the intrinsic structural characteristics of the long-tail node degree distribution in most real-world graphs violate the assumption. Numerous low-degree nodes lack accurate representation due to limited connections, while bits of high-degree nodes include redundancy and errors in observed neighborhoods, harming the performance of GNNs. In this paper, we propose Laner-GNN, which performs latent neighborhood restoration adapting to aforesaid structural characteristics and improves representation capacity. Specifically, to reduce computational complexity, latent disentangled neighborhood factors are extracted and then manipulated for restoration instead of explicitly altering the entire adjacency matrix. A Degree-Adaptive Restorer captures the dependency of restoration on node degrees, while a Label-Informed Restorer and the Structured Field Augmented Priori enhance restoration with information from the prediction target and the graph structure. Extensive experiments on real-world datasets in different domains verify that our model excels existing SOTAs in different degree bins, thus uplifting the model performance in the whole dataset.

1 Introduction

Graphs are powerful tools to represent relationships among entities, and have been applied in a wide variety of real-world domains, such as e-commercial platforms and social recommendation. A crucial task on graphs is to predict node properties like class labels, utilizing node attributes and edges depicting interactions between nodes. One of the most influential paradigms to achieve this goal is Graph Neural Networks (GNNs) [8, 4, 3, 19]. By performing message-passing within a node's local neighborhood, GNNs approximate functions defined on graphs that are related to the target properties. To train GNNs, it is desirable to have sufficient data to capture the correlation between the node neighborhood and the target to predict, i.e., adequate complete and clean neighborhoods along with their labels.

However, nodes in real-world graphs often exhibit a long-tail degree distribution, where diverse data patterns are associated with different degrees. This intrinsic structure significantly impacts the performance of GNNs. On one hand, a large fraction of nodes in the graph have low degrees (we call such nodes *tail nodes*), and their observed neighborhood is sparse and incomplete. On the other hand, extremely high-degree nodes (*head nodes*) are connected to numerous neighbors, some of which may be redundant or even spurious. The limited population of head nodes also hinders models from learning essential features within such data. To provide a more vivid depiction of the long-tail degree distribution, we take real-world e-commerce graphs as example, where nodes are items and edges are co-purchases. In these graphs, most of the items are cold (tail nodes) and isolated from other items, while a few extremely popular items (head nodes) can be co-purchased with a diverse set of items that may be distinct from them. In both scenarios, GNN models struggle to accurately capture the unique characteristics of these items. In stark contrast, the nodes in the middle part of the distribution (*body nodes*) have an appropriate population and relatively complete and less noisy observed neighborhoods, on which GNNs can achieve better performance. Figure 1 shows several benchmark graphs exhibiting a long-tail degree distribution. The performance of existing GNNs deteriorates when dealing with nodes of either low or high degrees because they treat all nodes identically and ignore the structural characteristics of the degree distribution. A naïve idea is to discard the nodes at both ends and train GNNs only on body nodes, which does nothing to solve the problem. Though nodes with different degrees cannot be managed identically, they play a vital role in the training of GNNs. Tail nodes constructively support effective training with their abundance, while head nodes offer valuable information for the model despite potential issues with their population or neighborhoods. Therefore, to better model the correlation between node neighborhood and target properties, we need to design GNN models and corresponding training paradigms that can adapt to and effectively utilize nodes



(a) Cora (b) Photo

Figure 1. Node count with different degrees (blue bars) and the performance of GCN [8] (grey lines) and GAT [19] (red lines) in all bins on Cora and Photo. The number of nodes decreases with node degree increases, and models perform worse at both ends of the degree distribution.

* Email: xlj_rk@pku.edu.cn
** Corresponding Author. Email: gjsong@pku.edu.cn

over the long-tail degree distribution.

Issues about degree distribution on graphs have gained increasing attention, the majority of which focus on the incompleteness of neighborhoods of tail nodes. They either re-normalize the adjacency matrix to exert higher impact from tail nodes over model training [7] or learn to complete the neighborhood with generative models [10] or knowledge transferring techniques [12, 26]. Nevertheless, rare attention is paid to the head nodes, except [13] introduces debiasing functions to relieve unfair issues introduced by structure. Another line of work considers optimizing graph structure and the GNN model simultaneously [5, 20, 9]. Unfortunately, such methods explicitly learn to adjust the whole adjacency matrix, neglecting local characteristics of node neighborhoods and high in complexity.

In this paper, we consider nodes all over the long-tail degree distribution, and *restore* the local neighborhood adapting to the intrinsic structural characteristics. However, there are several challenges associated with this restoration process. First, to reduce complexity and prioritize the node-centered characteristics, it is crucial to recover the local neighborhood without explicitly learning to manipulate the entire graph structure. Also, the position of nodes in the degree distribution decides the difference between the neighborhood currently observed and the ideal (ground-true) one. Effectively incorporating this dependency on degrees into the restoration process remains a challenging task. Furthermore, the prediction target, along with information relevant to prediction from the graph (e.g. feature distribution and topological structure), should guide and facilitate adjustments for better neighborhood restoration.

To combat these challenges, here we propose the **La**tent **Ne**ighborhood **R**estorers Boosted **G**raph **N**eural **N**etworks (Laner-GNN). Specifically, we model the correlation between the node neighborhood and the target under the framework of variational inference. First, rather than learning to modify explicit edges in the graph, we treat the latent variables in variational inference as disentangled structural factors and manipulate them to replenish or diminish the neighborhood structures. On top of that, we elaborate a Degree-Adaptive Restorer to inject the dependency on node degrees. To bring tighter integration of the prediction target and relevant information in the graph to local neighborhood restoration, a Label-Informed Restorer and the Structured Field Augmented Priori are introduced. Additionally, the high-quality neighborhoods of body nodes are leveraged to achieve improved model initialization and train the restorers. Subsequently, all restored nodes, spanning various degrees, actively partake in the training process, providing adequate data and information. Extensive experimental results on six real-world benchmarks from diverse domains with long-tail degree distribution demonstrate the improvements brought by our methods, especially on tail and head nodes.

