Curriculum-Enhanced Residual Soft An-Isotropic Normalization for Over-smoothness in Deep GNNs

Jin Li $^{1,\,2}$, Qirong Zhang 1 , Shuling Xu 1 , Xinlong Chen 1 , Longkun Guo $^{1,\,3}$, Yang-Geng Fu $^{1^*}$

¹College of Computer and Data Science, Fuzhou University, Fuzhou, China

²AI Thrust, Information Hub, HKUST (Guangzhou), Guangzhou, China

³Shandong Provincial Key Laboratory of Computer Networks,

Shandong Fundamental Research Center for Computer Science, Jinan, China

{jslijin2015, shuling xu}@outlook.com, {jiusizhang1, fjxinlong, longkun.guo}@gmail.com, fu@fzu.edu.cn

Abstract

Despite Graph neural networks' significant performance gain over many classic techniques in various graph-related downstream tasks, their successes are restricted in shallow models due to over-smoothness and the difficulties of optimizations among many other issues. In this paper, to alleviate the over-smoothing issue, we propose a soft graph normalization method to preserve the diversities of node embeddings and prevent indiscrimination due to possible over-closeness. Combined with residual connections, we analyze the reason why the method can effectively capture the knowledge in both input graph structures and node features even with deep networks. Additionally, inspired by Curriculum Learning that learns easy examples before the hard ones, we propose a novel labelsmoothing-based learning framework to enhance the optimization of deep GNNs, which iteratively smooths labels in an auxiliary graph and constructs many gradual non-smooth tasks for extracting increasingly complex knowledge and gradually discriminating nodes from coarse to fine. The method arguably reduces the risk of overfitting and generalizes better results. Finally, extensive experiments are carried out to demonstrate the effectiveness and potential of the proposed model and learning framework through comparison with twelve existing baselines including the state-of-the-art methods on twelve real-world node classification benchmarks.

1 Introduction

Graph neural networks (GNNs) (Wu et al. 2020) are widely used state-of-the-art techniques to solve many tasks on graph (*e.g.*, semi-supervised node classification (Feng et al. 2020), link prediction (Yun et al. 2021), graph classification (Xie et al. 2022), and community detection (Liu et al. 2021), etc). Also, GNNs have achieved outstanding results recently in many domains including texts (Fei, Zhang, and Zhou 2021), images (Guan et al. 2022), traffic (Choi et al. 2022), molecule (Han et al. 2022), and even electroencephalogram (*i.e.*, EEG) (Demir et al. 2021) compared to classic methods (*e.g.*, CNN and RNN). They are previously derived in spectral domain based on the eigen-decomposition of graph Laplacian (*e.g.*, Spectral-CNN (Bruna et al. 2014) and ChebNet (Defferrard, Bresson, and Vandergheynst 2016)). Graph Convolutional

Network (GCN) (Kipf and Welling 2017) is proposed to accelerate it via a linear approximation of universal filters on graph signals and also give an intuitive interpretation on spatial domain, *i.e.*, *message passing*, which is further developed by SGC (Wu et al. 2019), GAT (Veličković et al. 2018), GIN (Xu et al. 2019), and MPNN (Gilmer et al. 2017).

More recently, some deep GNNs (Chen et al. 2020; Li et al. 2019) are proposed to further improve the expressive power of GNNs in light of successes of other deep neural networks, *e.g.*, CNN and RNN. But unfortunately, GNNs are not easy to go deep and often suffer from severe performance degradation due to, *e.g.*, over-smoothness (Liu, Gao, and Ji 2020), difficulty in optimization (Yang et al. 2020), memory limitation (Li et al. 2021), time consumption (Li et al. 2021), and over-squashing (Topping et al. 2022). In this paper, we mainly focus on the first two problems, *i.e.*, improvements will be devised with respect to the following two aspects: 1) structures; and 2) the learning process.

In order to alleviate over-smoothness, various kinds of techniques (Chen et al. 2022) are proposed to prevent overcloseness of node embeddings or reduce the extent of aggregations including graph normalization (Zhou et al. 2021), residual connections (Chen et al. 2020), and random dropping (Huang et al. 2020), etc. Existing graph normalization techniques directly operate with norms, means, variances or distances of embeddings, and can effectively reduce the over-closeness. However, it's still unclear what or how much knowledge they can preserve in deep layers via only these numeric-related operations. Some residual connections methods are proven to keep useful feature semantics as GNNs go deep, but they may risk missing structural knowledge, *e.g.*, GCNII (Chen et al. 2020). Random dropping approaches are devised mainly for regularization without theoretical guarantees for alleviating information loss in over-smoothness and seldom perform competitively compared to other stateof-the-arts.

Thus for structures, we propose *R-SoftGraphAIN*, a residual connections-based soft graph normalization layer, which can be viewed as a combination of a *novel* soft graph normalization operation and two *improved* residual connections. Compared to Pairnorm (Zhao and Akoglu 2020), it can normalize embeddings in an an-isotropic manner instead of equally treating all nodes. Compared to GCNII, it can be

^{*}Corresponding author

This paper has been accepted at the 38th AAAI Conference on Artificial Intelligence (AAAI-24).

shown to preserve relatively high frequent structural knowledge instead of over-emphasizing features and can be viewed as a generalization of GCNII. In Sec. 3.1, theoretical analysis is carried out to show its following characteristics as the depth approaches infinity: 1) The mean distance of pairwise node embeddings will be kept nearly constant similar to Pairnorm; 2) The diversities of these signals are maximized; 3) It tends to extract the most d lowest frequent components of structural knowledge; 4) It never forgets the original feature's information. Experimental evaluations prove its effectiveness due to significant performance gain compared to others.

On the other hand, in order to ease the optimization of deep GNNs, we borrow ideas from Curriculum Learning (CL) (Wang, Chen, and Zhu 2022; Soviany et al. 2022). In CL, models are encouraged to first learn from easy examples and then examples with gradually increased difficulty (Bengio et al. 2009). The idea has been generalized to devise better curriculum applied to various scenarios in many domains including texts and images via, *e.g.*, designing multifarious tasks with increasing difficulties (Caubrière et al. 2019), gradually unleashing expressive powers of models (Sinha, Garg, and Larochelle 2020), or defining *pacing* functions to decide to which extent to learn from a task (Hacohen and Weinshall 2019). However, graphs contain specific structures and semisupervised learning has a special setting, which may require a careful curriculum design, but few prior works focus on this (see Sec. 2). Therefore in this paper, we give a simple yet effective label-smooth-based example called *SmoothCurriculum*. More specifically, we first employ *label propagation* to estimate unknown labels, and all labels are iteratively smoothed in an auxiliary graph built via a pre-trained teacher model in order to construct gradual non-smooth tasks. From the analysis in Sec. 3.2, this learning framework encourages graph encoders to extract increasingly complex knowledge and learn to gradually discriminate nodes from coarse to fine¹, which intuitively emphasizes relatively global knowledge and alleviates possible label noise, thus reducing the risk of overfitting and generalizing better. Obvious performance improvements across various real-world datasets reveal the potential of this simple framework.

The contribution of this paper can be summarized as follows:

- We propose a residual connections-favored soft graph normalization structure (called *R-SoftGraphAIN*) for preserving knowledge from both input graph topology and features and retaining the diversities of node embeddings in deep layers to consequently alleviate over-smoothness.
- We design a novel label-smoothing-based curriculum learning framework (called *SmoothCurriculum*) to ease the difficulty of optimization of deep GNNs and better their generalization via implicit coarse-to-fine node discrimination.
- Extensive experiments were carried out to demonstrate the effectiveness and potential of our method compared to twelve existing baselines including state-of-the-arts.

2 Preliminaries and Related work

Notation Let $G = (V, E)$ be an undirected graph with node set V and edge set E, where $n = |V|, m = |E|$ represent the numbers of its nodes and edges respectively. We denote by $A \in \{0,1\}^{n \times n}$ and $X \in \mathbb{R}^{n \times d}$ its adjacency and feature matrix where node i has feature $x_i = X_{i,:} \in \mathbb{R}^d$ and a ground-truth label $y_i = Y_i \in \mathbb{N}$. Define $I_n \in \mathbb{R}^{n \times n}$ as an identity matrix, $\mathbf{0}_n, \mathbf{1}_n \in \mathbb{R}^{n \times 1}$ as all-zero/one vectors.

GCN and SGC GCN can be formulated as follows:

$$
H^{(0)} = X, \quad H^{(l+1)} = \sigma\left(\hat{A}H^{(l)}W^{(l)}\right) \in \mathbb{R}^{n \times d}.\tag{1}
$$

SGC simplifies GCN by dropping its non-linear activation functions and its forward pass can be described as follows:

$$
H^{(l)} = \hat{A}^l X W \in \mathbb{R}^{n \times d}, \quad \forall l \in [0, L), \tag{2}
$$

where L is the number of layers and $\sigma(\cdot)$ denotes a non-linear activation function (*e.g.*, ReLU, or Softmax for the last layer). $\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$ is the symmetrically normalized matrix of $\tilde{A} = A + I_n$ and diagonal matrix $\tilde{D}_{i,i} = \sum_{j=1}^n \tilde{A}_{i,j}$. Sometimes, the probability transition matrix $\hat{A}_{rw} = \tilde{D}^{-1} \tilde{A}$ is employed for aggregating. $H^{(l)} \in \mathbb{R}^{n \times d}$ and $W, W^{(l)} \in$ $\mathbb{R}^{d \times d}$ represent the embeddings and the trainable parameters.

Deep GNNs and Over-smoothness Majority of GNN variants are shallow networks (*e.g.*, no more than three layers), thus restricting their expressive power and limiting distant message passing. Some prior works make efforts to deepen GNNs via modifications or tricks which can be categorized into three classes: residual connections, graph normalization (*e.g.*, Pairnorm (Zhao and Akoglu 2020), Nodenorm (Zhou et al. 2021), Meannorm (Yang et al. 2020)), and random dropping (*e.g.*, DropEdge (Rong et al. 2020) and DropNode (Huang et al. 2020)). Residual connections contain common residual connection (from last layer) (Li et al. 2019), initial connection (from the first layer) (Chen et al. 2020), dense connection (from every layer) (Liu, Gao, and Ji 2020; Luan et al. 2019), and jump connection (from every layer to the last layer only) (Xu et al. 2018). See supplementary materials or (Chen et al. 2022) for some more related work or a more detailed survey. Our method appropriately combines *improved* residual connections and a *novel* soft graph normalization enabling effective feature and structural knowledge extraction and preservation even with a sufficiently large depth.

Curriculum Learning CL has become a popular kind of training strategies for networks in many applications including texts (Liu et al. 2020), images (Zhou, Wang, and Bilmes 2020), speeches (Wang et al. 2020), reinforcement learning (Narvekar et al. 2020), etc. The basic idea is to give examples from easy to hard, and has been developed a lot (Wang, Chen, and Zhu 2022), but few are designed specifically for graphrated tasks (Wang et al. 2021; Chu et al. 2021). But note that besides over-smoothness, another non-negligible cause hindering deep GNNs is just the difficult optimization. Thus in this work, we hope to ease it via a novel curriculum learning framework based on iterative label smoothing on an auxiliary graph. Here we define a *curriculum* as a *task sequence*

 ${}^{1}E.g.,$ for collecting residential information of a person, first the information of the country and then the city he lives in will be collected.

 $\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_{n_T}$ with gradually increasing difficulties where $\mathcal{T}_i = \left(D^{(i)}, f_\theta^{(i)}, \ell^{(i)}(\cdot), p^{(i)}\right)$ denotes a task meaning that a model $f_{\theta}^{(i)}$ $\theta_{\theta}^{(i)}$ learns from the data $D^{(i)}$ with a loss function $\ell^{(i)}(\cdot)$ and *pacing* strategy $p^{(i)}$ (*e.g.*, the time it spends). $f_{\theta}^{(i)}$ θ is some modified or restricted version of f_θ .

3 SmoothCurriculum-improved R-SoftGraphAIN

In this section, we propose a novel model for alleviating the over-smoothness of graph neural networks by incorporating two main ingredients (*i.e.*, *R-SoftGraphAIN* for GNN structures, and the adaptive curriculum design). Although treated as a whole for reporting their performance in Sec. 4, we individually introduce each ingredient in the following for better clarity and briefness.

3.1 R-SoftGraphAIN

We will describe our normalization method and show how it can be improved via residual and initial connections in the following paragraphs.

Spectral Analysis on Over-smoothness Over-closeness (*i.e.*, embeddings are too close) is the external manifestation of over-smoothness leading to indiscrimination via a classifier. But it's just a direct cause instead of an essential reason analyzed by priors in the spectral domain on SGC as follows:

$$
h^{(L)} = \hat{A}^L x = U\Lambda^L U^T x = \sum_{i=1}^n \lambda_i^L u_i u_i^T x,
$$

\n
$$
\lim_{L \to \infty} h^{(L)} = u_1 u_1^T x = \tilde{D}^{\frac{1}{2}} \mathbf{1}_n \mathbf{1}_n^T \tilde{D}^{\frac{1}{2}} x \propto \tilde{D}^{\frac{1}{2}} \mathbf{1}_n,
$$
\n(3)

where x is a signal, $\hat{A} = U \Lambda U^T$ contains decreasing eigenvalues $\{\lambda_i\}$ with respective eigenvectors $\{u_i\}$ and $\lambda_1 = 1$ λ_2 . The conclusion on GCN is similar with a very different non-trivial analysis (Oono and Suzuki 2020). The analysis tells us deep GNNs tend to: 1) hardly keep important structural knowledge, 2) gradually forget the semantics contained in features, thus essentially leading to over-smoothness.

Pairnorm (Zhao and Akoglu 2020) attempts to numerically solve over-closeness via direct manipulation on mean distance formulated as: $H^{(l+1)} = C\sqrt{n} \cdot H'/\Vert H' \Vert_F$, $H' =$ $(I_n - 1/n \cdot \mathbf{1}_n \mathbf{1}_n^T) \hat{A} H^{(l)}$ with a constant $C > 0$. We conjecture its performance is limited even getting rid of indiscrimination due to normalizing embeddings: 1) only elementwisely and isotropically without modeling the complex relationships between nodes and between signals; 2) only numerically without interpretable knowledge preservation. These shortcomings motivate our *GraphAIN* considering normalization an-isotropically and in a distribution/knowledge-aware manner.

