Towards Deeper GCNs: Alleviating Over-smoothing via Iterative Training and Fine-tuning

Furong Peng[1], Jinzhen Gao[2], Xuan Lu[3](✉), Kang Liu[4], Yifan Huo[5], and Sheng Wang[6]

[1] Institute of Big Data Science and Industry, Shanxi University/Key Laboratory of Evolutionary Science Intelligence of Shanxi Province, Taiyuan, China
pengfr@sxu.edu.cn,
[2] Shanxi University, Taiyuan, China 202322404011@email.sxu.edu.cn,
[3] Shanxi University, Taiyuan, China xuanlu@sxu.edu.cn,
[4] Shanxi University, Taiyuan, China 202222407023@email.sxu.edu.cn,
[5] Shanxi University, Taiyuan, China 202422409007@email.sxu.edu.cn,
[6] Zhengzhou University of Aeronautics, Zhengzhou, China
wangsheng1910@zua.edu.cn

Abstract. Graph Convolutional Networks (GCNs) suffer from severe performance degradation in deep architectures due to over-smoothing. While existing studies primarily attribute the over-smoothing to repeated applications of graph Laplacian operators, our empirical analysis reveals a critical yet overlooked factor: trainable linear transformations in GCNs significantly exacerbate feature collapse, even at moderate depths (e.g., 8 layers). In contrast, Simplified Graph Convolution (SGC), which removes these transformations, maintains stable feature diversity up to 32 layers, highlighting linear transformations' dual role in facilitating expressive power and inducing over-smoothing. However, completely removing linear transformations weakens the model's expressive capacity.

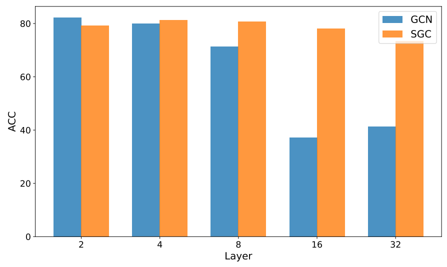
To address this trade-off, we propose **Layer-wise Gradual Training (LGT)**, a novel training strategy that progressively builds deep GCNs while preserving their expressiveness. LGT integrates three complementary components: (1) *layer-wise training* to stabilize optimization from shallow to deep layers, (2) *low-rank adaptation* to fine-tune shallow layers and accelerate training, and (3) *identity initialization* to ensure smooth integration of new layers and accelerate convergence. Extensive experiments on benchmark datasets demonstrate that LGT achieves state-of-the-art performance on vanilla GCN, significantly improving accuracy even in 32-layer settings. Moreover, as a training method, LGT can be seamlessly combined with existing methods such as PairNorm and ContraNorm, further enhancing their performance in deeper networks. LGT offers a general, architecture-agnostic training framework for scalable deep GCNs. The code is available at [https://github.com/jfklasdfj/LGT_GCN].

Keywords: GCNs · Over-smoothing · Fine-tune · LoRA

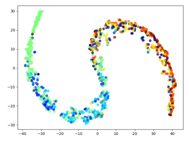
1 Introduction

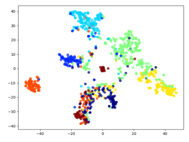
Graph Neural Networks (GNNs) [4,6,8] have become a powerful paradigm for learning from graph-structured data, achieving notable success across applications such as social network analysis [5,31], molecular property prediction [15], and traffic forecasting [13]. Among them, Graph Convolutional Networks (GCNs) [11,2] are widely adopted for their ability to aggregate neighborhood information through iterative message passing.