We summarized our contribution as follows:

- We dive into the intrinsic structural characteristics of the long-tail degree distribution, and give an overall consideration on nodes throughout the distribution, rather than only assuming that tail nodes are problematic.
- We propose Laner-GNN performing latent local neighborhood restoration utilizing degree information, prediction target and multifaceted information in the graph.
- We conduct experiments on six real-world datasets, and our model excels existing SOTAs in different degree bins, demonstrating the effectiveness of our design.

2 Related Work

2.1 Graph Neural Networks

Due to superior performance, GNNs have been widely used for graph analysis. The main idea of GNNs is message passing and neighborhood aggregation. Early GNNs [8, 4] obtain the representation of the central node by iteratively aggregating the representation of all or a sampled set of its neighbor nodes. [19, 2, 14, 27] further improve by adjusting the aggregation weight according to the importance of neighbors to reduce noise. Most of the existing GNNs assume that the underlying graphs are correct and reliable, but graphs in the real world may either contain error information or miss important ones, thus bringing greater bias to GNN models.

2.2 Improving Graph Neural Networks on Low-Degree Nodes

The issues of degree-related biases have gained increasing concerns in the field of graph embedding. Existing methods [21, 18] build multiple models for nodes with different degrees using hashing techniques, which suffer from high space complexity and neglect the problems existing in local neighborhood. Particularly, recent studies have been devoted to improving the performance of GNNs on tail nodes with very low degrees and large populations. RawlsGCN [7] increases the influence of low-degree nodes on model optimization by re-normalizing the adjacency matrix to a doubly stochastic matrix. meta-tail2vec [11] is a meta-learning framework for local-aware representation learning, which specifies the task of learning tail node representations as personalized regression problems based on local context. Tail-GNN [12] further introduces the concept of neighborhood transformation to model the variable relationships between target nodes and their neighbors. LAGNN [10] utilizes Variational Auto-encoder to learn the distribution of neighbor node representations conditioned on the central node representation and enhances the expressive power of graph neural networks through the generated features. Cold Brew [26], on the other hand, addresses isolated nodes and achieves the transfer of graph structural information using knowledge distillation techniques. DegFairGNN [13] introduces the problem of general degree unfairness and employs learnable debiasing functions for nodes with different structures. Some studies [25, 1] consider both degree and class long-tailedness, which are outside the scope of our discussion. Different from our model, these existing models for the long-tail degree distribution pay more attention to tail nodes while ignoring the problems of high-degree ones.

2.3 Graph Structure Learning

Graph structure learning (GSL) jointly optimizes graph structures and node representations to improve learning quality. As standing-out representatives, Pro-GNN [5] takes into account some properties shared by real-world graphs, such as low rank, sparsity and feature smoothness. GEN [20] takes the original graph structure, node features and multi-order neighborhood information as multifaceted observations to estimate a better graph using Bayesian inference, and then optimizes the parameters of GNN. WSGNN [9] harnesses a variational inference approach to solve a label-structure joint estimation problem, improving the robustness of the model when both labels and edges are sparse. However, these GSL methods explicitly learning to adjust the graph structure are high in complexity.

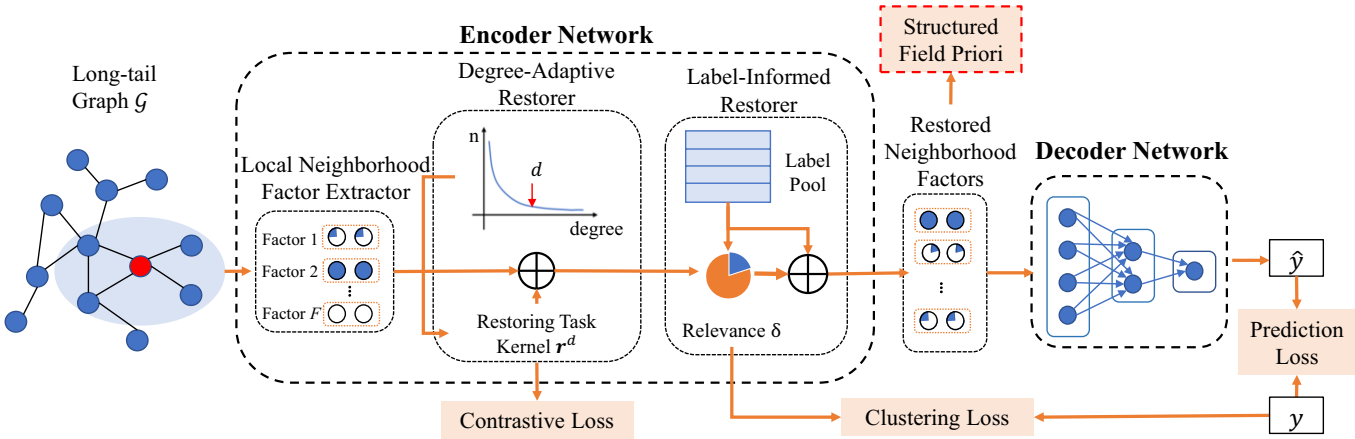


Figure 2. Overview framework of Laner-GNN made up of an encoder network and a decoder network defined with variational inference. Specifically in the former one, local neighborhood factors are first extracted, and then go through the Degree-Hinted Restorer and the Label-Informed Restorer to get more perfect neighborhoods. Then, the restored neighborhood factors are fed into the decoder network to get predictions.

3 Preliminaries

Graph. A *graph* is represented by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, where $\mathcal{V}$ is the node set and $\mathcal{E}$ is the edge set. The edges can also be represented by an adjacency matrix $\mathcal{A} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$, where $\mathcal{A}_{i,j} = 1$ iff $(v_i, v_j) \in \mathcal{E}$. $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}|\times d_x}$ is the node feature matrix, each row $\boldsymbol{x}_v$ of which denotes the features of node $v \in \mathcal{V}$. $\boldsymbol{Y} \in \mathbb{R}^{|\mathcal{V}|\times d_y}$ represents the node properties to predict. For example, in the node classification task, each row $\boldsymbol{y}_v \in \boldsymbol{Y}$ can be class labels. For each node v, *degree* d_v refers to the number of edges connected to it.