Soft Graph An-Isotropic Normalization As mentioned above, we propose *GraphAIN* to deal with the comprehensive distribution and keep diverse meaningful knowledge, which can be described as follows:

$$
H_t = B_{t-1} \left(B_{t-1}^T B_{t-1} \right)^{-\frac{1}{2}} \in \mathbb{R}^{n \times d}, \quad \forall \ t \ge 1,
$$

\n
$$
B_{t-1} = T \cdot \hat{A} H_{t-1} \in \mathbb{R}^{n \times d}, \quad H_0 = X,
$$
 (4)

where $H_t = H^{(t)}$ and X denote the embedding matrix of the t-th layer and the original features. And $T = \overline{I_n} - \frac{1}{n} \mathbf{1}_n \mathbf{1}_n^T \in$ $\mathbb{R}^{n \times n}$ represents a centering operator and $P^{1/2}$ refers to the square root of a positive matrix P . The following statement theoretically justifies our idea that we are normalizing the covariance matrix of H instead of directly normalizing the embedding for an individual node or signal independently:

Theorem 1.
$$
\forall t \ge 1
$$
, *GraphAIN satisfies that:*
\n1) $\mathbf{1}_n^T B_t = \mathbf{1}_n^T H_t = \mathbf{0}_n$; 2) $H_t^T H_t = I_d$;
\n3) $TH_t = H_t$; 4) $B_t = \overline{A}H_t$;
\nwhere $\overline{A} = T\hat{A}T$ denotes a doubly centered version of \hat{A} .

Next theorem theoretically analyzes *GraphAIN* from an optimization perspective and gives a deep understanding of combination of normalization and aggregations in GNNs:

Theorem 2. *GraphAIN can be viewed as an iterative process in order to solve the following restricted optimization problem via Projected Gradient Ascent method:*

$$
\max_{H} f(H) = \frac{1}{2} \cdot \text{tr}\left(H^{T} \bar{A} H\right), \quad s.t. \ H^{T} H = I_{d}, \tag{5}
$$

where H is initialized to input graph signals $X \in \mathbb{R}^{n \times d}$ and *set the step size* $\eta = 1$ *.*

All proofs can be found in supplementary materials. This theorem reveals what *GraphAIN* can learn. In fact, from another point of view, the optimal solution to this problem can be obtained via *Lagrange multiplier* method as follows:

$$
\mathcal{L}(H, \Lambda_L) = \text{tr}\left(H^T \bar{A} H\right) - \text{tr}\left(\Lambda_L \left(H^T H - I_d\right)\right), \quad (6)
$$

where $\mathcal L$ and Λ_L are the Lagrange function and multipliers. Let $\partial \mathcal{L}/\partial H = 0$, we get $\overline{A}H = \Lambda_L H$, which means that H tends to the eigenvectors corresponding to the top-d eigenvalues of \overline{A} with sufficient steps. Thus similar to Spectral Clustering, it can capture essential structural knowledge due to the similarity between \bar{A} and \hat{A} . In addition to spectral interpretations, we give some intuitions in spatial domain: 1) it can easily solve over-closeness, due to the facts that $||H||_F^2 = d$ and $\sum_i \sum_j \|H_{i,:} - H_{j,:}\|_2^2 = 2n \cdot \sum_i \|H_{i,:}\|_2^2 - 2 \cdot \|\sum_i H_{i,:}\|_2^2 =$ $2n \cdot d$ meaning the average pairwise distance is kept completely constant similar to Pairnorm; 2) the spatial variance in any direction is normalized to 1 for maximally preserving the diversity of knowledge in a circular distribution.

However, it still suffers from performance degradation due to the following four potential drawbacks: 1) too absolute; 2) numerical instability; 3) high time complexity; 4) risk of forgetting original features during iterations. The first three issues can be relieved via a soft version (*i.e.*, *SoftGraphAIN*):

$$
H_t \approx B_{t-1} \left[a \cdot U_{d_0} \Lambda_{d_0}^{-\frac{1}{2} \cdot b} U_{d_0}^T + (1 - a) \cdot I_d \right], \quad (7)
$$

where $B_{t-1}^T B_{t-1} \approx U_{d_0} \Lambda_{d_0} U_{d_0}^T$ is the d_0 -truncated SVD calculating only the top- $d_0 \le d$ eigenvectors and eigenvalues contained in $\overline{U}_{d_0}, \Lambda_{d_0} \in \mathbb{R}^{\overline{d} \times d}$, and $a, b \in [0, 1]$ are another hyper-parameters controlling the extent of normalizing. Formally, they transform the singular values $S \approx S_{d_0} = \Lambda_{d_0}^{1/2} \in$ $\mathbb{R}^{d \times d}$ into $(1 - a) \cdot S_{d_0} + a \cdot S_{d_0}^{1-b}$ thus flexibly reducing the absoluteness, possible noise in useless channels, risk of numerical zero-divisions, and empirical time-inefficiency.

Additional Residual Combination *R-SoftGraphAIN* can effectively alleviate the last drawback mentioned above via some residual and initial connections formulated as follows:

$$
B_t = \alpha \cdot T\hat{A}H_t + \beta \cdot H_t + \gamma \cdot X \in \mathbb{R}^{n \times d}, \forall t \ge 1, \quad (8)
$$

where the non-negative hyper-parameters meet $\alpha + \beta + \gamma = 1$. Intuitively, the commonly used residual connections can alleviate gradient-vanishing and the initial ones are expected to constantly supplement some feature information during aggregations in case of oblivion. Moreover, the motivation can be theoretically justified via the following similar theorem:

Theorem 3. *Residual-favored GraphAIN can be viewed as an iterative process in order to solve the following restricted optimization problem via Projected Gradient Ascent method:*

$$
\max_{H} f(H) = \frac{1}{2} \cdot \text{tr}\left(H^{T}\bar{A}H\right) - \frac{1}{2} \cdot \frac{\gamma}{\alpha} \cdot \|H - X\|_{F}^{2} \tag{9}
$$

s.t. $H^{T}H = I_{d},$

where H is initialized to input graph signals $X \in \mathbb{R}^{n \times d}$ and *set the step size* $\eta = \alpha \in [0, 1]$ *.*

This theoretically reveals that *R-SoftGraphAIN* never forgets the original features as GNNs go deep, simultaneously relieving two essential reasons in Sec. 3.1 and thus alleviating over-smoothness. Furthermore, from this we can get some intuitions on the roles of α , β , γ : α allows a sufficiently small step size ensuring better convergence, γ estimates the contribution of features. β can give some freedom to α and γ . In order to further improve its performance, we generalize these connections as *fuzzy* connections. We use Eq. 7 and Eq. 8 to substitute Eq. 8, and replace X in Eq. 8 by H_1 due to possible misalignment in dimensions. A GCN-based implementation of the whole structure is summarized in Algorithm 1 in supplementary materials with a line-by-line description therein.

Relations to Others In this paragraph, we detailedly compare ours with other related methods. SGC suffers from oversmoothness due to both structural and feature knowledge loss. Pairnorm numerically solves over-closeness without interpretable knowledge preservation. Meannorm and Spectral Clustering (SC) can keep the 2-th and lowest d frequent components in structures respectively while keeping little feature information. GCNII proves to be a universal approximator of any function on features, but it ignores structural semantics. Compared to them, ours can keep both features and structural knowledge inheriting both advantages. From another perspective, Pairnorm, Meannorm, and Nodenorm (Zhou et al. 2021) are only element-wise, signal-independent, and nodeindependent, respectively. However, our method considers the comprehensive distributions and effectively models the relationships between nodes and between signals, thus uttermost preserving the diversity of the embeddings. Fig. 3 in supplementary materials shows our superiority, where ours is similar to and even outperform SC while others suffer from over-smoothness to different extents.

3.2 SmoothCurriculum

In this section, we propose a *simple* yet *effective* curriculum learning framework based on *label-smoothing* on an auxiliary graph to ease the hardness of optimizing the proposed

R-SoftGraphAIN. Inspired by Curriculum Learning, the key idea of our framework is first to learn the low-frequent knowledge contained in labels before the high-frequent ones, and then to employ an easy-to-hard learning process that favors a better generalization. The framework will be described in detail regarding several of its important modules (*e.g.*, label estimation and smoothing, graph construction, and curriculum designs), intuitions, and interpretations.

Label Estimation and Auxiliary Graph Deep GNNs are powerful yet risk overfitting due to limited labeled data, especially in the semi-supervised setting. Thus we hope to enlarge the training set via label estimation. One of the most commonly used classic techniques is *Label Propagation*, whose iterative process is: $f \leftarrow P \cdot f, f_L \leftarrow Y_L$ initializing $f_L = Y_L, f_U = 0$. Furthermore, its limit can be formulated as: $Y_U^{(e)} = \lim f_U = (I - P_{UU})^{-1} P_{UL} Y_L$, where U, L represent unlabeled and labeled node sets, $P = D^{-1}A$, P_{UL} is a sub-matrix of P respect to lines U and columns L , and $Y_L, Y_U^{(e)}$ are the known and estimated labels. But it suffers from two shortcomings: 1) impossible propagation due to possible disconnectivity; 2) impractical matrix inversion with a large |U|. Thus, we estimate Y_U *implicitly* via a teacher model $f_t(\cdot)$ pre-trained on the labeled data $D_o = (X_L, Y_L)$, which can distill shared knowledge from distant nodes or disconnected components to favor more accurate estimation.

Moreover, we expect to capture and encode the similarities of nodes' ground-truth labels into the structure or communities of an auxiliary graph G_{aux} . It can be built as follows: 1) $G_{aux} = G$, input graph for graphs with noisy or missing features; 2) $G_{aux} = G_f$, a KNN-graph built according to node features for graphs with heterophily; 3) $G_{aux} = G_e$, a KNN-graph built according to node embeddings output by the teacher $f_t(\cdot)$ for others. More specifically, a KNN-graph $G(\mathbf{h})$ of a set of vectors h_1, \dots, h_n is built as follows: link every h_i to its top- k nearest vectors via a KNN algorithm with *Gaussian* distance, drop the edge directions, and then calculate the weights via similarity scores $W_{i,j}^{(aux)} = \text{ReLU} (h_i^T h_j)^{\gamma'}$ with a distribution-controlling hyper-parameter $\gamma' > 0$ for edge weights $W(aux) \in \mathbb{R}^{n \times n}$.

Label Smoothing and Curriculum Design To get multiscale label signals, we iteratively smooth labels on G_{aux} with initial signal $Y^{[0]} = Y$ from $f_t(\cdot)$ as: $Y^{[i+1]} = P_{aux}$. $Y^{[i]}$, $\forall i \in [0, n_T)$, where $P_{aux} = D_{aux}^{-1} W^{(aux)}$ and D_{aux} are the probability and degree matrix on G_{aux} , respectively. Note that this simplified Label Propagation without fixing f_L will definitely encounter over-smoothness similar to SGC, but it's just what we desire (see next paragraph). After that, a curriculum C can be defined as $\mathcal{T}_0, \mathcal{T}_1, \cdots, \mathcal{T}_{n_T}$, where task $\mathcal{T}_i = (D^{(i)}, f_\theta, \ell(\cdot), p^{(i)})$, *i.e.*, the graph encoder f_θ and the loss $\ell(\cdot)$ are shared in all tasks but the training data $D^{(i)} = (X, Y^{(i)})$ and pacing strategies $p^{(i)}$ vary. Here, we prepare $Y^{(i)} = Y^{[n_T - i]}, \forall i \in [0, n_T]$. In other words, the encoder f_θ will be encouraged to learn tasks from \mathcal{T}_0 to \mathcal{T}_{n_T} where easy tasks containing easy data $D^{(i)}$ are solved before the relatively harder ones with *even paces*. And finally, it will be fine-tuned to solve the original task with $D_0 = (X_L, Y_L)$. Analysis and Interpretations Next we give some analysis and intuitions on what *SmoothCurriculum* exactly does and how it guides the training in the following aspects: 1) optimizing from convex to non-convex: as claimed in (Bengio et al. 2009; Wang, Chen, and Zhu 2022), curriculum learning with priority for easy tasks can be equivalently understood as landscape smoothing for empirical loss contributing to a more convex optimization problem, which guides models to find a local minima with less vibration and better generalizability. 2) learning spectral knowledge from low- to high-frequency: $Y^{[i]} = (P_{aux})^i Y^{[0]} = U \Lambda^i U^T Y^{[0]}$ and $\Lambda^i = \text{diag}(\lambda_1^i, \lambda_2^i, \cdots, \lambda_n^i)$ with always decreasing eigenvalues. Let i vary *decreasingly*, and consider important values $\{i_j,j\in[1,n]\}$ where at time $i_j,Y^{[i_j]}\approx \sum_{t=1}^j \lambda_t^{i_j} u_t u_t^TY^{[0]}$ with the dominating top-j eigenvalues $\{\lambda_k^i, k \in [1, j]\}$. Then from $Y^{[i_j]}$ to $Y^{[i_{j+1}]}$, old knowledge will be reviewed due to $\lambda_t^{i_j} \leq \lambda_t^{i_{j+1}}$ and some relative high-frequent component $u_{j+1}u_{j+1}^T Y^{[0]}$ as new information will be injected to label signals. Additionally, if we independently consider a single signal $y^{[0]}$, then $u_{j+1}u_{j+1}^T y^{[0]} = (u_{j+1}^T y^{[0]}) u_{j+1} \propto u_{j+1}$ introducing a new channel for spectral embedding encoding some new details leading to more complex clustering structures. Thus models can learn to discriminate nodes from coarse to fine. 3) learning spatial knowledge from global to local: Intuitively, the shared commonsense is illustrated first due to $\lim_{i \to \infty} (P_{aux})^i Y^{[0]} = \mathbf{1}_n \mathbf{1}_n^T Y^{[0]}$, which encodes the global label frequency. Then the pieces of information in big communities, small communities, and local environments are presented in order because over-smoothness happens quickly in high-density regions but slowly otherwise. In other words, it also spatially favors coarse-to-fine node discrimination via perceiving the multi-scale community structure or density varieties of G_{aux} .

4 Experimental Results

In this section, we conduct extensive experiments to evaluate the effectiveness of our method (applied with GCN, GAT, and GIN) by comparing it with twelve baselines on twelve real-world graph benchmarks on semi-supervised node classification tasks. Note that our method can be applied to more sophisticated spatial propagation-based GNN backbones to further improve its performance, but we prefer basic ones to keep it simple and evaluate its potential. Due to space limitations, some experimental details are given in supplementary materials including dataset descriptions, implementations, omitted results (*e.g.*, with other layers, with different splits, comparisons with more baselines on heterophilous graphs, as well as standard errors), hyper-parameters (searching spaces and specific configurations), and some more visualizations.