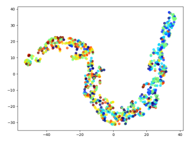
Despite their effectiveness, GCNs face severe performance degradation when scaled to deeper architectures due to the over-smoothing problem, where node representations become indistinguishable across layers [19,1]. Although previous studies primarily attribute over-smoothing to repeated applications of the graph Laplacian[14], our empirical analysis (Figure 1) reveals a critical yet under-explored cause: the linear transformations in GCN layers substantially accelerate feature collapse, even at moderate depths such as 8 layers. Interestingly, as shown in Figure 1, Simplified Graph Convolution (SGC) [28], which removes both linear transformations and nonlinearities, maintains stable performance and separable node representations even at 32 layers. This contrast highlights a fundamental trade-off: while linear transformations are essential for expressive feature learning, they also intensify over-smoothing in deep GCNs. Existing methods, however, largely overlook this dual role, leaving the challenge of balancing expressiveness and smoothness unresolved.



(a) Accuracy comparison of SGC and GCN at different depths.


(b) GCN (8 layers)


(c) SGC (8 layers)


(d) GCN (32 layers)

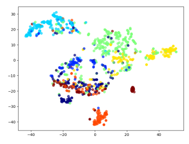

(e) SGC (32 layers)

Fig. 1: **Comparison of over-smoothing in GCN and SGC on Cora.** (a) Accuracy trend with increasing depth, where GCN suffers severe degradation beyond 8 layers, while SGC maintains stable performance. (b-e) Node embedding visualization (via t-SNE [16]) at different depths: GCN's features collapse as depth increases, whereas SGC preserves clear class separability even at 32 layers.

To address the limitations of deep GCNs, we introduce Layer-wise Gradual Training (LGT), a novel training paradigm designed to mitigate over-smoothing while preserving model expressiveness. Instead of training all layers simultaneously, LGT progressively expands the network depth, integrating three key strategies: incremental layer-wise training, low-rank adaptation (LoRA), and identity initialization.

Incremental Layer-wise Training. Traditional GCNs perform well in shallow settings but deteriorate rapidly as depth increases. To leverage this property, LGT adopts a staged training approach, gradually increasing depth while ensuring node embeddings remain discriminative at each step. Unlike prior layer-wise training strategies that require retraining the entire model upon adding new layers, LGT maintains previously learned representations, reducing computational redundancy and improving convergence stability.

Low-Rank Adaptation for Efficient Fine-tuning. While freezing earlier layers prevents over-smoothing, it also limits adaptation to new layers. To address this, LGT integrates Low-Rank Adaptation (LoRA) [10], enabling lightweight fine-tuning of pre-trained layers via a low-rank decomposition of weight updates. This approach significantly accelerates convergence, reducing the need for full retraining while maintaining representational flexibility.

Identity Initialization for Stable Expansion. Random initialization of newly added layers can disrupt learned representations, causing performance fluctuations. Inspired by SGC [28], which maintains performance using a fixed identity matrix as the linear transformation, LGT initializes new layers using identity matrices. This ensures smoother integration, reducing optimization instability and enhancing training efficiency.

By combining these strategies, LGT enables deep GCNs to achieve superior depth scalability, faster convergence, and improved classification accuracy while maintaining a lightweight and architecture-agnostic design. This framework not only provides a novel solution to over-smoothing but also establishes a robust foundation for training deep graph networks efficiently. Extensive experiments on semi-supervised node classification benchmarks demonstrate that LGT significantly alleviates over-smoothing and achieves state-of-the-art (SOTA) performance in deep vanilla GCNs (e.g., 32 layers), outperforming existing anti-over-smoothing methods. Furthermore, LGT is model-agnostic and can be seamlessly integrated with normalization-based techniques (e.g., PairNorm [30] and ContraNorm [9]) to further improve their performance.

In summary, our contributions are as follows:

– We identify trainable linear transformations as a critical cause of over-smoothing in moderate depth and propose a *training-based solution* fundamentally different from prior architectural modifications or regularization approaches.
– We introduce Layer-wise Gradual Training (LGT), a general and efficient strategy that combines incremental training, low-rank adaptation, and identity initialization, substantially improving the performance of deep vanilla

GCNs and achieving SOTA results. LGT is also compatible with existing anti-over-smoothing techniques.

The remainder of this paper is organized as follows: Section 2 reviews related work, Section 3 formalizes the problem to evaluate over-smoothing, Section 4 details the proposed LGT method, Section 5 presents experimental results, and Section 6 concludes the paper.