Problem. We consider the problem of predicting node properties on graphs. Specifically, in graph $\mathcal{G}$, the task is to estimate the distribution of node labels $P(\boldsymbol{Y}|\mathcal{G})$. We optimize a machine learning model parameterized by Θ using Maximum Likelihood Estimation (MLE):

$$\max \log P_\Theta(\boldsymbol{Y}|\mathcal{G}). \tag{1}$$

GNN models factorize and approximate the probability based on the local neighborhood $\mathcal{N}_v = \{v\} \cup \{u|(u,v) \in \mathcal{E}\}$ of each node v:

$$P_\Theta(\boldsymbol{Y}|\mathcal{G}) = \sum_{v\in\mathcal{V}} P_\Theta(\boldsymbol{y}_v|\mathcal{N}_v), \tag{2}$$

where $|\mathcal{N}_v| \propto d_v$. The error of such approximation depends strongly on the quality of the observed neighborhood $\mathcal{N}_v$, which is related to the neighborhood size.

Our goal is to design a better form of approximating function that can reduce the negative impact of low-quality neighborhoods, achieve better performance on nodes with different degrees, and thus improve the overall performance on the whole dataset.

4 Proposed Methods

4.1 Overall Framework

Instead of explicitly learning the optimal graph structure by generating/deleting the edges (i.e., editing the whole adjacency matrix $\mathcal{A}$), we propose to introduce an extra latent variable $\boldsymbol{M}$ as structural factors that constitute the neighborhoods. Each factor may represent specific patterns within the neighborhood. By manipulating and regularizing these factors, we adjust the composition of neighborhoods and achieve implicit restoration.

Incorporating $\boldsymbol{M}$, we rewrite Equation (1) as follows:

$$\log P_\Theta(\boldsymbol{Y}|\mathcal{G}) = \log \int P_\Theta(\boldsymbol{Y}, \boldsymbol{M}|\mathcal{G}) d\boldsymbol{M}. \tag{3}$$

To optimize this objective, we follow the variational inference approach [6] to derive the Evidence Lower Bound (ELBO) for the observed data:

$$\begin{aligned} &\log P_\Theta(\boldsymbol{Y}|\mathcal{G}) \\ \geq &\int Q_\phi(\boldsymbol{M}|\boldsymbol{Y},\mathcal{G}) \log \frac{P_\Theta(\boldsymbol{M}|\mathcal{G}) P_\Theta(\boldsymbol{Y}|\mathcal{G},\boldsymbol{M})}{Q(\boldsymbol{M}|\boldsymbol{Y},\mathcal{G})} \\ = &\mathbb{E}_{Q_\phi(\boldsymbol{M}|\cdot)}[\log P_\Theta(\boldsymbol{Y}|\mathcal{G},\boldsymbol{M})] \\ &- D_{KL}(Q_\phi(\boldsymbol{M}|\boldsymbol{Y},\mathcal{G})||P_\Theta(\boldsymbol{M}|\mathcal{G})). \end{aligned} \tag{4}$$

The overview framework of our model is shown in Figure 2. Practically, we need to instantiate the variational distribution ($Q_\phi(\boldsymbol{M}|\boldsymbol{Y},\mathcal{G})$, $P_\Theta(\boldsymbol{Y}|\mathcal{G},\boldsymbol{M})$ and $P_\Theta(\boldsymbol{M}|\mathcal{G})$) with neural networks and learn to find their optimal parameters.

The key lies in how the encoder network (modeling $Q_\phi(\boldsymbol{M}|\boldsymbol{Y},\mathcal{G})$) extracts and restores the latent neighborhood of nodes. To cope with the intrinsic structural characteristics of the long-tail degree distribution, we hope that the distribution modeling is dependent on the node degree (thus we denote it as $Q_{\phi(d)}(\boldsymbol{M}|\boldsymbol{Y},\mathcal{G})$ instead). Moreover, the prediction target $\boldsymbol{Y}$ also plays an important guiding role in the network. The details are introduced in Section 4.2.

The prior $P_\Theta(\boldsymbol{M}|\mathcal{G})$ imposes distribution constraints on the restored latent factors outputted by the encoder network leveraging existing information from the data, making sure that the restoration accords with the expectation. To capture the complex structural characteristics of the graph, we propose using a structured field, further introduced in Section 4.3.

The decoder network (modeling $P_\Theta(\boldsymbol{Y}|\mathcal{G},\boldsymbol{M})$) performs target prediction given the recovered neighborhood factors which are endowed with sufficient knowledge. For simplicity, we instantiate it with a Multilayer Perceptron (MLP):

$$\hat{\boldsymbol{y}}_v = \mathrm{MLP}([\boldsymbol{W}_x\boldsymbol{x}_v||\boldsymbol{m}_v]), \tag{5}$$

and learn by minimizing the prediction loss $\mathcal{L}_{pred} = -\log P_\Theta(\boldsymbol{Y}|\mathcal{G}, \boldsymbol{M})$ given the observed training set.

4.2 Latent Neighborhood Restoration

In this section, we present the panorama of the encoder network with latent neighborhood restoration. First, we extract the initial latent neighborhood factors. Then, the Degree-Adaptive Restorer mines information ***inside*** the degree distribution. The Label-Informed Restorer further resorts to ***external*** sources of information relevant to the prediction, complementing each other.