Experimental Settings They are performed on an Ubuntu system with a single GeForce RTX 2080Ti GPU (12GB Memory) and 40 Intel(R) Xeon(R) Silver 4210 CPUs. And the proposed model is implemented by Pytorch (Paszke et al. 2019) and optimized with Adam Optimizer. For a fair comparison, twelve real-world public benchmarks are chosen, including two kinds: 1) eight graphs with homophily: four widely used scientific citation networks (*i.e.*, Core, Citeseer, Pubmed (Sen

et al. 2008), and a large-scale graph OGBN-ArXiv (Hu et al. 2020)), scientific co-authorship networks Physics and CS (Mernyei and Cangea 2020), as well as Amazon purchasing system Computers and Photo (Shchur et al. 2018); 2) four graphs with heterophily: webpage datasets Texas, Wisconsin, and Cornell (Pei et al. 2020) as well as an actor co-occurrence network Actor (Tang et al. 2009). Their statistics and adopted splits are summarized in Tab. 4 in supplementary materials. We adopt the standard semi-supervised training/validation/testing splits for them following prior works (Kipf and Welling 2017; Chen et al. 2020, 2022). Furthermore, twelve baselines or state-of-the-art GNN models are applied for comparison including four vanilla classic models (GCN, SGC, GAT, and GIN), two spectral-based methods (ChebNet (Defferrard, Bresson, and Vandergheynst 2016) and BernNet (He et al. 2021)), a normalization-based method Pairnorm (Zhao and Akoglu 2020), some residual connections-based methods including GCNII (Chen et al. 2020), GPRGNN (Chien et al. 2021), APPNP (Gasteiger, Bojchevski, and Günnemann 2019), JKNet (Xu et al. 2018), and DAGNN (Liu, Gao, and Ji 2020). For a fair comparison with spectral-based baselines, we view the orders of Laplacian used in filters as the depths.

Node Classification with Homophily and Heterophily We call the proposed method applied to GCN, GAT, and GIN *Ours(GCN)*, *Ours(GAT)*, and *Ours(GIN)*, respectively, where Ours(GCN) is the default, *i.e.*, *Ours*. We run each experiment five times with different initializations, and report the average accuracies in Tab. 1 and Tab. 2 with varied numbers of layers on these benchmarks. The standard errors and results with some other layers are given in supplementary materials. From Tab. 1 and 2, it is shown that our model consistently achieved the best results against these state-ofthe-art counterparts in almost all listed layers. Notably, for Amazon Computers Dataset, we get a performance improvement compared to DAGNN of more than 6.8% and 7.4% in 32 and 64 layers, respectively. As observed from Tab. 2, our method outperforms any other listed model by very large margins on four heterophilous graphs and the large-scale graph OGBN-ArXiv. In supplementary materials, we also provide results with fully supervised random splits compared to the listed counterparts (see Tab. 16) and more baselines on these heterophilous graphs (see Tab. 17). These results demonstrate its potential to alleviate over-smoothness in deep layers.

Node Classification with Noisy Features Sometimes features can provide enough meaningful supervision signals for node prediction, which veils the ability of a GNN for *structure understanding*. In this subsection, we evaluate our model in a challenging task called *Node Classification with Noisy Features* where all node features are substituted by noise sampled from Standard Normal Distribution $\mathcal{N}(0, 1)$ while only the structure of input graph remains. This task is more difficult than that in (Zhao and Akoglu 2020), since: 1) The feature substitution is conducted for all nodes instead of the nodes out of the training set only. 2) We make the features noisy instead of replacing them with zeros. Intuitively, this task tests *how deeply GNNs can understand the input structure*, *i.e.*, whether they can capture more useful structural knowledge for alleviating the adverse effect of noise in

Method		Cora		Citeseer		Pubmed		CS		Physics		Computers	Avg.Rank
#Layes	32	64	32	64	32	64	32	64	32	64	32	64	
GCN	31.90	27.56	36.66	25.40	44.22	32.65	41.29	34.23	79.87	75.34	58.30	37.58	12.75
SGC	63.84	55.39	67.50	63.08	70.70	65.33	70.52	72.51	91.46	90.77	37.44	37.50	10.33
ChebNet	31.90	20.63	33.43	24.90	48.67	45.37	29.28	23.24	70.35	50.74	58.58	50.12	12.67
GAT	72.28	31.92	59.08	22.90	78.72	41.88	85.85	12.83	91.87	17.75	76.05	37.18	10.92
GIN	60.92	31.90	47.32	23.10	72.94	40.23	52.90	20.79	83.02	24.98	39.18	37.50	12.42
Pairnorm	65.00	66.24	44.20	41.48	72.12	71.72	72.71	68.62	88.51	89.11	74.96	74.35	9.67
GCNII	85.29	85.34	73.24	73.00	79.81	79.88	71.67	72.11	93.15	92.79	37.56	37.50	6.67
JKNet	73.23	72.54	50.68	52.22	63.77	69.10	81.82	82.84	90.92	89.88	67.99	67.78	9.17
GPRGNN	83.13	82.48	71.01	70.96	78.46	78.92	89.56	89.33	93.49	93.26	41.94	78.30	6.83
DAGNN	83.39	82.16	72.59	71.00	80.58	80.44	89.60	89.47	93.31	93.52	79.73	79.23	4.92
APPNP	83.68	83.66	72.13	72.02	80.24	80.08	91.61	91.58	93.75	91.61	43.02	41.42	5.67
BernNet	81.38	17.72	70.82	39.10	70.24	32.86	91.54	9.20	92.27	19.38	81.06	12.81	10.67
Ours(GCN)	85.12	84.87	74.42	74.50	81.28	81.58	92.11	92.05	94.22	94.20	85.21	85.13	1.42
Ours(GAT)	84.60	84.68	74.26	74.04	80.46	80.20	91.30	91.26	94.02	94.02	84.82	85.04	3.33
Ours(GIN)	84.18	83.80	74.88	74.16	80.32	80.94	91.79	91.71	94.56	94.53	85.16	85.13	2.17

Table 1: Results of node classification tasks on Cora, Citeseer, Pubmed, CS, Physics, and Computers

Method		Photo		Texas		Wisconsin		Cornell		Actor		OGBN-ArXiv	Avg.Rank
#Layes	32	64	32	64	32	64	32	64	32	64	32	64	
GCN	58.47	50.21	62.16	62.16	57.84	57.84	56.76	56.76	25.16	25.16	46.38	42.95	10.00
SGC	26.08	24.57	56.41	56.96	51.29	52.16	58.57	55.41	26.17	25.88	34.22	23.14	11.67
ChebNet	65.28	64,83	64.86	64.86	52.94	52.94	55.86	52.25	25.46	25.46	41.00	35.16	10.33
GAT	83.73	25.36	65.41	64.86	53.73	53.33	54.05	55.14	25.54	25.62	59.78	36.63	9.33
GIN	65.98	25.27	62.17	60.00	47.06	50.98	54.59	55.14	24.36	23.45	65.17	60.56	11.33
PairNorm	82.66	79.55	41.08	40.68	52.84	52.94	36.89	40.68	24.33	23.23	63.32	43.57	11.91
GCNII	62.95	65.12	69.19	65.41	70.31	59.02	74.16	56.92	34.28	34.64	72.60	70.07	5.33
JKNet	78.42	79.73	61.08	66.49	52.76	56.08	57.30	51.49	28.80	28.26	66.31	65.80	8.25
GPRGNN	91.74	91.28	62.27	61.08	71.35	64.90	58.27	52.16	29.88	32.43	70.18	69.98	6.25
DAGNN	89.96	87.86	57.68	60.27	50.84	51.76	58.43	52.43	27.73	25.45	71.46	70.58	8.92
APPNP	59.62	63.63	60.68	64.32	54.24	59.90	58.43	54.69	28.65	28.19	66.94	66.90	8.25
BernNet	91.59	16.53	61.08	16.76	63.53	24.71	52.43	11.89	28.19	20.66	45.16	37.18	11.92
Ours(GCN)	92.06	92.03	85.41	84.86	83.14	84.71	82.70	82.62	38.42	38.49	74.07	73.95	1.33
Ours(GAT)	91.98	92.00	78.92	78.38	73.73	74.90	77.84	81.62	34.97	35.54	74.37	74.41	2.50
Ours(GIN)	92.03	91.98	82.57	83.12	81.78	81.20	81.08	81.08	34.51	34.50	75.02	74.85	2.25

Table 2: Results of node classification tasks on Photo, heterophilous graphs (*e.g.*, Texas), and a large-scale graph OGBN-ArXiv

features. We adopt our standard model *Ours(GCN)* itself as the teacher and the original graph as the auxiliary graph. As observed from Fig. 1, our model outperforms most evaluated baselines in nearly all layers by a significant margin, showing its effective extraction and preservation of structural semantics. While SGC with no more than 64 layers can achieve better results in some layers, it suffers from over-smoothness severely with sufficient large layers (*e.g.*, 10^3 or 10^4 layers).

Ablation, Hyper-parameter Studies, and Visualizations In order to demonstrate the effects of each individual part of our model and learning framework, we conduct extensive ablation studies following (Chen et al. 2020) and report the results on seven graph benchmarks in Tab. 3. In the following, we take *Ours(GCN)* as our standard model and independently drop each of the five parts: SoftGraphAIN (SG), residual connections (RC), R-SoftGraphAIN (R-SG), label smoothing (LS), and the holistic curriculum learning framework (CL), where *w.o. X* means that we drop the part *X*. From Tab. 3, we observe that every part contributes a portion to the performance gain, among which R-SG is the most significant since it reduces the risk of features and structure forgetting simultaneously. To facilitate a better understanding, we plot the varying effects of softly normalizing extents (the hyperparameter a in Eq. 7) in Fig. 2, from which we can see a comprehensive ascending-and-then-descending trend, showing the benefits of this soft version compared to the hard one. In supplementary materials, we study some other hyperparameters (*e.g.*, α and k_{KNN}), and detailedly visualize the embeddings produced by our method and some counterparts.

Discussion On the Time and Space Complexities The theoretical time complexity is analysed $O(Ld^2d_0)$ where $d, d_0 \ll n$ if partial-SVD or truncated-SVD (Halko, Martinsson, and Tropp 2011) is utilized. And it would become $O(Ld^3)$ with a full SVD decomposition. However, it is ef-

Figure 1: Results of different models with varying layers in node classification tasks with noisy features

Method	#Layers	Cora	Citeseer	Pubmed	CS	Physics	Computers	Photo
w.o. SG	32	83.30 ± 0.12	72.98 ± 0.50	80.26 ± 0.98	91.79 ± 0.07	94.23 ± 0.25	82.34	90.26
	64	84.60 ± 0.29	72.62 ± 0.98	80.30 ± 0.23	91.70 ± 0.07	94.25 ± 0.13	84.50	91.69
w.o. RC	32	82.88 ± 0.79	72.82 ± 1.01	78.84 ± 0.59	91.40 ± 0.26	92.99 ± 0.19	83.78	91.42
	64	82.40 ± 0.33	71.04 ± 0.55	78.60 ± 0.27	89.03 ± 0.62	92.59 ± 0.63	84.25	91.14
$w.o. R-SG$	32	40.22 ± 5.71	29.32 ± 3.47	46.28 ± 4.91	51.61 ± 18.23	77.20 ± 11.95	61.52	83.77
	64	36.74 ± 4.36	27.70 ± 2.94	28.61 ± 3.64	28.51 ± 4.70	60.94 ± 8.23	49.94	70.09
w.o. LS	32	83.96 ± 0.65	73.64 ± 0.68	81.04 ± 0.21	92.02 ± 0.05	94.10 ± 0.11	84.52	91.50
	64	84.00 ± 0.22	74.34 ± 0.79	80.86 ± 0.79	91.95 ± 0.08	94.06 ± 0.08	84.91	91.64
w.o. CL	32	82.40 ± 0.65	73.20 ± 0.77	79.34 ± 0.39	89.60 ± 0.20	92.46 ± 0.61	84.00	90.58
	64	82.58 ± 0.87	72.54 ± 0.42	79.00 ± 0.54	89.23 ± 0.12	92.22 ± 0.56	84.44	90.41
Ours(GCN)	32	85.12 ± 0.15	74.42±0.26	81.28 ± 0.18	92.50 ± 0.09	94.45 ± 0.06	85.21	92.06
	64	84.87 ± 0.26	74.50 ± 0.23	81.58 ± 0.66	92.41 ± 0.07	94.52 ± 0.04	85.13	92.03

Table 3: Ablation studies on seven benchmarks including Cora, Citeseer, Pubmed, CS, Physics, Computers, and Photo

Figure 2: Results of Ours(GCN/GAT/GIN) against the varying hyper-parameter $a \in [0, 1]$ controlling the normalizing extent

ficient under the high-parallelizability implemented via Pytorch. The theoretical space complexity is $O(L(n+d) d)$.

5 Conclusion

In this paper, we propose *R-SoftGraphAIN* to alleviate the over-smoothness of deep GNNs, by novelly employing soft normalization of the covariance matrix with appropriately incorporated residual connections. We show in theory that the technique can maximally preserve the diversities of knowledge from both structures and features even at a sufficiently large depth against over-smoothness. Furthermore, in order to ease the difficulty of the optimization of deep

GNNs, a label-smoothing-based curriculum learning framework (called *SmoothCurriculum*) is proposed to intuitively encourage the encoder to digest knowledge from low- to highfrequency and to learn to discriminate nodes from coarse to fine. Extensive experiments were carried out against semisupervised node classification tasks to show the effectiveness of our model by demonstrating its practical performance gain compared to twelve state-of-the-art baselines on twelve realworld graph benchmarks. In future work, we will explore more applications of our method such as link prediction, graph classification, and community detection tasks.

Acknowledgements

This work is supported by the National Science Foundation of China (Nos. 12271098) and Taishan Scholars Young Expert Project of Shandong Province (No. tsqn202211215) and the University-Industry Cooperation Project of Fujian Province, China (2023H6008). The complete version of this paper can be found at https://arxiv.org/abs/2312.08221 with all codes at https://github.com/jslijin/Research-Paper-Codes/.

References

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, 41–48.

Bo, D.; Hu, B.; Wang, X.; Zhang, Z.; Shi, C.; and Zhou, J. 2022. Regularizing graph neural networks via consistencydiversity graph augmentations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 3913–3921.

Bo, D.; Wang, X.; Shi, C.; and Shen, H. 2021. Beyond lowfrequency information in graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 3950–3957.

Bodnar, C.; Di Giovanni, F.; Chamberlain, B. P.; Lio, P.; and ` Bronstein, M. M. 2022. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. *arXiv preprint arXiv:2202.04579*.

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2014. Spectral Networks and Locally Connected Networks on Graphs. arXiv:1312.6203.

Caubrière, A.; Tomashenko, N.; Laurent, A.; Morin, E.; Camelin, N.; and Estève, Y. 2019. Curriculum-based transfer learning for an effective end-to-end spoken language understanding and domain portability. In *20th Annual Conference of the International Speech Communication Association (InterSpeech)*, 1198–1202.