2 Related Work

Existing approaches to alleviating over-smoothing in GCNs can be broadly categorized into two groups: (1) modifying message-passing mechanisms and (2) constraining node representations. Below, we review models in each category.

2.1 Message-Passing-Based Approaches

Message-passing-based methods aim to adjust the information aggregation process to mitigate over-smoothing [21,27]. Graph Attention Networks (GAT) [26] assign adaptive weights to neighbors, promoting selective aggregation. However, GAT models often face gradient instability and complex optimization in deep architectures. DropEdge [22] randomly removes edges during training, reducing redundant aggregation. OrderedGNN [25] structurally organizes neurons based on hop distances to constrain message passing. ResGCN [12] and JK-Net [29] introduce cross-layer connections to retain information from shallow layers, while PSNR [32] combines residual links with adaptive normalization to capture multi-hop features. Although effective to some extent, these methods rely heavily on graph-dependent heuristics or sensitive hyperparameter tuning, limiting their generalization.

2.2 Representation-Based Approaches

Another line of work focuses on directly constraining node representations to prevent feature homogenization. EGNN [33] introduces Dirichlet energy regularization to preserve informative gradients across layers. ContraNorm [9] and PairNorm [30] maintain feature variance through normalization, while BatchNorm-GCN [3] applies batch normalization to stabilize feature distributions. Although these methods effectively mitigate over-smoothing, they may also suppress useful feature interactions when applied excessively, limiting model expressiveness.

2.3 Our Motivation and Distinction

In contrast to existing methods that focus on modifying message passing or enforcing explicit feature constraints, our proposed LGT addresses over-smoothing from a training strategy perspective. By incrementally training deeper layers and applying low-rank adaptation to earlier layers, LGT implicitly regularizes feature propagation without altering GCN architecture or graph structure. Unlike prior methods, LGT maintains representation diversity and depth scalability through an optimization-centered approach, and is inherently compatible with normalization-based techniques such as PairNorm and ContraNorm for further performance gains.

3 Problem Definition

Referring the methods of evaluating over-smoothing [11,21], we consider the standard semi-supervised node classification task on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Each node $v_i \in \mathcal{V}$ is associated with a feature vector $x_i \in \mathbb{R}^f$, and all node features form the feature matrix $\mathbf{X} \in \mathbb{R}^{n \times f}$. A subset of nodes $\mathcal{V}_L = \{v_1, \ldots, v_m\}$ with $m \ll n$ are labeled, while the remaining nodes $\mathcal{V}_U = \mathcal{V} \setminus \mathcal{V}_L$ are unlabeled.

The graph structure is encoded via the adjacency matrix $\mathbf{A} \in \{0,1\}^{n \times n}$, where $\mathbf{A}_{ij} = 1$ if an edge exists between nodes v_i and v_j. We adopt the normalized graph Laplacian:

$$\mathbf{L} \ = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I})\mathbf{D}^{-\frac{1}{2}}, \tag{1}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{D}$ is the diagonal degree matrix with $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$.

The objective of semi-supervised node classification is to train a GCN that minimizes the cross-entropy loss over labeled nodes:

$$\mathcal{L} = -\sum_{v_i \in Y_L} \sum_{c=1}^{C} y_{i,c} \log \hat{y}_{i,c}, \tag{2}$$

where C is the number of classes, $y_{i,c}$ is the one-hot label, and $\hat{y}_{i,c}$ is the predicted class probability for node v_i.

A key challenge arises when scaling GCNs to deep architectures due to the over-smoothing effect, where node representations $H^{(K)}$ progressively converge to a subspace with minimal discriminative information:

$$\lim_{K \to \infty} H^{(K)} \approx \mathbf{C}, \tag{3}$$

where $\mathbf{C}$ is a constant matrix independent of node identity. This phenomenon severely degrades classification accuracy on unlabeled nodes. Therefore, an effective GCN should maintain high classification accuracy even as the network depth K increases. Our goal is to design a training strategy that mitigates over-smoothing while enabling deeper GCNs to preserve discriminative node representations.