4.2.1 Local Neighborhood Factor Extractor

We define that each row of $\boldsymbol{M}$ contains multiple latent neighborhood structural factors, which are disentangled so they may focus on mutually distinct structural components. We harness an extracting module inspired by [14] to get original factors from the currently observed neighborhoods. The representations of factors are initialized with:

$$\boldsymbol{z}_{v,i} = \mathrm{norm}(\sigma(\boldsymbol{W}_i \boldsymbol{x}_v + \boldsymbol{b}_i)), \quad i = 1, 2, ..., F, \tag{6}$$

where F is the number of factors. Then, to achieve factor disentanglement, they are iteratively spread to different fields with T updates:

$$p_{v,i} = \mathrm{softmax}_F(\cos(\boldsymbol{z}_{u,i}, \boldsymbol{c}_{v,i})/\tau_1), \quad u \in \mathcal{N}_v, \tag{7}$$

$$\boldsymbol{c}_{v,i} = \boldsymbol{z}_{v,i} + \sum_{u \in \mathcal{N}_v} p_{u,i} \boldsymbol{z}_{u,i}, \quad i = 1, 2, ..., F, \tag{8}$$

where $\boldsymbol{c}_{v,i}$ are initialized with $\boldsymbol{z}_{v,i}$ and τ_1 is a hyper-parameter that controls the hardness of factor assignment.

4.2.2 Degree-Adaptive Restorer

Intuitively, the disparity between the observed neighborhood and the ground truth depends on node degree. For head/tail nodes, there may be a huge gap, while for body nodes the observation may have minor differences with the ground truth. Hence, we propose to **use node degree information as clues in neighborhood restoration**.

Here we define the task for degree d as outputting the ground-true factor composing the ideal neighborhood $\tilde{\boldsymbol{c}}_{v,i} = f_d^i(\boldsymbol{c}_{v,i})$ given the observed neighborhood factor $c_{v,i}$, where f_d is the mapping performed by the task. For different degrees, the tasks are different while sharing global correlation. Specifically, we consider $d_v = d_v / \max(d)$ as the normalized degree to show the position of node v within the degree distribution. Suppose we already have pairs of data $\mathcal{D} = \{(\boldsymbol{c}_{u,i}, \tilde{\boldsymbol{c}}_{u,i})\}$ to learn the relationship, and we first introduce the **architecture of the restorer model**. Then explain how to obtain and utilize the data $\mathcal{D}$, and **how the model is trained**.

Degree-Adaptive Restoration Task. We represent the task with a degree-adaptive *kernel* vector. First, data pairs are embeded with node degrees:

$$\boldsymbol{r}_{u,i} = \boldsymbol{W}_{r,i}[d_u || \boldsymbol{c}_{u,i} || \tilde{\boldsymbol{c}}_{u,i}], \quad (\boldsymbol{c}_{u,i}, \tilde{\boldsymbol{c}}_{u,i}) \in \mathcal{D}. \tag{9}$$

To produce more adaptive contexts for each node v, we re-weight the data points with the similarity between them and node v and aggregate them to get the task kernel for degree d_v. We consider degrees and observed factors here, because they stand for the task by definition and the characteristics of the current neighborhood:

$$weight_{v,u} = \mathrm{softmax}_u(\exp(-|d_u - d_v|^2/\tau_2)\cos(\boldsymbol{c}_{u,i}, \boldsymbol{c}_{v,i})), \tag{10}$$

$$\boldsymbol{r}_i^{d_v} = \sum_{u \in \mathcal{D}} weight_{v,u} \boldsymbol{r}_{u,i}, \tag{11}$$

where τ_2 is the hyper-parameter controlling the correlation across degrees. To avoid extra calculation, we propose to use a cache with LRU replacement mechanism to store the calculated kernels during training, and directly refer to the cache during inference.

The calculated kernels are then fused with the input to perform restoration. Here, we propose a scale-and-shift mechanism. Given degree clues, the model learns adaptively to address varying degrees. The gate $\boldsymbol{\gamma}_{v,i}$ reduces the intensity of factors to remove excessive neighborhood factors. Also, the bias $\boldsymbol{\beta}_{v,i}$ can introduce additional knowledge from the task kernel exposure to data beyond the current neighborhood during the whole learning process:

$$\tilde{\boldsymbol{c}}_{v,i} = \boldsymbol{\gamma}_{v,i} \odot \boldsymbol{c}_{v,i} + \boldsymbol{\beta}_{v,i}, \tag{12}$$

where the gate and bias are calculated as:

$$\boldsymbol{\gamma}_{v,i} = \tanh(\boldsymbol{W}_{\gamma,i}[d_v || \boldsymbol{r}_i^{d_v}] + \boldsymbol{b}_{\gamma,i}), \tag{13}$$

$$\boldsymbol{\beta}_{v,i} = \tanh(\boldsymbol{W}_{\beta,i}[d_v || \boldsymbol{r}_i^{d_v}] + \boldsymbol{b}_{\beta,i}). \tag{14}$$

Learning with Disturbing and Contrasting. Now we go back to acquire the data pairs $\mathcal{D} = \{(\boldsymbol{c}_{u,i}, \tilde{\boldsymbol{c}}_{u,i})\}$. Since there is no access to the actual pre-restoration and post-restoration data, we need to simulate and self-supervise the restoration module based on the available observed data. Compared to head/tail nodes, the body nodes with median degrees can be assumed that the observed neighborhood is relatively cleaner. Thus, we can corrupt the local topology of body nodes w to imitate tail or head nodes by randomly dropping or adding neighbors from/to w and get Γ disturbed versions of neighborhoods $\boldsymbol{c}_{w_j,i}, j = 1, ..., \Gamma$. We treat them as observed factors, and the originally observed neighborhood $\boldsymbol{c}_{w,i}$ as pseudo ground truth. Notice that the difference between this corruption and the GSL methods is that we do not learn how to drop or add edges, and the altering is localized to the node rather than the whole adjacency matrix.