Chamberlain, B.; Rowbottom, J.; Gorinova, M. I.; Bronstein, M.; Webb, S.; and Rossi, E. 2021. Grand: Graph neural diffusion. In *International Conference on Machine Learning*, 1407–1418. PMLR.

Chen, M.; Wei, Z.; Huang, Z.; Ding, B.; and Li, Y. 2020. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, 1725–1735.

Chen, T.; Zhou, K.; Duan, K.; Zheng, W.; Wang, P.; Hu, X.; and Wang, Z. 2022. Bag of Tricks for Training Deeper Graph Neural Networks: A Comprehensive Benchmark Study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, DOI:10.1109/TPAMI.2022.3174515.

Chien, E.; Peng, J.; Li, P.; and Milenkovic, O. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *International Conference on Learning Representations*.

Choi, J.; Choi, H.; Hwang, J.; and Park, N. 2022. Graph neural controlled differential equations for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 6367–6374.

Chu, G.; Wang, X.; Shi, C.; and Jiang, X. 2021. CuCo: Graph Representation with Curriculum Contrastive Learning. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, 2300–2306.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29.

Demir, A.; Koike-Akino, T.; Wang, Y.; Haruna, M.; and Erdogmus, D. 2021. EEG-GNN: Graph Neural Networks for Classification of Electroencephalogram (EEG) Signals. In *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, 1061–1067.

Eliasof, M.; Haber, E.; and Treister, E. 2021. Pde-gcn: Novel architectures for graph neural networks motivated by partial differential equations. *Advances in Neural Information Processing Systems*, 34: 3836–3849.

Eliasof, M.; Haber, E.; and Treister, E. 2022. pathgcn: Learning general graph spatial operators from paths. In *International Conference on Machine Learning*, 5878–5891. PMLR.

Fei, Z.; Zhang, Q.; and Zhou, Y. 2021. Iterative GNN-based decoder for question generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2573–2582.

Feng, W.; Zhang, J.; Dong, Y.; Han, Y.; Luan, H.; Xu, Q.; Yang, Q.; Kharlamov, E.; and Tang, J. 2020. Graph random neural networks for semi-supervised learning on graphs. *Advances in neural information processing systems*, 33: 22092– 22103.

Gasteiger, J.; Bojchevski, A.; and Günnemann, S. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *International Conference on Learning Representations*.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*, 1263–1272.

Guan, Y.; Zhang, J.; Tian, K.; Yang, S.; Dong, P.; Xiang, J.; Yang, W.; Huang, J.; Zhang, Y.; and Han, X. 2022. Node-Aligned Graph Convolutional Network for Whole-Slide Image Representation and Classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 18813–18823.

Hacohen, G.; and Weinshall, D. 2019. On the power of curriculum learning in training deep networks. In *International Conference on Machine Learning*, volume 97, 2535–2544.

Halko, N.; Martinsson, P. G.; and Tropp, J. A. 2011. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Review*, 53(2): 217–288.

Han, P.; Zhao, P.; Lu, C.; Huang, J.; Wu, J.; Shang, S.; Yao, B.; and Zhang, X. 2022. GNN-Retro: Retrosynthetic Planning with Graph Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 4014–4021.

He, M.; Wei, Z.; Xu, H.; et al. 2021. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. *Advances in Neural Information Processing Systems*, 34: 14239– 14251.

Hertz, A.; Perel, O.; Giryes, R.; Sorkine-Hornung, O.; and Cohen-Or, D. 2021. Sape: Spatially-adaptive progressive

encoding for neural optimization. *Advances in Neural Information Processing Systems*, 34: 8820–8832.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33: 22118–22133.

Huang, W.; Rong, Y.; Xu, T.; Sun, F.; and Huang, J. 2020. Tackling over-smoothing for general graph convolutional networks. arXiv:2008.09864.

Kipf, T. N.; and Welling, M. 2016. Variational graph autoencoders. *arXiv preprint arXiv:1611.07308*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations*.

Li, G.; Müller, M.; Ghanem, B.; and Koltun, V. 2021. Training graph neural networks with 1000 layers. In *International conference on machine learning*, 6437–6449.

Li, G.; Muller, M.; Thabet, A.; and Ghanem, B. 2019. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision*, 9267–9276.

Liu, F.; Xue, S.; Wu, J.; Zhou, C.; Hu, W.; Paris, C.; Nepal, S.; Yang, J.; and Yu, P. S. 2021. Deep learning for community detection: progress, challenges and opportunities. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 4981–4987.

Liu, M.; Gao, H.; and Ji, S. 2020. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 338–348.

Liu, X.; Lai, H.; Wong, D. F.; and Chao, L. S. 2020. Norm-Based Curriculum Learning for Neural Machine Translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 427–436.

Luan, S.; Zhao, M.; Chang, X.-W.; and Precup, D. 2019. Break the ceiling: stronger multi-scale deep graph convolutional networks. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 10945–10955.

Maurya, S. K.; Liu, X.; and Murata, T. 2022. Simplifying approach to node classification in Graph Neural Networks. *Journal of Computational Science*, 101695.

Mernyei, P.; and Cangea, C. 2020. Wiki-cs: A wikipediabased benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*.

Narvekar, S.; Peng, B.; Leonetti, M.; Sinapov, J.; Taylor, M. E.; and Stone, P. 2020. Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. *Journal of Machine Learning Research*, 21(181): 1–50.

Oono, K.; and Suzuki, T. 2020. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. In *International Conference on Learning Representations*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style,high-performance deep learning library. *Advances in neural information processing systems*, 32: 8024–8035.

Pei, H.; Wei, B.; Chang, K. C.-C.; Lei, Y.; and Yang, B. 2019. Geom-GCN: Geometric Graph Convolutional Networks. In *International Conference on Learning Representations*.

Pei, H.; Wei, B.; Chang, K. C.-C.; Lei, Y.; and Yang, B. 2020. Geom-GCN: Geometric Graph Convolutional Networks. In *International Conference on Learning Representations*.

Rong, Y.; Huang, W.; Xu, T.; and Huang, J. 2020. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *International Conference on Learning Representations*.

Rusch, T. K.; Chamberlain, B.; Rowbottom, J.; Mishra, S.; and Bronstein, M. 2022. Graph-coupled oscillator networks. In *International Conference on Machine Learning*, 18888– 18909. PMLR.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine*, 29(3): 93–93.

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

Sinha, S.; Garg, A.; and Larochelle, H. 2020. Curriculum by smoothing. *Advances in Neural Information Processing Systems*, 33: 21653–21664.

Soviany, P.; Ionescu, R. T.; Rota, P.; and Sebe, N. 2022. Curriculum Learning: A Survey. *International Journal of Computer Vision*, 130(6): 1526–1565.

Tang, J.; Sun, J.; Wang, C.; and Yang, Z. 2009. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 807–816.

Thakoor, S.; Tallec, C.; Azar, M. G.; Azabou, M.; Dyer, E. L.; Munos, R.; Veličković, P.; and Valko, M. 2021. Large-Scale Representation Learning on Graphs via Bootstrapping. In *International Conference on Learning Representations*.

Topping, J.; Giovanni, F. D.; Chamberlain, B. P.; Dong, X.; and Bronstein, M. M. 2022. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Verma, V.; Qu, M.; Kawaguchi, K.; Lamb, A.; Bengio, Y.; Kannala, J.; and Tang, J. 2021. Graphmix: Improved training of gnns for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 10024–10032.

Wang, C.; Wu, Y.; Liu, S.; Zhou, M.; and Yang, Z. 2020. Curriculum Pre-training for End-to-End Speech Translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 3728–3738.

Wang, H.; and Leskovec, J. 2020. Unifying graph convolutional neural networks and label propagation. *arXiv preprint arXiv:2002.06755*.

Wang, X.; Chen, Y.; and Zhu, W. 2022. A Survey on Curriculum Learning. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 44(9): 4555–4576.

Wang, Y.; Wang, W.; Liang, Y.; Cai, Y.; and Hooi, B. 2021. Curgraph: Curriculum learning for graph classification. In *Proceedings of the Web Conference 2021*, 1238–1248.

Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; and Weinberger, K. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*, 6861–6871.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1): 4–24.

Xie, Y.; Lv, S.; Qian, Y.; Wen, C.; and Liang, J. 2022. Active and Semi-Supervised Graph Neural Networks for Graph Classification. *IEEE Transactions on Big Data*, 8(4): 920– 932.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*.

Xu, K.; Li, C.; Tian, Y.; Sonobe, T.; Kawarabayashi, K.-i.; and Jegelka, S. 2018. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, 5453–5462.

Yang, C.; Wang, R.; Yao, S.; Liu, S.; and Abdelzaher, T. 2020. Revisiting Over-smoothing in Deep GCNs. arXiv:2003.13663.

Yang, H.; Ma, K.; and Cheng, J. 2021. Rethinking graph regularization for graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 4573–4581.

Yun, S.; Kim, S.; Lee, J.; Kang, J.; and Kim, H. J. 2021. Neognns: Neighborhood overlap-aware graph neural networks for link prediction. *Advances in Neural Information Processing Systems*, 34: 13683–13694.

Zhao, L.; and Akoglu, L. 2020. PairNorm: Tackling Oversmoothing in GNNs. In *International Conference on Learning Representations*.

Zhou, D.; Bousquet, O.; Lal, T.; Weston, J.; and Schölkopf, B. 2003. Learning with local and global consistency. *Advances in neural information processing systems*, 16.

Zhou, K.; Dong, Y.; Wang, K.; Lee, W. S.; Hooi, B.; Xu, H.; and Feng, J. 2021. Understanding and resolving performance degradation in deep graph convolutional networks. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2728–2737.

Zhou, T.; Wang, S.; and Bilmes, J. 2020. Curriculum learning by dynamic instance hardness. *Advances in Neural Information Processing Systems*, 33: 8602–8613.

Zhu, J.; Yan, Y.; Zhao, L.; Heimann, M.; Akoglu, L.; and Koutra, D. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33: 7793–7804.

Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; and Wang, L. 2021. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, 2069–2080.

A The Organization of Supplementary Material

The supplementary material is organized as follows:

- some more related work in Sec. B;
- Algorithm 1 and its detailed description (including the details on *Fuzzy Connections*) omitted in the main text due to space limit (see Sec. C);
- an additional pre-processing node filtering operation in our implementation of the proposed curriculum learning framework (see Sec. D);
- the downstream linear classifier and loss function for node classification tasks in Sec. E;
- omitted proofs in Sec. F;
- experimental details including dataset descriptions, more experimental results, implementation details, and hyperparameter configurations in Sec. G;
- extensive visualizations in Sec. H;
- the time/space complexities in Sec. I;
- additional hyper-parameter studies on α , γ , and k_{KNN} in Sec. J;
- discussion on the connections between R-SoftGraphAIN and SmoothCurriculum in Sec. K.

B More Possibly Related Literature

In this section, we give a short review of more related work in order to enrich the background of this paper and improve its completeness. However, we think the literature referenced in the main text is enough to understand the proposed method for readers.

Though some methods also aim to alleviate oversmoothness in deep GNNs, they have different design motivations or perspectives. PDE-GCN (Eliasof, Haber, and Treister 2021) and (Rusch et al. 2022) are both ODE- or PDE-based methods. (Chamberlain et al. 2021) and (Bodnar et al. 2022) are designed from different diffusion processes. (Zhu et al. 2020) is mainly designed by dealing with heterophily and does not focus on the over-smoothness. (Eliasof, Haber, and Treister 2022) discards the traditional aggregation operator in GCN and proposes a novel path-based or random walkbased operator. Its authors argue that the over-smoothness will not happen in this novel aggregation process and one can naturally avoid the necessity of alleviating this issue.

Besides, many works consider improving the training processes of models from different perspectives. (Yang, Ma, and Cheng 2021) is based on regularization. (Verma et al. 2021) and (Bo et al. 2022) are augmentation-based methods. And compared to them, we propose a novel curriculum learning framework to find local minima with better generalizing characteristics. (Hertz et al. 2021) proposes an optimization algorithm sharing a similar concept of training from low to high frequency. However, we are essentially different works from different domains with the aim of solving different problems. (Wang and Leskovec 2020) and (Zhou et al. 2003) are also label propagation-related works. However, they seem to have little connection with ours.

C Algorithm 1 (Fuzzy R-SoftGraphAIN)

Recall Sec. 3.1 in the main text. For clearer understanding and better reproducibility, here we give an implementation of the comprehensive structure of the proposed *Fuzzy R-SoftGraphAIN* in Algorithm 1.

We give a brief description of Algorithm 1:

- Lines $1 \rightarrow 3$ compute the symmetrically normalized adjacency matrix \ddot{A} from \ddot{A} .
- Lines $4 \rightarrow 5$ compute the centering operator T, which can normalize the row mean vector of a node embedding matrix into 0, as well as the parameter β , and initializes $q' = q^0 = 1.$
- Lines $6 \rightarrow 8$ obtain H_1 from H_0 without any skip connection.
- Line 9 initializes the matrices S_{last} and S_{init} , which represent the items in fuzzy *last* residual connection and fuzzy *initial* connection, respectively (please kindly see the specific introduction in the following subsection).
- Line $10 \rightarrow 17$ describe the comprehensive updating process of node embedding matrix H_t .
- Line 11 (corresponding to Eq. 8 in the main text) computes B_{t-1} from H_{t-1} with *fuzzy connections* (please kindly see the specific introduction in the following subsection).
- Line 12 does a partial singular value decomposition retaining the d_0 most dominating components of $B_{t-1}^T B_{t-1}$.
- Line 13 computes H_t from these components.
- Line 14 updates q' and keeps $q' = q^{t-1}$.
- Line $15 \rightarrow 16$ update the items S_{last} and S_{init} in *fuzzy connections*.
- Line 18 returns the final node embedding matrix H_L , which will be fed into a linear classifier to solve the subsequent downstream tasks such as node classification (see Sec. E).