4 Layer-wise Gradual Training

While deep GCNs suffer from over-smoothing, our empirical analysis (Figure 1) reveals that trainable linear transformations accelerate feature collapse even at moderate depths. However, removing these transformations compromises model expressiveness, highlighting a key trade-off: GCNs require transformation layers for feature learning but must mitigate their role in over-smoothing to remain effective in deep architectures.

To address this, we introduce Layer-wise Gradual Training (LGT)—a progressive training strategy that incrementally deepens GCNs while preserving feature discriminability at each stage, as shown in Figure 2. Instead of training all layers simultaneously, LGT stabilizes optimization and prevents feature degradation through structured training. LGT consists of three key components: (1) *Incremental Layer-wise Training*, which prevents abrupt optimization difficulties by progressively adding layers, ensuring stable feature learning at each depth; (2) *Low-Rank Adaptation (LoRA)* [10], which lightly fine-tunes frozen layers to retain expressiveness without excessive parameter updates. (3) *Identity Matrix Initialization*, which ensures smooth integration of new layers, reducing parameter shifts and accelerating convergence. Empirical results (Section 5) confirm that LGT significantly improves classification accuracy in deep GCNs (e.g., 32 layers) while reducing computational overhead. The following sections detail each component.

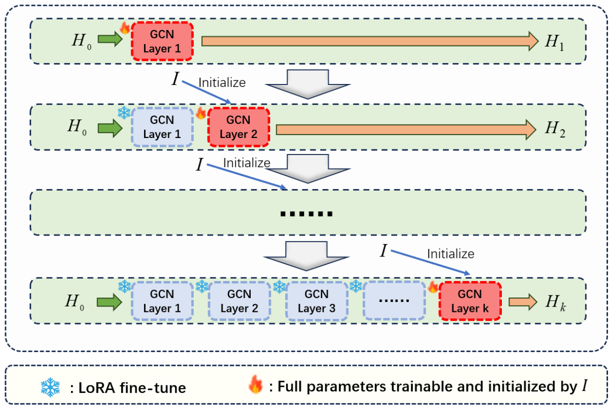


Fig. 2: **Illustration of Layer-wise Gradual Training (LGT).** At each stage, only the newly added layer is fully trained, while shallow layers are fine-tuned via LoRA. Once stabilized, a new layer is added and initialized with an identity matrix to ensure smooth integration.

4.1 Incremental Layer-wise Training

To address over-smoothing caused by rapid parameter growth, LGT decomposes GCN training into multiple stages. At each stage, only one new layer is added and trained, while all previously trained layers are fine-tuned (via LoRA). This staged optimization prevents feature collapse between layers and stabilizes feature propagation as depth increases.

First Layer Training. We start by training the first GCN layer to capture local neighborhood structures. Training proceeds until convergence based on validation performance, after which the layer is frozen.

Incremental Layer Addition. New GCN layers are added sequentially. At each step, only the new layer is fully trained, while shallow layers are slightly adapted using LoRA. Early stopping is applied to each stage to prevent overfitting. This procedure ensures that each layer progressively refines representations without causing over-smoothing or collapse.

4.2 Low-Rank Adaptation (LoRA)

While freezing earlier layers stabilizes training, it may restrict representation learning capacity. To balance stability and flexibility, we introduce LoRA [10] to adaptively fine-tune frozen layers using low-rank updates. Instead of fully updating the weight matrix $\mathbf{W}^{(k)} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$, we add a learnable low-rank component:

$$\tilde{\mathbf{W}}^{(k)} = \mathbf{W}_0^{(k)} + \mathbf{A}^{(k)}\mathbf{B}^{(k)}, \tag{4}$$

where $\mathbf{W}_0^{(k)}$ is the frozen pretrained weight, and $\mathbf{A}^{(k)} \in \mathbb{R}^{d_{\text{in}} \times r}, \mathbf{B}^{(k)} \in \mathbb{R}^{r \times d_{\text{out}}}$ are low-rank matrices with $r \ll \min(d_{\text{in}}, d_{\text{out}})$. The forward propagation becomes:

$$\mathbf{H}^{(k+1)} = \sigma(\mathbf{L}\mathbf{H}^{(k)}(\mathbf{W}_0^{(k)} + \mathbf{A}^{(k)}\mathbf{B}^{(k)})), \tag{5}$$

and only $\mathbf{A}^{(k)}$, $\mathbf{B}^{(k)}$ are updated during training. This significantly reduces memory and computational cost while preserving adaptability in deep models.

4.3 Identity Matrix Initialization

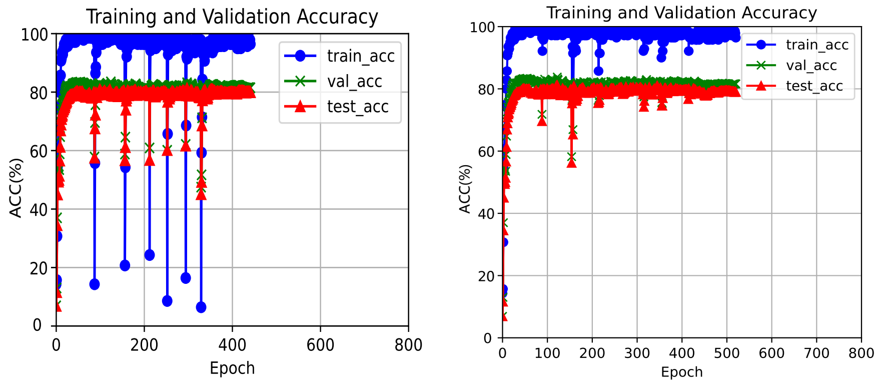
Proper initialization of new layers is crucial for stable incremental training. Random initialization may lead to unstable gradients and disrupt previously learned representations, as shown in Figure 3. We initialize each newly added layer's weights as an identity matrix, ensuring initial outputs align with the prior layer's features:

$$\mathbf{H}^{(k+1)} = \sigma(\mathbf{L}\mathbf{H}^{(k)}\mathbf{W}^{(k)}) \approx \sigma(\mathbf{L}\mathbf{H}^{(k)}), \tag{6}$$

thus providing a smooth transition for training deeper layers. As demonstrated in Figure 3, identity initialization significantly improves stability and accelerates convergence compared to random initialization.

4.4 Summary

In summary, LGT integrates incremental layer-wise training, low-rank adaptation, and identity initialization into a unified framework that enables deep GCNs to avoid over-smoothing while retaining strong representational power. Unlike



(a) Random Initialization. (b) Identity Initialization.

Fig. 3: **Effect of initialization methods.** Identity initialization stabilizes the training of newly added layers compared to random initialization.

prior methods that modify GCN architectures or impose explicit constraints on node representations, LGT addresses over-smoothing through an optimization-centric training paradigm, offering a scalable and compatible solution for training deep GCNs. The effectiveness of LGT is validated through extensive experiments in the next section.

5 Experiments

We conduct comprehensive experiments to evaluate the effectiveness of our proposed LGT in alleviating over-smoothing for deep GCNs on *semi-supervised node classification* tasks [11]. All models are evaluated using classification accuracy (ACC) as the primary metric. Our evaluation covers 4 benchmark datasets and 6 state-of-the-art baselines.

5.1 Experimental Setup

Datasets. To evaluate the generalizability of LGT across diverse graph domains, we conduct experiments on 4 widely used citation and co-authorship datasets: Cora [17,23], Citeseer [7,23], Pubmed [18,23], and AmazonPhoto [24]. The statistics of these datasets are summarized in Table 1. For all datasets, we follow the widely adopted semi-supervised learning setting [11], randomly selecting 20 labeled nodes per class for training, and reserving 1,000 nodes each for validation and test sets. The remaining nodes are treated as unlabeled. To ensure robustness, we repeat each experiment with 5 random splits and report the mean performance.