In each epoch, we sample n_k body nodes with all disturbed versions to form $\mathcal{D}$ for the computation of task kernels. In addition, we sample n_{con} nodes to guide the optimization of the restorer, using the contrastive loss to force the output of disturbed views to be closed to the ground truth:

$$\mathcal{L}_{con} = -\sum_{w}^{n_{con}} \sum_{j=1}^{\Gamma} \log \frac{\mathrm{sim}(\boldsymbol{c}_{w,i}, \tilde{\boldsymbol{c}}_{w_j,i})}{\mathrm{sim}(\boldsymbol{c}_{w,i}, \tilde{\boldsymbol{c}}_{w_j,i}) + \sum_q \mathrm{sim}(\boldsymbol{c}_{w_q,i}, \tilde{\boldsymbol{c}}_{w_j,i})}, \tag{15}$$

where $\mathrm{sim}(\cdot,\cdot) = \exp(\cos(\cdot,\cdot)/\tau_3)$, τ_3 is the temperature hyper-parameter and q is the number of negative samples.

4.2.3 Label-Informed Restorer

Although degrees are informative clues for restoration, we need **augmentation from the target labels** to make sure that such a process accords with the prediction task. Specifically, we use a learnable vector pool $\{\boldsymbol{h}_{c,i}\}_{c=1}^{C}$ to represent the information of labels. We first calculate the relevance of nodes to each label with t-distribution:

$$\delta_{v,c,i} = \frac{(1 + ||\tilde{\boldsymbol{c}}_{v,i} - \boldsymbol{h}_{c,i}||^2/\tau_4)^{-\frac{\tau_4+1}{2}}}{\sum_{c'}(1 + ||\tilde{\boldsymbol{c}}_{v,i} - \boldsymbol{h}_{c',i}||^2/\tau_4)^{-\frac{\tau_4+1}{2}}}, \tag{16}$$

where τ_4 is the hyper-parameter to control the degree of freedom of t-distribution. Based on the relevance, knowledge are fetched from

the pool, and fused to the observed factors:

$$\hat{\boldsymbol{c}}_{v,i} = \alpha\tilde{\boldsymbol{c}}_{v,i} + (1-\alpha)\sum_{c}\delta_{v,c,i}\boldsymbol{h}_{c,i}, \tag{17}$$

where α is a parameter to control the impact from labels. This approach has dual benefits. For tail nodes, we complement missing information from the pool, while for head nodes, the common information stored in the pool can well compensate for the lack of population.

To guide the optimization of this restorer, we force the relevance $\boldsymbol{\delta}$ to be closed to the ground-true labels on the training set $\boldsymbol{Y}_L$:

$$\mathcal{L}_{clu,sup} = D_{KL}(\boldsymbol{\delta}_i||\boldsymbol{Y}_L). \tag{18}$$

On top of that, to adjust to circumstances where limited labels are available, an unsupervised clustering loss [22] is introduced for augmentation, which is calculated on the whole batch:

$$\mathcal{L}_{clu,unsup} = D_{KL}(\boldsymbol{\delta}_i||\boldsymbol{D}_i), \tag{19}$$

$$\mathcal{L}_{clu} = \mathcal{L}_{clu,sup} + \mathcal{L}_{clu,unsup}, \tag{20}$$

where each item in $\boldsymbol{D}_i$ is

$$D_{v,c,i} = \frac{\delta_{v,c,i}^2/\sum_{v'}\delta_{v',c,i}}{\sum_{c'}((\delta_{v,c',i})^2/\sum_{v'}\delta_{v',c',i})}. \tag{21}$$

Finally, we concatenate the factor representations to the latent variable for each node (restored latent neighborhood):

$$\boldsymbol{m}_v = \sigma([\hat{\boldsymbol{c}}_{v,1}||...||\hat{\boldsymbol{c}}_{v,F}]). \tag{22}$$

4.3 Structured Field Augmented Priori

Now we consider the prior $P_\Theta(\boldsymbol{M}|\mathcal{G})$. The traditional variational inference assumes that each data point follows i.i.d. standard Gaussian Distribution, which is contradictory to the nature of interconnectedness of nodes in graphs. Also, such an assumption treats neighborhood representations separately and prevents them from fully utilizing the information in graphs. Therefore, we propose to introduce a Structured Field Augmented Priori for neighborhood $\boldsymbol{M}$, assuming that $\boldsymbol{M}$ lies in an underlying force field defined on the graph, reflecting the interconnected characteristics of graph structure.

Specifically, we instantiate the structured field as Graph Markov Random Field (GMRF) [16] here. GMRF assumes the joint distribution of all nodes and edges in the graph share the same probability density function of the multivariate Gaussian Distribution $N(\mathbf{0}, \mathcal{K}^{-1})$ and is easy to calculate. Let

$$\mathcal{S} = \mathcal{A} + \eta\mathcal{A}_{knn}, \tag{23}$$

$$\mathcal{K} = \text{Laplacian}(\mathcal{S}), \tag{24}$$

where $\mathcal{A}_{knn}$ is the k-nn graph of node features, k, η are hyper-parameters. With this assumption, the priori of node neighborhoods are correlated to the similarity of node features and topological distance, which contains more information and can result in better restoration.

To efficiently calculate the KL-divergence in the ELBO loss, we follow the implementation in [24]:

$$\begin{aligned}\mathcal{L}_{gmrf} =& D_{KL}(Q_\phi(\boldsymbol{M}|\boldsymbol{Y},\mathcal{G})||P_\Theta(\boldsymbol{M}|\mathcal{G}))\\ =& \text{tr}(\boldsymbol{M}^T\mathcal{K}\boldsymbol{M}) - \frac{1}{2}\log|\boldsymbol{I}+\boldsymbol{M}^T\boldsymbol{M}|.\end{aligned} \tag{25}$$

4.4 Training Paradigm

In this section, we propose a training paradigm adapting to the inherent strucutral characteristics of the long-tail degree distribution. First, due to the good quality of observed neighborhoods of body nodes, they are used to train the Degree-Adaptive Restorers as mentioned before. In addition, we use them for pre-training and get a better initialization of factor extractor and the generative model using the prediction loss $\mathcal{L}_{pred}$. Finally, using all restored nodes containing sufficient information, we train the whole model together, with the following total optimization objective:

$$\mathcal{L} = \mathcal{L}_{pred} + \theta_{gmrf}\mathcal{L}_{gmrf} + \theta_{clu}\mathcal{L}_{clu} + \theta_{con}\mathcal{L}_{con}, \tag{26}$$

where $\theta_{gmrf}, \theta_{clu}, \theta_{con}$ are hyper-paramters.