C.1 Fuzzy Connections: Smoother Versions For Skip (Residual and Initial) Connections

To counteract the negative impacts of possible noise in the node embeddings in an individual layer (*i.e.*, H_{t-1} for a residual connection and H_1 for an initial connection ²), we design a smoother version of these two connections called fuzzy residual connection and fuzzy initial connection, respectively. They can be specifically formulated by expanding Line 12 in Algorithm 1 as follows.

$$
S_{init}^{(t)} = \sum_{i=1}^{t-1} q^{i-1} \cdot H_i \in \mathbb{R}^{n \times d},
$$

\n
$$
S_{last}^{(t)} = \sum_{i=1}^{t-1} q^{t-1-i} \cdot H_i \in \mathbb{R}^{n \times d}.
$$
\n(10)

²Note that we have substituted X by H_1 due to possible mismatch between their dimensions (see the last two lines in Sec. 3.1 in the main text), which makes our implementation more convenient. **Input**: adjacency matrix $A \in \{0,1\}^{n \times n}$, node features $H_0 = X \in \mathbb{R}^{n \times f}$ (with n the node num and f the feature dim)

Output: embeddings $H = H_L \in \mathbb{R}^{n \times d}$ (with d the hidden dim)

Hyper-Parameters:

the coefficients in the soft graph normalization: $a, b \in [0, 1]$ $(a + b \le 1)$,

the coefficients in the skip connections: $\alpha, \gamma \in [0, 1]$ ($\alpha + \gamma \leq 1$),

the coefficients in the fuzzy connections: $p, q \in [0, 1]$,

the parameter in partial SVD algorithm: $d_0 \leq n$,

the layer num: $L \in \mathbb{N}_+$,

the non-linear activation function: $\sigma(\cdot)$ (ReLU(\cdot) default)

Trainable-Parameters: $\{W_t\}_{1 \leq t \leq L}$ (in GCN)

Note: Here we consider general situations, where $W_1 \in \mathbb{R}^{f \times d}$ if $f \neq d$, otherwise $W_t \in \mathbb{R}^{d \times d}$ for $\forall t \in [1, L]$.

1:
$$
\tilde{A} \leftarrow A + I_n
$$

\n2: $\tilde{D} \leftarrow diag(\tilde{A} \cdot 1_n)$
\n3: $\hat{A} \leftarrow \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$
\n4: $T \leftarrow I_n - \frac{1}{n} I_n I_n^T \in \mathbb{R}^{n \times n}$
\n5: $\beta \leftarrow 1 - \alpha - \gamma$, $q' \leftarrow 1$
\n6: $B_0 \leftarrow T \cdot \hat{A} H_0 W_1$
\n7: $U_{d_0}, \Lambda_{d_0}^{\frac{1}{2}} \leftarrow d_0\text{-SVD} (B_0^T B_0)$
\n8: $H_1 \leftarrow \sigma (B_0 \cdot \left[a \cdot U_{d_0} \Lambda_{d_0}^{-\frac{1}{2} \cdot b} U_{d_0}^T + (1 - a) \cdot I_d\right])$
\n9: $S_{last} \leftarrow H_1$, $S_{init} \leftarrow H_1$
\n10: **for each** $t \in [2, L]$ **do**
\n11: $B_{t-1} = \alpha \cdot T \hat{A} H_{t-1} W_t + \beta \cdot S_{last} + \gamma \cdot S_{init}$
\n12: $U_{d_0}, \Lambda_{d_0}^{\frac{1}{2}} \leftarrow d_0\text{-SVD} (B_{t-1}^T B_{t-1})$
\n13: $H_t \leftarrow \sigma (B_{t-1} \cdot \left[a \cdot U_{d_0} \Lambda_{d_0}^{-\frac{1}{2} \cdot b} U_{d_0}^T + (1 - a) I_d\right])$
\n14: $q' \leftarrow q'$
\n15: $S_{last} \leftarrow S_{last} \cdot p + H_t$
\n16: $S_{init} \leftarrow S_{init} + q' \cdot H_t$
\n17: **end for**

 \triangleright obtain H_1

used in Line 11)

18: return H_L

Figure 3: Scatter plots for different deep models on synthetic data (considering only forward pass without trainable parameters). This vividly illustrates the phenomenon of the over-smoothness where node embeddings tend to collapse to a line or several line segments. We can clearly observe that ours can successfully keep the clusters away from blending, thus effectively alleviating the over-smoothness.

Then we can reformulate Line 8 in Algorithm 1 as:

$$
B_{t-1} = \alpha \cdot T\hat{A}H_{t-1}W_t + \beta \cdot S_{last}^{(t)} + \gamma \cdot S_{init}^{(t)}.\tag{11}
$$

One can easily check that $S_{init}^{(t)} \rightarrow H_1$ and $S_{last}^{(t)} \rightarrow H_{t-1}$ when $q \to 0$, which is a special degenerate case, *i.e.*, vanilla initial and residual connections.

C.2 How To Apply It On Other GNN Encoders

Since our method does not essentially depend on or alter the aggregating operations employed in GCN, it can be naturally viewed as a plug-and-play module for nearly all spatial aggregation-based GNN encoders. In our experiments, in addition to GCN (Kipf and Welling 2017), we also evaluate our method based on GAT (Veličković et al. 2018) and GIN (Xu et al. 2019) (*i.e.*, *Ours(GAT)* and *Ours(GIN)*), and we found that on some graph benchmarks (*e.g.*, OGBN-ArXiv) they perform quite competitive or better than *Ours(GCN)* even adopting the exactly same hyper-parameters as *Ours(GCN)*'s without any further tuning. Here, we give a short description of how to achieve that.

From Line 6 and Line 11 in Algorithm 1, one can easily find that the formula $\hat{A}H_{t-1}W_t$ keeps intact as that in the t-th layer of GCN, which might imply that it can be substituted by some other aggregating formulas in another GNN encoders, *i.e.*, a universal form of $H_t = \text{Agg}(H_{t-1}; W_t)$.

More specifically, in GAT, it is:

$$
H_t^{(i)} = \Big\|_{k=1}^K \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k W_t^k H_{t-1}^{(j)} \right), \tag{12}
$$

where $H_t^{(i)}$ denotes the embeddings of node $i \in V$ at layer $t \in [1, L]$, K is the number of attention heads, \mathcal{N}_i is the first-order neighborhood of node i, and α_{ij}^k denoting the attention coefficient from node i to node j via head k can be represented as follows:

$$
\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\left[H^{(i)}\|H^{(j)}\right]W'\right)\right)}{\sum_{j\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\left[H^{(i)}\|H^{(j)}\right]W'\right)\right)},\tag{13}
$$

where W' is another trainable matrix.

As for GIN, $H_t = \text{Agg}(H_{t-1}; W_t)$ has a simple formula:

$$
H_t^{(i)} = \text{MLP}\left((1 + \epsilon) \cdot H_{t-1}^{(j)} + \sum_{j \in \mathcal{N}_i} H_{t-1}^{(j)} \right). \tag{14}
$$

Note that although we can always apply it to more sophisticated GNN encoders in addition to the selected basic ones for the sake of further performance improvement, we want to keep it simple and evaluate its potential. Some expanding work might be left as future work.

D Additional Node Filtering Pre-Processing Operation in Curriculum Learning Framework

We know that there would exist possible noises in the label estimation process (Sec. 3.2). Furthermore, we found in

experiments that the negative impacts introduced by these nettlesome noises could be accumulated during the subsequent smoothing process (Sec. 3.2), which would potentially damage the final performance to some non-negligible extent. To alleviate this issue, we apply an additional node filtering operation after label estimation based on the unconfidence scores estimated by the following normalized entropy function of these pseudo labels (*i.e.*, lots of probability distributions):

$$
H(p) = \frac{-\sum_{i=1}^{C} p_i \cdot \log(p_i)}{\log C} \in [0, 1],
$$
 (15)

where we consider $p \in \mathbb{R}^C$ as a probability distribution or a pseudo label of some node $v \in V$ and C is the number of classes in the node classification tasks. We filter those nodes with too large normalized entropy values out by substituting their pseudo labels via C-dimensional all-zero vectors because we think that they are unconfident predictions with little meaningful knowledge.

And in this process, we introduce a hyper-parameter $mask_ratio \in [0, 1]$ to control how many nodes we hope to filter out (see the hyper-parameter Tab. 21 and Tab. 22).

Besides, after this filtration, some pseudo-labels would become invalid probability distributions during the smoothing process. To fix this issue, we re-normalize all the pseudo- $\overline{\text{labels}}^3$.

Please kindly notice that there would be some other alternative methods to alleviate the negative impacts due to noise. Here we just give an example, which is employed in our experiments.

E Downstream Linear Classifier and Loss Function for Node Classification Tasks

We employ the following downstream linear classifier and the cross-entropy loss function:

$$
L_{CE}(H^{(L)}, y) = -\frac{1}{|V|} \sum_{v \in V} \sum_{c \in C} y_v(c) \cdot \log \hat{y}_v(c),
$$

$$
\hat{y}_v = \text{Softmax}\left(H_v^{(L)} W_{cls}\right) \in \mathbb{R}^{1 \times C},
$$
 (16)

where $H^{(L)}$ is the final node embedding matrix, y is the ground-truth one-hot node labels, and $\widetilde{W}_{cls} \in \mathbb{R}^{d \times C}$ is a trainable projection matrix. We train our model end-to-end by minimizing this loss function.

Note that we should substitute the y vector with the corresponding node pseudo-labels used in different stages of the proposed curriculum learning framework.

F Omitted Proofs

In this section, we give the proofs of the theorems and corollaries omitted in the main text.

Theorem 4. *(Corresponding to Theorem* 1 *in the main text)* $\forall t \geq 1$ *, GraphAIN satisfies that: 1*) $\mathbf{1}_n^T B_t = \hat{\mathbf{1}}_n^T H_t = \mathbf{0}_n$; 2) $H_t^T H_t = I_d$;

3)
$$
\overline{T}H_t = H_t^{\prime}
$$
;
\n4) $B_t = \overline{A}H_t$;
\nwhere $\overline{A} = T\hat{A}T$ denotes a doubly centered version of \hat{A} .

 3 to make the sum of elements in every pseudo-label equal to 1

Dataset	Nodes	Edges	Ave.Degree	Features	Classes	train/val/test
Cora	2.708	5.429	4.0	1.433	7	140/500/1000
Citeseer	3.327	4.732	2.84	3.703	6	120/500/1000
PubMed	19.717	44.338	4.5	500	3	60/500/1000
Coauthor CS	18.333	81.894	8.93	6.805	15	300/450/17583
Coauthor Physics	34.493	247,962	14.38	8.415	5	100/150/34243
Amazon Computers	13.381	245,778	36.74	767	10	200/300/12881
Amazon Photo	7.487	119,043	31.8	745	8	160/240/7087
Texas	183	309	3.38	1.703	5	87/59/37
Wisconsin	251	499	5.45	1.703	5	120/80/51
Cornell	183	295	3.22	1.703	5	87/59/37
Actor	7600	33544	8.83	931	5	3648/2432/1520
OGBN-ArXiv	169.343	1,166,243	13.77	128	40	91446/30482/47415

Table 4: Dataset statistics of twelve real-world benchmarks with their splits

Figure 4: Results of Ours(GCN/GAT/GIN) against the varying hyper-parameter $\gamma \in [0,1]$ controlling the coefficient of the trade-off in Skip Connections, *i.e.*, between initial connections and residual connections

Figure 5: Results of Ours(GCN/GAT/GIN) against the varying hyper-parameter $k_{KNN} \in \mathbb{N}_+$ controlling the number of the closest neighbors chosen by the vanilla KNN algorithm for every node

Proof. Recall that *GraphAIN* can be formulated as follows:

$$
H_t = B_{t-1} \left(B_{t-1}^T B_{t-1} \right)^{-\frac{1}{2}} \in \mathbb{R}^{n \times d}, \quad \forall \ t \ge 1,
$$

$$
B_{t-1} = T \cdot \hat{A} H_{t-1} \in \mathbb{R}^{n \times d}, \quad H_0 = X,
$$
 (17)

1) Because $T = I_n - \frac{1}{n} \mathbf{1}_n \mathbf{1}_n^T$, $\mathbf{1}_n^T T = \mathbf{0}_n^T$ holds. That is, it is true that $\mathbf{1}_n^T B_t = \mathbf{0}_d^T, \forall t \geq 0$, and consequently $\mathbf{1}_n^T H_t = \mathbf{0}_d^T, \forall t \geq 1.$

2) According to the definition:
$$
\forall t \ge 1
$$
, $H_t^T H_t = (B_{t-1}^T B_{t-1})^{-\frac{1}{2}} (B_{t-1}^T B_{t-1}) (B_{t-1}^T B_{t-1})^{-\frac{1}{2}} = I_n$.
\n3) $\mathbf{1}_n^T H_t = \mathbf{0}_d$, so $T \cdot H_t = (I_n - \frac{1}{n} \mathbf{1} \mathbf{1}^T) H_t = H_t - \frac{1}{n} \mathbf{1} \mathbf{1}^T H_t = H_t$.
\n4) $\forall t \ge 1, B_t = T \hat{A} H_t = T \hat{A} T H_t = \bar{A} H_t$.

Lemma 1. *Consider the following linear regression with an orthogonal condition:*

$$
Q^* = \arg\min_Q \|QA - B\|_F \quad \text{s.t.} \quad Q^T Q = I_d, \tag{18}
$$

and then we have $Q^* = UV^T$, where $BA^T = U\Sigma V^T$ is *the eigen-decomposition of* BA^T *,* $U, V \in \mathbb{R}^{d \times d}$ *are two* $orthogonal$ matrices, and $\sum \in \mathbb{R}^{d \times d'}$ is a diagonal matrix *containing all the singular values of* BA^T *.*

Proof.

$$
Q^* = \arg\min_{Q} ||QA - B||_F^2
$$

= $\arg\min_{Q} \langle QA - B, QA - B \rangle_F$
= $\arg\min_{Q} ||QA||_F^2 + ||B||_F^2 - 2\langle QA, B \rangle_F$
= $\arg\min_{Q} ||A||_F^2 + ||B||_F^2 - 2\langle QA, B \rangle_F$
= $\arg\max_{Q} \langle Q, BA^T \rangle_F$
= $\arg\max_{Q} \langle Q, U\Sigma V^T \rangle_F$
= $\arg\max_{Q} \langle U^T QV, \Sigma \rangle_F$
= $\arg\max_{Q} \langle S, \Sigma \rangle_F$ where $S = U^T QV$,

where S is an orthogonal matrix and Σ is a diagonal matrix. Then S should equal I_d so as to maximize the last objective. That is $S = U^T Q^* V = I_d$, which means the solution is $Q^* = UV^T$. \Box

Corollary 1. *Consider the following projection problem:*

$$
Q^* = \arg\min_Q \|Q - B\|_F \quad \text{s.t.} \quad Q^T Q = I_d,\qquad(19)
$$

if $B = U \Sigma V^T$ is the eigen-decomposition of B, then:

$$
Q^* = UV^T. \tag{20}
$$

Proof. Incorporate $A = I_n$ into Lemma 1, the conclusion can be immediately achieved. \Box Theorem 5. *(Corresponding to Theorem* 2 *in the main text) GraphAIN can be viewed as an iterative process in order*

to solve the following restricted optimization problem via Projected Gradient Ascent method:

$$
\max_{H} f(H) = \frac{1}{2} \cdot \text{tr}\left(H^{T} \bar{A} H\right), \quad s.t. \ H^{T} H = I_{d},\tag{21}
$$

where H is initialized to input graph signals $X \in \mathbb{R}^{n \times d}$ and $\eta = 1$ *is set as the step size.*

Proof. We consider the following equivalent objective via adding an additional constant:

$$
H^* = \arg\max_{H} f(H) = \frac{1}{2} \cdot \text{tr}\left(H^T\left(\bar{A} - I_n\right)H\right). \quad (22)
$$

Then we take the derivative of $f(H)$:

$$
\frac{\partial f(H)}{\partial H} = (\bar{A} - I_n) H = \bar{A}H - H.
$$
 (23)

Projected Gradient Ascent includes two steps in every iteration: First, we use original gradient ascent, that is:

$$
H' = H + \eta \cdot \frac{\partial f(H)}{\partial H},\tag{24}
$$

where η is step-size, H is the old H, H' is the new one. If we use $\eta = 1$, then we get the following equation:

$$
H' = H + \frac{\partial f(H)}{\partial H} = H + \bar{A}H - H = \bar{A}H.
$$
 (25)

Second, since H' might not satisfy the constraint, we project it to the set satisfying the constraint:

$$
H'' = H' (H'^T H')^{-\frac{1}{2}}.
$$
 (26)

The projection operation makes sense because H'' is the nearest orthogonal matrix satisfying the constraint around H′ . To see this, we do singular value decomposition for $H' = U\Lambda V^T$, then:

$$
H'' = H'\left(H'^T H'\right)^{-\frac{1}{2}}
$$

= $U\Lambda V^T \left(V\Lambda U^T U\Lambda V^T\right)^{-\frac{1}{2}}$
= $U\Lambda V^T V\Lambda^{-1} V^T$
= UV^T . (27)

Thus according to Corollary 1, we know that H'' can be seen as a projection of H' onto the set consists of all orthogonal matrixs. \Box

Theorem 6. *(Corresponding to Theorem* 3 *in the main text) Residual-favored GraphAIN can be viewed as an iterative process of solving the following restricted optimization problem via Projected Gradient Ascent method:*

$$
\max_{H} f(H) = \frac{1}{2} \cdot \text{tr}\left(H^{T} \bar{A} H\right) - \frac{1}{2} \cdot \frac{c}{a} \cdot \|H - X\|_{F}^{2}
$$
\ns.t.

\n
$$
H^{T} H = I_{d},
$$
\n(28)

where H is initialized to input graph signals $X \in \mathbb{R}^{n \times d}$ and *set the step size* $\eta = a \in [0, 1]$ *.*

Cora Method	\overline{c}	8	16	32	64
GCN	81.45 ± 0.37	72.72 ± 2.37	63.40 ± 4.40	31.90 ± 0.70	27.56 ± 3.36
SGC	79.00±0.46	78.02±1.06	75.26 ± 1.06	63.84 ± 3.63	55.39 ± 2.51
ChebNet	80.33 ± 1.19	78.70±1.51	68.40 ± 1.41	31.90 ± 1.77	20.63 ± 5.81
GAT	82.40 ± 0.05	78.30±0.47	76.50±0.34	72.28 ± 0.61	31.92 ± 0.04
GIN	80.54 ± 0.36	76.74±192	68.72 ± 1.30	60.92 ± 3.10	31.90 ± 0.00
PairNorm	78.30 ± 1.33	71.06 ± 1.74	66.80 ± 2.71	65.00 ± 3.74	66.24 ± 1.58
GCNII	82.19 ± 0.77	84.23 ± 0.42	84.69 ± 0.51	85.29 ± 0.47	85.34 ± 0.32
JKNet	79.06±0.11	75.66 ± 0.38	72.97±3.94	73.23 ± 3.59	72.54 ± 3.65
GPRGNN	82.53 ± 0.49	84.19 ± 0.40	83.69 ± 0.55	83.13 ± 0.60	82.48 ± 0.26
DAGNN	80.30 ± 0.78	84.28 ± 0.59	84.14 ± 0.59	83.39 ± 0.59	82.16 ± 0.35
APPNP	82.06 ± 0.46	83.59 ± 0.40	83.64 ± 0.48	83.68 ± 0.48	83.66 ± 0.36
BernNet	81.88±0.55	83.44 ± 1.06	82.44 ± 0.40	81.38 ± 0.44	17.72 ± 6.44
Ours(GCN)	84.20 ± 0.12	85.22 ± 0.20	84.96 ± 0.21	85.12 ± 0.15	84.60±0.29
Ours(GAT)	84.62 ± 0.08	84.92 ± 0.26	84.44 ± 0.30	84.60 ± 0.31	84.68 ± 0.31
Ours(GIN)	84.24±0.33	83.96 ± 0.35	84.08 ± 0.41	84.18 ± 0.58	83.80 ± 0.16

Table 5: Accuracy comparison of node classification tasks on Cora

Citeseer Method	\mathfrak{D}	8	16	32	64
GCN	69.46 ± 0.29	57.74 ± 6.30	48.70 ± 3.48	36.66±8.61	25.40 ± 2.34
SGC	67.92 ± 0.85	68.42 ± 0.46	68.08 ± 0.73	67.50 ± 1.43	63.08 ± 0.29
ChebNet	69.27 ± 0.28	63.20 ± 1.92	58.73 ± 2.62	33.43 ± 3.10	24.90 ± 1.25
GAT	71.38±0.04	64.69 ± 0.27	62.20 ± 0.25	59.08±0.44	22.90 ± 2.23
GIN	68.02 ± 0.41	59.80±0.79	53.55 ± 1.98	47.32 ± 3.90	23.10 ± 0.00
PairNorm	65.80 ± 1.35	54.81 ± 6.46	46.26 ± 2.69	$44.20 + 1.23$	41.48 ± 4.63
GCNII	67.81 ± 0.89	70.62 ± 0.63	72.97 ± 0.71	73.24±0.78	73.00 ± 0.75
JKNet	66.98 ± 1.82	60.56 ± 1.41	54.33 ± 7.74	50.68 ± 8.73	52.22 ± 6.99
GPRGNN	70.49±0.95	71.47 ± 0.58	71.39 ± 0.73	71.01±0.79	70.96 ± 0.38
DAGNN	70.91 ± 0.68	72.44 ± 0.54	73.05 ± 0.62	72.59±0.54	71.00 ± 0.55
APPNP	71.67±0.78	72.04 ± 0.52	72.13 ± 0.53	72.13±0.59	72.02 ± 0.46
BernNet	72.02±0.53	71.44 ± 0.22	70.70±0.91	70.82±0.56	39.10±12.92
Ours(GCN)	74.84±0.32	74.72 ± 0.26	74.12±0.08	74.42±0.26	74.50 ± 0.23
Ours(GAT)	74.82±0.36	74.24 ± 0.51	74.24 ± 0.47	74.26±0.11	74.04±0.19
Ours(GIN)	74.58±0.81	74.22 ± 0.08	74.04±0.33	74.88±0.31	74.16±0.29

Table 6: Accuracy comparison of node classification tasks on Citeseer

For convenience, we slightly abuse the symbols used in the main text.

Proof. Consider the following generalized objective:

$$
f(H) = \frac{1}{2} \cdot \text{tr}\left(H^T\left(\bar{A} - p \cdot I_n\right)H\right) - \frac{1}{2} \cdot \beta \cdot ||H - X||_F^2.
$$
\n(29)

Then we take the derivative of $f(H)$:

$$
\frac{\partial f(H)}{\partial H} = (\bar{A} - p \cdot I_n) \cdot H - \beta \cdot (H - X) \tag{30}
$$

$$
= \bar{A} \cdot H - (\beta + p) \cdot H + \beta \cdot X.
$$

We apply the SGD updating formulation with step size η as follows:

$$
H' = H + \eta \cdot \frac{\partial f(H)}{\partial H}
$$

= $H + \eta \cdot (\bar{A} \cdot H - (\beta + p) \cdot H + \beta \cdot X)$ (31)
= $\eta \cdot \bar{A} \cdot H + (1 - \eta \cdot (\beta + p)) \cdot H + \eta \cdot \beta \cdot X.$

Then let the coefficients equal to a, b and c , respectively, we can get the following equations:

$$
\begin{cases}\n\eta = a \\
1 - \eta \cdot (\beta + p) = b \\
\eta \cdot \beta = c.\n\end{cases}
$$
\n(32)

Thus, we achieve:

$$
\begin{cases}\n\eta = a \\
\beta = \frac{c}{a} \\
p = \frac{1 - b - c}{a}.\n\end{cases}
$$
\n(33)

Then Eq. 31 can be transformed into:

$$
H' = a \cdot \bar{A}H + b \cdot H + c \cdot X. \tag{34}
$$

This is exactly the propagation rule in *Residual-favored GraphAIN*.

Moreover, according to the restriction $a + b + c = 1$, we can determine the value of p as follows:

$$
a + b + c = \eta + (1 - \eta \cdot (\beta + p)) + \eta \cdot \beta
$$

= $(1 - p) \cdot \eta + 1 = 1,$ (35)

which means $p = 1$ is true.

Then let $\beta = \frac{c}{a}$ and $p = 1$, and the objective Eq. 29 becomes:

$$
f(H) = \frac{1}{2} \cdot \text{tr}\left(H^T\left(\bar{A} - I_n\right)H\right) - \frac{1}{2} \cdot \frac{c}{a} \cdot \left\|H - X\right\|_F^2. \tag{36}
$$

This equation is equivalent to Equ. 28 due to the condition that $H^TH = I_d$, *i.e.*, $\text{tr}\left(H^TH\right) = \|H\|_F^2 = d$.

Note that the step size $\eta = a$, and the projection formulation of *Residual-favored GraphAIN* is the same as *GraphAIN* (see Theorem 2), which completes the proof. \Box

G Experimental Details

In this section, details of more experiments were provided here to further evaluate the performance of our model as complementary to the experiments in the main text.

G.1 Dataset Descriptions

Dataset statistics including their splits are summarized in Tab. 4. In (Chen et al. 2022), a detailed introduction can be found including all twelve homogeneous and heterophilous graphs.

G.2 More experimental results

In this subsection, to further evaluate our model we report more experimental results including comparative experiments on homogeneous and heterophilous graphs with more layers, ablation studies with specific standard errors, and results in node classification tasks with noisy features. Some results of layers 10^3 and 10^4 are also reported for reference in tasks with noisy features (see next subsection for details), where our models are always implemented based on SGC.

- Common semi-supervised node classification tasks on homogeneous graphs: Tab. 5, Tab. 6, Tab. 7, Tab. 8, Tab. 9;
- Common semi-supervised node classification tasks on heterophilous graphs: Tab. 10, Tab. 11;
- Ablation studies: Tab. 12, Tab. 13, and Tab. 14;
- Node classification tasks on a large-scale graph OGBN-ArXiv in its public fixed split: Tab. 15;
- Node classification tasks in fully supervised settings (average results with multiple random $60\%/20\%/20\%$ splits following (Chen et al. 2020)) on three citation graphs: Tab. 16;
- Results of node classification tasks on heterophilous graphs compared to more state-of-the-art counterparts (including some models specifically designed for heterophilous graphs): Tab. 17;
- Node classification tasks with noisy features: Tab. 18, Tab. 19, Tab. 20.

G.3 Implementation details

In this subsection, we give some details on the implementations of our models against different tasks.

For seven homogeneous graphs in common semisupervised node classification tasks, we adopt Our model itself based on GCN as a teacher model for fair comparison (*i.e.*, we don't use other pre-trained models) and we build the auxiliary graphs using $G_{aux} = G_{emd}$ because we think G_{emd} can be a rough estimation of similarities of groundtruth labels of those nodes in the input graph.

For four heterophilous graphs, we adopt different implementations for adaptation to these specific scenes. Specifically, we build and aggregate in the auxiliary graphs G_{aux} = G_{fea} and also employ Ours as the teacher to teach Ours itself for fairness. We choose G_{fea} instead of G_{emd} because we assume that on these heterophilous graphs, features are information more related to ground-truth labels of nodes compared to their noisy structures.

Pubmed Method	\mathcal{L}	8	16	32	64
GCN	77.40 ± 0.37	77.24 ± 0.71	70.02 ± 4.23	44.22 ± 0.93	32.65 ± 1.15
SGC	77.50±0.66	70.90 ± 0.59	71.34 ± 0.09	70.70 ± 0.32	65.33 ± 0.52
ChebNet	78.53±0.34	77.63 ± 0.43	72.87 ± 0.57	48.67 ± 3.20	45.37 ± 0.57
GAT	77.62 ± 0.10	77.39 ± 0.09	76.28 ± 0.15	78.72±0.22	41.88±1.78
GIN	77.54±0.30	75.84 ± 1.18	72.69 ± 3.45	72.94±4.37	40.23 ± 0.31
PairNorm	75.50±0.41	74.82 ± 1.03	74.34 ± 0.68	72.12 ± 3.01	71.72 ± 3.15
GCNII	78.05±1.53	79.34 ± 0.51	80.03 ± 0.50	79.81±0.27	79.88±0.17
JKNet	77.24±0.92	76.92 ± 1.03	64.37 ± 8.80	$63.77 + 9.21$	69.10 ± 7.33
GPRGNN	78.73±0.63	78.90±0.47	78.78±1.02	78.46±1.03	78.92±1.45
DAGNN	77.74±0.57	79.68±0.37	80.32 ± 0.38	80.58 ± 0.51	80.44 ± 0.46
APPNP	79.46±0.47	80.02 ± 0.30	80.30 ± 0.30	80.24 ± 0.33	80.08 ± 0.35
BernNer	79.12±0.66	78.32 ± 0.79	77.92 ± 0.79	70.24 ± 10.07	32.86 ± 8.26
Ours(GCN)	81.18±0.19	80.96 ± 0.15	81.14 ± 0.17	81.28 ± 0.18	81.58 ± 0.66
Ours(GAT)	80.64 ± 0.25	80.34 ± 0.27	80.16 ± 0.18	80.46 ± 0.34	80.20 ± 0.24
Ours(GIN)	81.28 ± 0.15	80.70 ± 0.39	80.72 ± 0.24	80.32 ± 0.24	80.94 ± 0.36