5 Experiment

5.1 Experimental Setup

5.1.1 Datasets

Our model is evaluated on six public benchmarks from various domains for node classification: citation networks Cora and Pubmed [23], e-commercial networks Photo and Computer [17], a co-authorship network Coauthor-CS [17] and a social network Facebook [15]. The statistics of them are summarized in Table 1. The degree distribution of all benchmarks exhibits a characteristic long tail.

5.1.2 Baselines

We compare our model with the following models: (1) GCN [8] and GraphSAGE [4] are traditional GNNs, aggregating neighborhood features without discrimination. (2) GAT [19], GPR-GNN [2], DisenGCN [14], S2GC [27] adjust the weights of existing edges during aggregation. (3) RawlsGCN [7], Tail-GNN [12], LAGNN [10], ColdBrew [26] emphasize to improve performance on tail nodes; DegFairGNN [13] further consider structural fairness. (4) Pro-GNN [5], GEN [20], WSGNN [9] reduce noise by explicitly refining the graph structure, such as generating or removing edges.

5.1.3 Settings

For Cora and Pubmed, we follow the same train/val/test split ratio as [23] while others follow [17]. For all methods, we take a two-layer network structure and set the hidden dimension as 64. For Laner-GNN, we set the number of latent factors $F = 8$, each of which contains an 8-dimensional vector. For the number of pre-training epochs, we set it to 10 for Cora, Pubmed and Coauthor and 50 for the others. The thresholds to divide head/body/tail nodes are decided by the 95% percentile and mean value of the degree distribution. We tune the hyper-parameters on the validation set, and run all models for 5 times and report their average performance on the test set.

5.2 Main Results

In this section, we evaluate the semi-supervised node classification performance of Laner-GNN against the state-of-the-art baselines. Table 2 shows that our Laner-GNN achieves superior performance on all datasets. Firstly, traditional GNNs ignore the problem of the long-tail distribution of degrees, so the performance is poor. Secondly, GNNs for tail node embedding mostly only consider the problem of

Table 1. Statistics of datasets.

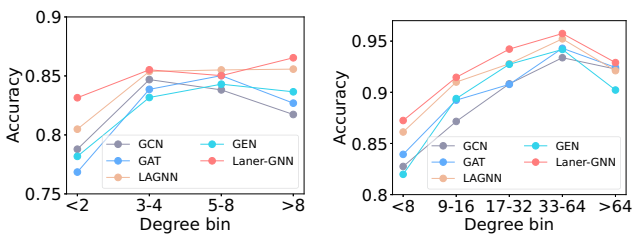
Dataset	Cora	Pubmed	Photo	Computer	CoauthorCS	Facebook
# Nodes	2708	19717	7650	13572	18333	22470
# Edges	5278	44327	119082	245861	81894	171002
# Features	1433	500	745	767	6805	4714
# Classes	7	3	8	10	15	4
Avg. degree	3.90	4.50	31.13	35.76	8.93	15.22
Std of degree	5.23	7.43	47.27	70.31	9.11	26.41
95% perc. of degree	9	18	87	101	26	55

Table 2. Accuracy (%±std.) on node classification task. Boldfaced letters mark the best results and underlined ones are the runners-up. OOM denotes out-of-memory on an NVIDIA GeForce RTX 3090 (24GB).

Model	Cora	Pubmed	Photo	Computer	Coauthor-CS	Facebook
GCN	81.88±0.70	78.72±0.32	89.91±0.78	78.39±1.71	88.90±0.89	69.29±2.48
GraphSAGE	82.50±0.49	78.68±0.45	89.41±0.90	78.69±1.99	91.06±0.60	67.47±2.67
GAT	82.24±0.47	77.66±0.27	90.82±0.91	76.04±1.90	91.29±0.40	72.41±2.49
DisenGCN	81.96±0.47	78.80±0.84	89.95±0.91	80.16±1.89	90.98±0.35	72.31±2.95
S2GC	82.62±0.04	79.97±0.05	89.91±1.49	78.15±1.04	91.63±0.54	72.24±2.24
GPR-GNN	83.34±0.64	78.96±0.43	89.91±0.44	78.98±2.39	87.24±0.62	72.60±1.87
RawlsGCN	78.52±0.20	75.30±0.23	88.40±0.19	79.67±1.28	91.16±0.66	68.35±1.34
Tail-GNN	81.52±0.26	77.86±0.94	87.77±1.56	78.52±1.34	90.44±0.82	70.15±1.32
LAGNN	83.70±0.29	80.28±0.37	91.81±0.38	79.87±2.30	91.00±0.65	71.67±2.20
ColdBrew	80.16±0.39	77.84±0.75	90.72±0.31	81.24±0.60	90.24±0.05	69.44±0.45
DegFairGNN	78.28±0.72	76.84±0.43	88.74±0.66	79.23±1.74	88.59±0.96	70.47±1.38
Pro-GNN	80.66±0.68	78.01±0.25	87.28±1.37	73.15±1.95	90.57±0.60	OOM
GEN	82.18±0.18	79.06±0.16	89.40±0.33	81.42±1.93	92.26±0.63	70.10±1.22
WSGNN	83.28±0.49	OOM	91.09±0.95	OOM	OOM	OOM
Laner-GNN	**83.94±0.46**	**81.20±0.67**	**92.32±0.31**	**81.88±0.76**	**92.88±0.52**	**72.76±1.12**

missing information. However, our Laner-GNN focuses on the problems existing in all nodes of the long-tail distribution graph, taking into account the missing information of tail nodes and noise and limited samples about head nodes with full use of the graph structure information. Finally, GSL methods, explicitly adjusting the graph structure, suffer from poor space complexity and are out of memory in relatively large datasets like Pubmed, Computer, Coauthor-CS and Facebook. On the contrary, Laner-GNN uses the neighborhood factors to perform implicit restoration and thus achieve even better performance. In summary, the results significantly demonstrate the effectiveness of our proposed framework and our motivation.