Table 7: Accuracy comparison of node classification tasks on Pubmed

Method		CoauthorCS		CoauthorPhysics				
#Layers	16	32	64	16	32	64		
GCN	53.19 ± 7.23	41.29 ± 5.11	34.23 ± 8.31	85.23 ± 2.18	79.87±3.86	75.34 ± 1.12		
SGC	71.75 ± 3.65	70.52 ± 3.96	72.51 ± 0.89	92.34 ± 0.20	91.46 ± 0.48	90.77 ± 0.67		
ChebNet	48.43 ± 8.82	29.28 ± 6.85	23.24 ± 0.00	77.62 ± 2.09	70.35 ± 3.41	50.74 ± 0.00		
GAT	85.32 ± 0.22	85.85 ± 0.22	12.83 ± 2.58	91.84 ± 0.16	91.87 ± 0.11	17.75 ± 1.53		
GIN	70.77 ± 3.02	52.90 ± 3.87	20.79 ± 5.21	85.22 ± 2.61	83.02 ± 2.53	24.98 ± 10.76		
PairNorm	75.17 ± 5.15	72.71 ± 3.21	68.62 ± 6.79	90.18 ± 1.17	88.51 ± 0.95	89.11 ± 1.49		
GCNII	58.94 ± 2.63	71.67 ± 2.68	72.11 ± 4.19	92.13 ± 1.31	93.15 ± 0.92	92.79 ± 0.52		
JKNet	81.31 ± 3.21	81.82 ± 3.32	82.84 ± 3.15	91.24 ± 0.97	90.92 ± 1.61	89.88±1.99		
GPRGNN	89.39 ± 0.39	89.56 ± 0.47	89.33 ± 0.62	93.64 ± 0.31	93.49 ± 0.59	93.26 ± 0.46		
DAGNN	91.13 ± 0.50	89.60 ± 0.71	89.47 ± 0.68	93.77 ± 0.29	93.31 ± 0.60	93.52 ± 0.32		
APPNP	91.64 ± 0.53	91.61 ± 0.49	91.58 ± 0.36	93.96 ± 0.36	93.75 ± 0.61	91.61 ± 0.33		
BernNet	91.53 ± 0.57	91.54 ± 0.24	9.19 ± 7.43	91.51 ± 0.76	92.67 ± 0.63	19.38 ± 0.99		
Ours(GCN)	92.47 ± 0.07	92.50 ± 0.09	92.41 ± 0.07	94.51 ± 0.04	94.45 ± 0.06	94.52 ± 0.04		
Ours(GAT)	91.32 ± 0.06	91.30 ± 0.07	91.26 ± 0.03	94.00 ± 0.08	94.02 ± 0.05	94.02 ± 0.12		
Ours(GIN)	92.06 ± 0.21	91.79 ± 0.06	91.71 ± 0.04	94.44±0.04	94.56 ± 0.12	94.53 ± 0.16		

Table 8: Node classification accuracy comparison on CS and Physics

Additionally, for node classification tasks with noisy features, we maintain the basic settings in the common semisupervised node classification tasks (*e.g.*, we use the same splits for all benchmarks). We employ G_{ori} as the auxiliary graph and also use Ours as a teacher similar to the other scenes. But here we utilize SGC-based implementations for our models because we hope to conveniently implement models with 10^3 and 10^4 layers. More specifically, we first use a simple linear layer for efficient dimension reduction and use the non-parametric SGC with L layers for propagation and finally use k-layer MLP to do feature transformation due to limited non-linearity where $k \in [l, 5]$ is a hyper-parameter chose via their validation sets. For layers no more than 64, we directly train Our model in an end-to-end manner.

But for deeper layers (*e.g.*, 10^3 and 10^4 layers), we tend to stop the gradient and avoid building the computation graphs. Thus this trick can efficiently reduce the time for training and the out-of-memory issue, though the expressivity is restricted. In other words, it's a trade-off between runtime, space complexity, and nonlinearity. So to supplement non-linearity, we tend to use an MLP instead of a simple linear layer.

We employ a 3090 GPU (with 24G Memory) to run other models due to possible OOM issues while a 2080Ti GPU (with 12G Memory) for ours. *Pairnorm* means GCN-based Pairnorm (Zhao and Akoglu 2020). For layers more than 64 (*i.e.*, 10^3 and 10^4 layers), the motivation for them is as follows: 1) to show SGC can unavoidably suffer from oversmoothness when the layers are deep enough; 2) to show our model will not suffer from over-smoothness even with a sufficiently large depth, conforming to the theoretical analysis.

Because of the high time and space complexity for our model with $10³$ layers and above, we do not give the comparison results for the case. Anyhow, by compromising the performance and reducing the time and space complexity, we present those comparison results in the scenarios with noisy features at 10^3 and 10^4 layers. We trust those comparisons can similarly carry over to the non-noisy scenarios and achieve even better accuracy.

G.4 Hyper-parameter Configurations

It's difficult to well-tune all the hyper-parameters simultaneously, so we do it in three stages: 1) Tune hyper-parameters in *R-SoftGraphAIN* but with all other hyper-parameters fixed; 2) Tune hyper-parameters in *Curriculum Learning* but with all other hyper-parameters fixed; 3) fine-tune other hyperparameters excepting that of *R-SoftGraphAIN* and *Curriculum Learning* (*e.g.*, learning rate, dropout rate, and weight decay).

In every stage above, we choose the hyper-parameters according to their best validation performance. The search spaces of all meaningful hyper-parameters are listed in Tab. 22, and we omit some unimportant ones because they are actually robust to model performance. For every kind of experiment, we search in the space randomly for 500 times in total (*i.e.*, the sum of all layers). Additionally, we report all hyper-parameters configurations used by *Ours(GCN)* for comparative experiments in Tab. 21. Note that the exactly same hyper-parameters are employed for *Ours(GAT)* and *Ours(GIN)* to reduce the burden of computational resources. In other words, with further appropriate hyper-parameter tuning, these two versions might have more outstanding performance, possibly due to the fact that the encoders (*i.e.*, GAT and GIN) are more powerful than GCN itself.

In addition, note that in our experiments, we do not directly use the hyper-parameters α, β, γ in Tab. 21. In fact, we use their values after normalization, *i.e.*, letting $\alpha = \alpha/(\alpha + \beta + \gamma) \in [0,1]$ in order to satisfy the rule $\alpha + \beta + \gamma = 1$ in the main text (see Sec. 3.1).

H Extensive Visualizations

In this section, we plot the original features, embeddings output by SGC, and embeddings output by our model on eleven real-world graph benchmarks respectively in Fig. 6, Fig. 7, and Fig. 8. These demonstrate that our model obtains clearer clustering boundaries compared to SGC, which conforms to the fact that our model achieves a performance gain in comparison.

Moreover, Fig. 3 gives visualizations of the embeddings produced by different deep models with synthetic data that is generated as follows: first choose several cluster centers ${c_i}$ in \mathbb{R}^3 , then sample points in $\mathcal{N}(c_i, I)$, where colors and edges are created following Bernoulli distributions to ensure that points in the same class tend to share a link and the same color. These visualizations show that our model obtains clearer decision boundaries compared to the baseline models including SGC and Pairnorm, conforming to the performance gain of our model.

I Discussion on the Time and Space **Complexities**

The theoretical time complexity is analysed $O(Ld^2d_0)$ where $d, d_0 \ll n$ if partial-SVD or truncated-SVD (Halko, Martinsson, and Tropp 2011) is utilized. And it would become $O(Ld^3)$ with a full SVD decomposition. However, it is efficient under the high-parallelizability implemented via Pytorch. The theoretical space complexity is $O(L(n+d) d)$.

J Additional Hyper-parameter Studies

In this section, to help readers better understand the properties of our method, we further conduct some additional hyperparameter studies on the hyper-parameter α , γ , and k_{KNN} , which we think are somewhat important in our framework. We study these hyper-parameters based on all versions, *i.e.*, Ours(GCN), Ours(GAT), and Ours(GIN). We visualize the line graph considering the effects of them on the final results (see Fig. 4 and Fig. 5).

Fig. 4 illustrates how the trade-off between residual connections and initial connections influences the final performance, considering fixing the hyper-parameter β . Note that studying the hyper-parameter γ is equivalent to studying the hyper-parameter α in this situation. From this figure, we can see that in some datasets, γ should not be too small; otherwise, the performance would decline or fluctuate. However, a too-high value of γ also hurt the performance in some cases (*e.g.*, nearly all versions on Cora and Ours(GAT) on Pubmed). Therefore, we had better keep its value ranging between 0.3 and 0.7 on Cora, Citeseer, and Pubmed, because it seems

Method		AmazonComputers		AmazonPhoto			
#Layers	16	32	64	16	32	64	
GCN	64.02 ± 2.38	58.30 ± 3.35	37.58 ± 0.05	70.81 ± 3.48	58.47 ± 8.80	50.21 ± 3.99	
SGC	37.48±0.07	37.44 ± 0.12	37.50 ± 0.08	35.64 ± 6.11	26.08 ± 1.39	24.57 ± 1.32	
ChebNet	65.42 ± 4.04	58.58±3.29	50.12 ± 2.15	69.63 ± 6.90	65.28 ± 3.55	$64,83\pm0.72$	
GAT	76.90±0.49	76.05±0.20	37.18 ± 0.53	89.27 ± 0.29	83.73 ± 0.74	25.36 ± 0.20	
GIN	71.91 ± 2.71	39.18±3.77	37.50 ± 0.09	82.25 ± 2.04	65.98 ± 2.14	25.27 ± 0.13	
PairNorm	77.41 ± 1.85	74.96±2.09	74.35 ± 1.82	82.72 ± 1.19	82.66 ± 2.47	79.55 ± 2.51	
GCNII	38.88±4.26	37.56 ± 0.43	37.50 ± 0.08	68.37 ± 6.61	62.95 ± 9.41	65.12 ± 2.86	
JKNet	60.76 ± 5.10	67.99 ± 5.07	$67.78{\pm}4.79$	74.86±8.45	78.42±6.95	79.73±7.26	
GPRGNN	76.07 ± 2.28	41.94 ± 9.95	78.30 ± 2.51	91.55 ± 0.43	91.74 ± 0.81	91.28 ± 0.88	
DAGNN	80.33 ± 1.04	79.73 ± 3.63	79.23 ± 2.36	90.81 ± 0.59	89.96 ± 1.16	87.86 ± 0.70	
APPNP	39.41 ± 5.80	43.02 ± 10.16	41.42 ± 7.50	64.59 ± 20.09	59.62 ± 23.27	63.63 ± 19.28	
BernNet	84.23 ± 1.62	81.86 ± 1.63	12.81 ± 12.10	89.17 ± 1.91	91.59 ± 0.15	16.53 ± 5.78	
Ours(GCN)	85.24 ± 0.05	85.21 ± 0.11	85.13 ± 0.08	91.98 ± 0.09	92.06 ± 0.19	92.03 ± 0.12	
Ours(GAT)	84.88±0.14	84.82±0.27	85.04 ± 0.15	92.04 ± 0.07	91.98 ± 0.07	92.00 ± 0.05	
Ours(GIN)	85.30 ± 0.13	85.16 ± 0.12	85.13 ± 0.08	92.00 ± 0.03	92.03 ± 0.05	91.98 ± 0.04	

Table 9: Node classification accuracy comparison on Computers and Photo

somewhat stable in this range. And definitely, we can specifically and carefully tune it according to the validation sets of all these datasets.

From Fig. 5, we can clearly observe a relatively stable line trend between the hyper-parameter k_{KNN} and the model performance. In most cases, we think the range [5, 7] is quite appropriate despite some exceptions. However, according to this line graph, we generally think that it is not necessary to carefully select or tune this hyper-parameter, though one can definitely do it.

K Discussion On The Connections Between R-SoftGraphAIN and SmoothCurriculum

In this section, to help readers better understand our method, we briefly discuss the connection of our proposed two parts, *i.e.*, *R-SoftGraphAIN* and *SmoothCurriculum*. At first glimpse, the proposed two parts have little essential relationship, because both of them can be applied to different graph encoders to improve their performance. But under the intrinsic design and basic motivation on facilitating a better deep GNN model, they can be parts that get along with each other quite well, and thus can be viewed as a whole part. Specifically, one can easily notice that, according to its properties, R-SoftGraphAIN can preserve knowledge into d (or d_0 , more accurately) different channels and organize them well, *e.g.*, the information in different channels would not intervene with each other. With the aid of it, GNN encoders can definitely improve performance when layers go deep. However, without the help of curriculum learning, the supervised knowledge (*i.e.*, label information) absorbed by this module keeps constantly the same. Therefore, it would organize this knowledge adaptively and in its own way, which is not quite understandable and depends on both the feature and the input graph structure. But under the guide of the proposed curriculum learning framework, it will be gradually fed with the knowledge that is from easy to hard, from low-frequency to high-frequency, from global to local, and from generalizable

to relatively noisy or unnecessary. So fortunately, it has some opportunities to load different knowledge hierarchies into different channels and intuitively avoid some unnecessary forgetting. For example, it could employ the first several channels to differentiate relatively larger communities, and use additional channels to more accurately locate neighborhoods and nodes themselves. Besides, to reduce the negative influence of noise, it can stop learning early when necessary. On the other hand, if using the curriculum learning framework individually and, *e.g.*, applying it to GCN, GCN might have only one channel, and thus the new knowledge might replace the old one and the model tends to forget some basic yet important and generalizable label information, assuming the graph is connected. But with the help of R-SoftGraphAIN, it might help organize this information well. That's why we think these two parts can be viewed as a whole to significantly improve the model's performance.