5.3 Model Performance in Different Degree Bins



(a) Cora (b) Photo

Figure 3. Comparison of models in different degree bins.

We divide the nodes on Cora and Photo into bins according to their degrees, and report model performance on different bins to demonstrate our superiority. The results are shown in Figure 3. By adjusting aggregation weights, GAT can somehow reduce the noise in head nodes, but is helpless to the lack of information in tail bins. For LAGNN, the plugged-in features add more information to neighborhoods. However, compared to Laner-GNN, it cannot fully utilize information from the prediction task and underlying the graph, and fail to achieve better performance. Our Laner-GNN completes the latent neighbor structure for tail nodes and significantly outperforms all baselines. Also, after removing the noise and obtaining supplements for the limited samples from the label pool, the performance on head nodes is improved. Notice that although the neighborhood of the middle nodes is relatively high in quality, it is not perfect. Through neighborhood restoration, their quality is improved to some extent. Therefore, our model indeed has the ability to handle the structural problem existing in long-tail degree distribution.

5.4 Model Analysis

5.4.1 Ablation Studies

In this section, we conduct ablation studies on Cora and Photo. The results are shown in Table 3.

Table 3. Ablation Studies on Cora and Photo.

Variants	Cora	Photo
Laner-GNN	**83.94**	**92.32**
w/o Degree-Adaptive Restorer	81.23	91.70
w/o Degree Clues	82.19	91.71
w/o Label-Informed Restorer($\alpha = 0$)	83.30	90.98
w/o GMRF	80.94	89.58
w/o KNN ($\eta = 0$)	82.10	91.10
w/o Pre-train	80.41	90.57

Effects of Degree-Adaptive Restorer. To validate the effectiveness of Degree-Adaptive Restorer, we compare the full model with

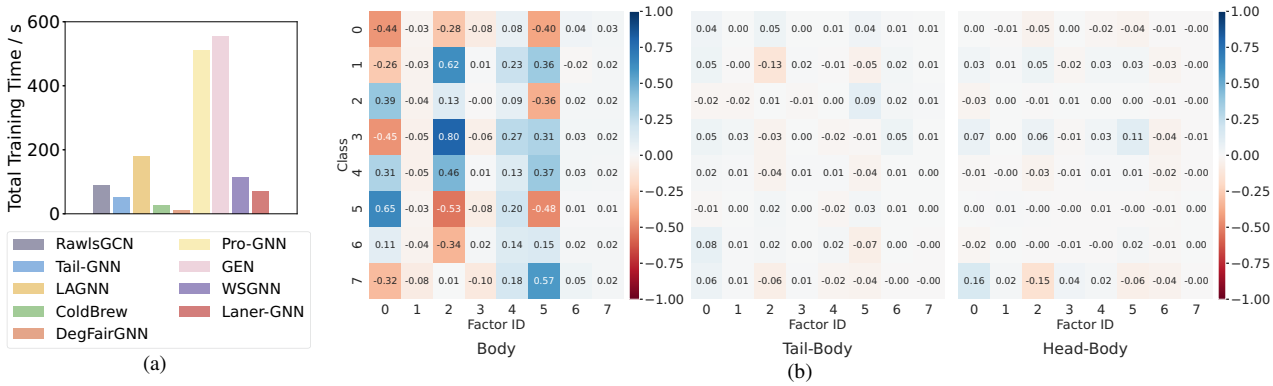


Figure 4. (a) Total training time until convergence of different models on Cora. (b) Visualization of latent structural factors learned by Laner-GNN on Photo, where the leftmost one shows the mean factors of body nodes in each classes. The two figures on the right respectively show the differences of tail and head to the body.

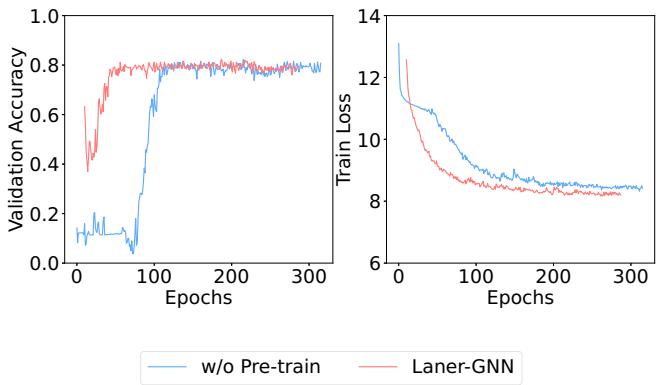


Figure 5. Performance on Cora w.r.t. epochs, which demonstrates faster convergence and better model performance brought by pre-training on body nodes. For Laner-GNN, the start points of lines are at the number of pre-training epochs

variants that remove it (w/o Degree-Adaptive Restorer) or do not use degree information in the restorer (w/o Degree Clues). As we can see, the two variants underperform full Laner-GNN in both datasets, which demonstrates that the component leads to better representation capacity. Also, degrees are important clues to determine how far the observed neighborhoods are from the ground truth.

Effects of Label-Informed Restorer. We compare Laner-GNN with the variant without Label-Informed Restorer (w/o Label-Informed Restorer). Because labels are elements directly associated with the prediction task, incorporating them into the restorer can uplift the quality of local neighborhood restoration.

Effects of Structure Field Augmented Priori. We conduct ablation experiments to prove the improvement brought by the structured field, with a variant removing such constraint (w/o GMRF) and the one without $\mathcal{A}_{knn}$ in the priori (w/o KNN). The variant w/o GMRF has a significant drop compared to the full counterpart, concluding that GMRF better utilizes the nature of linking in graphs and benefits local neighborhood restoration. Moreover, feature distribution included in $\mathcal{A}_{knn}$ is vital knowledge in the priori to ensure the quality of restoration. Without the distribution constraints imposed by the priori, the model fails to fully fulfill its intended purpose, as the restoration module cannot capture the characteristics of the graph data.

Effects of Pre-training with Body Nodes. Compared to the variant directly training on the whole training set (w/o Pre-train), our Laner-GNN first warming up on body nodes converges faster and has better performance as shown in Figure 5(a). We conjecture that the relatively complete neighborhood structure of body nodes can help generate a better initialization of the model, and thus the learning progress of the restorers can be accelerated.

5.4.2 *Runtime Analysis*

We test Laner-GNN training efficiency on Cora and the results are shown in Figure 4(a). The time complexity of Laner-GNN is comparable to most existing GNNs focusing on low-degree nodes. Notice that the training time of GSL models, especially Pro-GNN and GEN, is more than five times that of our model. The reason is that they need to search for the optimal graph structure by explicitly learning to add or remove edges, while Laner-GNN restores local neighborhoods in latent vector spaces.

5.5 *Visualization*

To have a deeper insight into the behaviors of Laner-GNN, we visualize the mean latent factors of each class in Photo (see Figure 5(b). On one hand, Laner-GNN can learn discriminable composition of structural factors for each class, so the idea of latent neighborhood restoration is reasonable. On the other hand, the gaps between tail/head and body are narrow, demonstrating the ability of Laner-GNN to find back missing information or reduce noise, and thus it can achieve good latent neighborhood restoration.

6 Conclusion

In this paper, we introduce Laner-GNN, a method that improves node property prediction by restoring the latent neighborhood of nodes to compensate for structural imperfections. It consists of a Degree-Adaptive Restorer, which considers the influence of node degrees on restoration, and a Label-Informed Restorer and Structured Field Augmented Priori, which leverage graph information for better restoration. Our experiments on real-world datasets validate the effectiveness of our model and its ability to handle nodes with varying degrees in the long-tail distribution.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No. 62276006).

References

[1] Z. Chen, T. Xiao, and K. Kuang. Ba-gnn: On learning bias-aware graph neural network. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 3012–3024. IEEE, 2022.

[2] E. Chien, J. Peng, P. Li, and O. Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021.

[3] J. Gasteiger, A. Bojchevski, and S. Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*, 2018.

[4] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[5] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 66–74, 2020.

[6] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Mach. Learn.*, 37(2):183–233, 1999. doi: 10.1023/A:1007665907178. URL https://doi.org/10.1023/A:1007665907178.

[7] J. Kang, Y. Zhu, Y. Xia, J. Luo, and H. Tong. Rawlsgcn: Towards rawlsian difference principle on graph convolutional network. In *Proceedings of the ACM Web Conference 2022*, pages 1214–1225, 2022.

[8] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[9] D. Lao, X. Yang, Q. Wu, and J. Yan. Variational inference for training graph neural networks in low-data regime through joint structure-label estimation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 824–834, 2022.

[10] S. Liu, R. Ying, H. Dong, L. Li, T. Xu, Y. Rong, P. Zhao, J. Huang, and D. Wu. Local augmentation for graph neural networks. In *International Conference on Machine Learning*, pages 14054–14072. PMLR, 2022.

[11] Z. Liu, W. Zhang, Y. Fang, X. Zhang, and S. C. Hoi. Towards locality-aware meta-learning of tail node embeddings on networks. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 975–984, 2020.

[12] Z. Liu, T.-K. Nguyen, and Y. Fang. Tail-gnn: Tail-node graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1109–1119, 2021.

[13] Z. Liu, T.-K. Nguyen, and Y. Fang. On generalized degree fairness in graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 4525–4533, 2023.

[14] J. Ma, P. Cui, K. Kuang, X. Wang, and W. Zhu. Disentangled graph convolutional networks. In *International conference on machine learning*, pages 4212–4221. PMLR, 2019.

[15] B. Rozemberczki, C. Allen, and R. Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

[16] H. Rue and L. Held. *Gaussian Markov random fields: theory and applications*. CRC press, 2005.

[17] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[18] X. Tang, H. Yao, Y. Sun, Y. Wang, J. Tang, C. Aggarwal, P. Mitra, and S. Wang. Investigating and mitigating degree-related biases in graph convoltuional networks. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1435–1444, 2020.

[19] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *stat*, 1050:20, 2017.

[20] R. Wang, S. Mou, X. Wang, W. Xiao, Q. Ju, C. Shi, and X. Xie. Graph structure estimation neural networks. In *Proceedings of the Web Conference 2021*, pages 342–353, 2021.

[21] J. Wu, J. He, and J. Xu. Net: Degree-specific graph neural networks for node and graph classification. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 406–415, 2019.

[22] J. Xie, R. B. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In M. Balcan and K. Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 478–487. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/xieb16.html.

[23] Z. Yang, W. Cohen, and R. Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[24] J. Yoo, H. Jeon, J. Jung, and U. Kang. Accurate node feature estimation with structured variational graph autoencoder. In A. Zhang and H. Rangwala, editors, *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 2336–2346. ACM, 2022. doi: 10.1145/3534678.3539337. URL https://doi.org/10.1145/3534678.3539337.

[25] S. Yun, K. Kim, K. Yoon, and C. Park. Lte4g: Long-tail experts for graph neural networks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 2434–2443, 2022.

[26] W. Zheng, E. W. Huang, N. Rao, S. Katariya, Z. Wang, and K. Subbian. Cold brew: Distilling graph node representations with incomplete or missing neighborhoods. In *International Conference on Learning Representations*, 2021.

[27] H. Zhu and P. Koniusz. Simple spectral graph convolution. In *International conference on learning representations*, 2020.