Figure 6: Scatter plots of original features and embeddings output via 32-layer SGC and our model on real-world benchmarks Cora, Citeseer, and Pubmed

Figure 7: Scatter plots of original features and embeddings output via 32-layer SGC and our model on real-world benchmarks CS, Physics, Computers, and Photo

Figure 8: Scatter plots of original features and embeddings output via 32-layer SGC and our model on real-world benchmarks Texas, Wisconsin, Cornell, and Actor

Method		Texas			Wisconsin	
#Layers	16	32	64	16	32	64
GCN	62.16 ± 2.70	62.16 ± 2.70	62.16 ± 2.70	57.84±4.90	57.84 ± 4.90	57.84±4.90
SGC	62.03 ± 0.59	56.41 ± 4.25	56.96 ± 5.28	52.25 ± 7.19	51.29 ± 6.44	52.16 ± 7.08
ChebNet	69.37 ± 1.56	64.86 ± 0.00	64.86 ± 0.00	71.90 ± 3.00	52.94 ± 0.00	52.94 ± 0.00
GAT	64.32 ± 2.96	65.41 ± 1.21	64.86 ± 0.00	53.33 ± 1.64	53.73 ± 1.07	53.33 ± 0.88
GIN	52.97 ± 6.78	$62.17+4.44$	60.00 ± 4.01	47.06 ± 3.67	47.06 ± 2.40	50.98 ± 4.38
PairNorm	32.70 ± 17.41	41.08 ± 18.04	40.68 ± 16.71	44.51 ± 11.33	52.84 ± 8.97	52.94 ± 11.35
GCNII	69.59 ± 6.10	69.19 ± 6.56	65.41 ± 2.02	71.86 ± 1.89	$70.31 + 4.75$	59.02 ± 0.78
JKNet	65.41 ± 3.03	61.08 ± 6.23	66.49 ± 2.76	55.98 ± 3.53	52.76 ± 5.69	56.08 ± 4.04
GPRGNN	62.70 ± 2.65	$62.27 + 4.97$	61.08 ± 1.99	67.94 ± 3.58	71.35 ± 5.56	64.90 ± 2.77
DAGNN	61.08 ± 1.99	57.68 ± 5.07	60.27 ± 2.43	55.49 ± 3.78	50.84 ± 6.62	51.76 ± 4.78
APPNP	64.46 ± 3.11	$60.68{\pm}4.50$	64.32 ± 2.78	59.31 ± 3.71	54.24 ± 5.94	59.90 ± 3.00
BernNet	76.22 ± 2.02	61.08 ± 1.32	16.76 ± 7.33	80.00 ± 2.88	63.53 ± 4.74	24.71 ± 7.60
Ours(GCN)	85.95 ± 1.21	85.41 ± 1.48	84.86 ± 1.48	82.35 ± 1.39	83.14 ± 1.75	84.71 ± 0.88
Ours(GAT)	79.46±1.48	78.92±4.83	78.38±1.91	72.94±4.25	73.73 ± 2.97	74.90±1.64
Ours(GIN)	82.70 ± 1.58	82.57 ± 1.48	83.12 ± 1.33	82.35 ± 1.39	81.78 ± 2.15	81.20 ± 1.21

Table 10: Node classification accuracy comparison on heterophilous graphs: Texas and Wisconsin

Method		Cornell			Actor	
#Layers	16	32	64	16	32	64
GCN	56.76 ± 2.70	56.76 ± 2.70	56.76 ± 2.70	25.16 ± 0.30	25.16 ± 0.30	25.16 ± 0.30
SGC	55.41 ± 1.35	58.57 ± 3.44	55.41 ± 1.35	25.88 ± 0.49	26.17 ± 1.15	25.88 ± 0.49
ChebNet	55.86±1.56	55.86 ± 1.56	52.25 ± 1.56	34.71 ± 0.11	25.46 ± 0.00	25.46 ± 0.00
GAT	53.51 ± 2.26	54.05 ± 0.00	55.14 ± 1.48	26.70 ± 0.13	25.54 ± 0.09	25.62 ± 0.13
GIN	55.68 ± 2.42	54.59 ± 1.21	55.14 ± 1.48	24.78 ± 1.08	24.36 ± 2.48	23.45 ± 2.73
PairNorm	35.95 ± 16.77	36.89 ± 18.63	40.68 ± 12.89	22.84 ± 2.29	24.33 ± 1.60	23.23 ± 2.84
GCNII	66.49 ± 5.88	74.16 ± 6.48	56.92 ± 2.02	33.79 ± 0.68	34.28 ± 1.12	34.64 ± 0.71
JKNet	50.27 ± 3.95	57.30±4.95	$51.49{\pm}4.40$	29.23 ± 0.65	28.80 ± 0.97	28.26 ± 0.43
GPRGNN	53.11 ± 3.45	58.27 ± 3.96	52.16 ± 3.74	32.83 ± 0.88	29.88±1.82	32.43 ± 0.49
DAGNN	55.27 ± 2.49	58.43 ± 3.93	52.43 ± 5.01	27.66 ± 0.67	27.73 ± 1.08	25.45 ± 0.64
APPNP	56.35 ± 2.46	58.43 ± 3.74	54.69 ± 2.55	28.38 ± 0.79	28.65 ± 1.28	28.19 ± 0.97
BernNet	72.97±1.71	52.43 ± 9.46	11.89 ± 1.32	36.31 ± 0.39	28.19 ± 1.10	20.66 ± 3.10
Ours(GCN)	82.16 ± 1.48	82.70 ± 1.48	82.62 ± 1.21	38.16 ± 0.15	38.42 ± 0.23	38.49 ± 0.34
Ours(GAT)	81.62 ± 2.26	77.84 ± 2.26	81.62 ± 1.21	35.04 ± 0.42	34.97±0.34	35.54 ± 0.61
Ours(GIN)	80.00 ± 1.48	81.08 ± 1.91	81.08 ± 1.91	34.46 ± 0.51	34.51 ± 0.27	34.50±0.61

Table 11: Node classification accuracy comparison on heterophilous graphs: Cornell and Actor

Method	Cora			Citeseer	Pubmed		
#Layers	32	64	32	64	32	64	
w.o. SG	83.30 ± 0.12	84.60 ± 0.29	72.98 ± 0.50	72.62 ± 0.98	80.26 ± 0.98	80.30 ± 0.23	
W_{0} . RC	82.88±0.79	82.40 ± 0.33	72.82 ± 1.01	71.04 ± 0.55	78.84±0.59	78.60 ± 0.27	
$w.o. R-SG$	$40.22 + 5.71$	36.74 ± 4.36	29.32 ± 3.47	27.70 ± 2.94	46.28 ± 4.91	28.61 ± 3.64	
$W.0.$ LS	83.96 ± 0.65	84.00 ± 0.22	73.64 ± 0.68	74.34 ± 0.79	81.04 ± 0.21	80.86 ± 0.79	
$W, O, \mathcal{C}L$	82.40 ± 0.65	82.58 ± 0.87	73.20 ± 0.77	72.54 ± 0.42	79.34 ± 0.39	79.00 ± 0.54	
Ours(GCN)	85.12 ± 0.15	84.87 ± 0.26	74.42 ± 0.26	74.50 ± 0.23	81.28 ± 0.18	81.58 ± 0.66	

Table 12: Ablation Studies on Cora, Citeseer, and Pubmed

Method	CoauthorCS		CoauthorPhysics				
#Layers	32	64	32	64			
w.o. SG	91.79 ± 0.07	91.70 ± 0.07	94.23 ± 0.25	94.25 ± 0.13			
W, O, RC	91.40 ± 0.26	89.03 ± 0.62	92.99 ± 0.19	92.59 ± 0.63			
$w.o. R-SG$	51.61 ± 18.23	28.51 ± 4.70	77.20 ± 11.95	60.94 ± 8.23			
$W.0.$ LS	92.02 ± 0.05	91.95 ± 0.08	94.10 ± 0.11	94.06 ± 0.08			
$W, O, \mathcal{C}L$	89.60 ± 0.20	89.23 ± 0.12	92.46 ± 0.61	92.22 ± 0.56			
Ours(GCN)	92.50 ± 0.09	92.41 ± 0.07	94.45 ± 0.06	94.52 ± 0.04			

Table 13: Ablation Studies on CS aand Physics

Method		AmazonComputers		AmazonPhoto
#Layers	32	64	32	64
w.o. SG	82.34 ± 0.67	84.50 ± 0.28	90.26 ± 1.57	91.69 ± 0.37
W_0 . RC.	83.78 ± 0.52	84.25 ± 0.37	91.42 ± 0.32	91.14 ± 0.48
w_0 . R-SG	61.52 ± 8.27	$49.94 + 9.69$	$83.77 + 9.52$	70.09 ± 12.39
W_0 . LS	84.52 ± 0.93	84.91 ± 0.16	91.50 ± 0.30	91.64 ± 0.40
$W.0. \overline{CL}$	84.00 ± 0.97	84.44 ± 0.56	90.58 ± 0.94	90.41 ± 1.07
Ours(GCN)	85.21 ± 0.11	85.13 ± 0.08	92.06 ± 0.19	92.03 ± 0.12

Table 14: Ablation Studies on Computers and Photo

Method		OGBN-ArXiv
#Layers	32	64
GCN	46.38±3.87	42.95 ± 3.93
SGC	34.22±0.04	23.14 ± 7.85
ChebNet	41.00 ± 1.59	35.16 ± 2.23
GAT	59.78 ± 0.25	36.63 ± 1.71
GIN	65.17 ± 0.31	60.56 ± 0.57
PairNorm	63.32 ± 0.97	43.57+1.24
GCNII	72.60±0.25	70.07 ± 0.14
JKNet	66.31 ± 0.63	65.80 ± 0.27
GPRGNN	70.18±0.16	69.98 ± 0.15
DAGNN	71.46±0.27	70.58±0.12
APPNP	66.94 ± 0.26	66.90 ± 0.15
BernNet	45.16±0.33	37.18±2.52
Ours(GCN)	74.07±0.08	73.95±0.06
Ours(GAT)	74.37±0.07	74.41±0.05
Ours(GIN)	75.02±0.12	74.85±0.09

Table 15: Node classification accuracy comparison on the large-scale graph: OGBN-ArXiv

Method	Cora	Citeseer	Pubmed
GCN	52.53 ± 8.94	29.98 ± 6.25	48.75 ± 9.33
SGC	76.16 ± 0.27	69.62 ± 0.63	78.09±0.51
PairNorm	80.50 ± 0.19	63.34 ± 5.93	80.55 ± 3.16
GCNII	88.08±0.25	75.47 ± 0.32	89.53 ± 0.16
JKNet	85.05 ± 0.47	74.15 ± 0.70	89.25 ± 0.39
GPRGNN	87.47 ± 0.35	74.29 ± 0.48	90.61 ± 0.12
DAGNN	87.67 ± 0.39	73.82 ± 0.49	87.21 ± 0.13
APPNP	88.28 ± 0.63	75.76 ± 0.51	89.05 ± 0.11
Ours(GCN)	90.16 ± 0.33	77.23 ± 0.35	91.01 ± 0.12

Table 16: Results comparison of node classification tasks in multiple random splits (60%/20%/20%) on three citation graphs

Method	Texas	Wisconsin	Cornell	Actor
GAT (Veličković et al. 2018)	61.62	54.71	59.46	28.06
Gemo-GCN (Pei et al. 2019)	67.57	64.12	60.81	31.63
H2GCN (Zhu et al. 2020)	84.86	86.67	82.16	35.86
FAGCN (Bo et al. 2021)	84.32	83.33	81.35	35.74
GCA (Zhu et al. 2021)	59.46	50.78	55.41	29.65
BGRL (Thakoor et al. 2021)	59.19	52.35	57.30	29.86
VGAE (Kipf and Welling 2016)	59.20	56.67	59.19	26.99
FSGNN* (Maurya, Liu, and Murata 2022)	87.30	88.43	88.11	35.75
Ours(GCN)	85.95	84.71	82.70	38.49
$Ours(GCN)*$	94.74	88.24	91.05	37.73

Table 17: More result comparison on these heterophilous graphs (models with the suffix ∗ are evaluated in multiple random splits, *i.e.*, 60%/20%/20%)

Cora Method		4		12	16	24	32	64	1000	10000
GCN	54.8	66.1	72.2	70.2	65.6	56.6	56.0	31.9	31.9	31.9
SGC	55.0	65.2	72.1	75.1	75.0	77.0	75.6	75.6	16.8	15.7
GCNII	40.7	49.4	60.1	65.8	62.4	62.6	65.9	62.4	65.7	63.4
Pairnorm	56.6	60.6	63.2	67.9	70.5	70.8	61.9	35.5	54.4	31.8
Ours(GCN)	69.9	72.4	74.4	78.3	78.8	78.5	77.8	76.7	76.6	75.0

Table 18: Accuracy of different models with varying layers in node classification tasks with noisy features on Cora

Citeseer Method			8	12	16	24	32	64	1000	10000
GCN	36.1	45.4	49.4	48.4	47.6	43.3	43.8	23.3	18.1	18.1
SGC	37.9	44.0	51.1	52.3	54.0	53.0	55.2	56.0	47.0	22.4
GCNII	32.8	33.5	34.7	43.4	45.1	42.8	45.9	46.8	48.5	46.8
Pairnorm	37.5	43.4	48.2	47.7	46.9	44.3	38.9	34.8	34.4	34.4
Ours(GCN)	46.5	49.4	53.1	51.8	54.6	55.1	55.4	55.7	55.0	53.5

Table 19: Accuracy of different models with varying layers in node classification tasks with noisy features on Citeseer

Pubmed Method			8	12	16	24	32	64	1000	10000
GCN	40.4	48.0	57.3	61.2	60.4	59.4	57.3	44.8	40.7	OOM
SGC	40.6	47.6	62.3	69.2	71.2	74.6	74.9	78.6	34.1	34.1
GCNII	39.7	41.3	41.2	40.6	40.3	40.0	41.0	39.7	39.3	OM
Pairnorm	41.3	48.8	61.4	65.7	68.1	67.4	65.3	66.6	65.7	OOM
Ours(GCN)	45.7	52.7	70.1	69.8	73.9	76.2	78.0	78.5	71.0	71.0

Table 20: Accuracy of different models with varying layers in node classification tasks with noisy features on Pubmed

Table 21: Hyper-parameters Configurations of our model *Ours(GCN)* with 32 layers for node classification tasks on twelve public graph benchmarks. The exactly same hyper-parameters are employed for *Ours(GAT)* and *Ours(GI* Table 21: Hyper-parameters Configurations of our model *Ours(GCN)* with 32 layers for node classification tasks on twelve public graph benchmarks. The exactly same hyper-parameters are employed for *Ours(GAT)* and *Ours(GIN)* to reduce the burden of computational resources.

Hyper-parameter Names	Hyper-parameters Ranges
d	64,128,256
₫	$0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 5, 10, 100$
Ø	$0.01, 0.1, 0.3, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 5, 10, 100$
$\check{}$	$0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 5, 10, 100$
a	$0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99$
ς	1.0, 0.95, 0.9, 0.85, 0.8
n_T	5, 10, 20, 50, 100
σ	$0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99$
P	$0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99$
K_{knn}	1, 2, 3, 4, 5, 6, 7, 8
$\check{\mathcal{C}}$	$0.001, 0.01, 0.1, 0.3, 0.5, 0.7, 1, 2, 5, 10, 100$
d_0/d	$0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.91, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.98, 0.91$
F	0.1, 0.01, 0.05, 0.001, 0.005
lr_decay	50, 100, 150, 200, 250, 300
dropout	$0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$
weight-decay	$0, 10^{-4}, 5 \times 10^{-4}, 10^{-5}, 5 \times 10^{-5}, 10^{-6}, 5 \times 10^{-6}$
mask_ration	10^{-5} , 10^{-4} , 10^{-3} , 10^{-2} , 0.1 , 0.2 , 0.3 , 0.4 , 0.5 , 0.6

Table 22: The Hyper-parameter Search Spaces Table 22: The Hyper-parameter Search Spaces