Table 1: The summary of dataset

Datasets	Cls.	Nodes	Edges	Feat.	Train/Valid/Test
Cora	7	2708	5429	1433	140/1000/1000
Citeseer	6	3327	4732	2703	120/1000/1000
Pubmed	3	19717	44338	500	60/1000/1000
AmazonPhoto	8	7650	119043	745	160/1000/1000

Baselines. To thoroughly validate the effectiveness of LGT, we compare several representative and state-of-the-art GCN-based models that address over-smoothing. The baselines include:

- **GCN (ICLR'17)** [11]: A foundational graph convolutional network model that performs neighborhood aggregation with learned linear transformations.
- **SGC (ICML'19)** [28]: A simplified GCN variant that removes nonlinear activations and collapses multiple layers into a single linear transformation to study the role of propagation.
- **PairNorm (ICLR'20)** [30]: A normalization technique that preserves feature variance across layers to prevent feature collapse.
- **ContraNorm (ICLR'23)** [9]: A method introducing implicit feature decorrelation to maintain node diversity and alleviate over-smoothing.
- **IresGCN (ICML'24)** [20]: An inverse residual GCN framework that adjusts message passing directions to preserve personalized node information.
- **PSNR (NeurIPS'24)** [32]: A posterior sampling and adaptive residual method that adaptively retains hierarchical information during propagation.

These baselines cover a wide range of anti-over-smoothing techniques, including normalization, residual connections, and structural adjustments. Thus, they provide a strong basis for assessing the performance and compatibility of LGT.

5.2 Effectiveness on Alleviating Over-smoothing

To evaluate LGT's effectiveness in mitigating over-smoothing, we measure classification accuracy across different depths (4, 8, 16, and 32 layers) on four benchmark datasets: Cora, Citeseer, Pubmed, and AmazonPhoto. LGT is applied to GCN, ContraNorm, and PairNorm to assess its generalizability. The complete results are shown in Table 2, where the performances of best and second-best are highlighted, and the improvements brought about by LGT are marked with ↑.

Performance Gains in Deep GCNs. Applying LGT to vanilla GCN leads to a significant accuracy boost in deep layers. On Cora, for example, GCN+LGT achieves 82.0% accuracy at 32 layers, doubling the performance of standard GCN (41.3%). Similar trends are observed across datasets, confirming that *LGT effectively prevents over-smoothing and supports stable deep GCN training.* Additionally, GCN+LGT consistently outperforms SGC at 32 layers across all datasets, with 10–15% gains on Cora and AmazonPhoto. Interestingly, GCN+LGT also

Table 2: Classification accuracy (ACC) across different datasets (in percent)

Dataset	Model	Layers			
		4	8	16	32
Cora	SGC	$79.88_{\pm 1.00}$	$79.08_{\pm 1.42}$	$75.90_{\pm 1.91}$	$69.62_{\pm 2.57}$
	IresGCN	$79.28_{\pm 1.85}$	$78.78_{\pm 0.63}$	$79.04_{\pm 0.54}$	$78.94_{\pm 0.92}$
	PSNR	$79.90_{\pm 1.00}$	$79.00_{\pm 1.39}$	$79.54_{\pm 0.73}$	$79.48_{\pm 0.83}$
	GCN	$78.58_{\pm 1.29}$	$71.30_{\pm 1.54}$	$37.20_{\pm 2.22}$	$40.46_{\pm 0.78}$
	GCN+LGT	$80.94_{\pm 1.03}^{\uparrow}$	$80.76_{\pm 1.28}^{\uparrow}$	$80.58_{\pm 0.49}^{\uparrow}$	$81.06_{\pm 0.97}^{\uparrow}$
	ContraNorm	$79.74_{\pm 0.77}$	$78.96_{\pm 1.06}$	$79.42_{\pm 1.88}$	$80.10_{\pm 1.01}$
	ContraNorm+LGT	$80.36_{\pm 1.20}^{\uparrow}$	$80.34_{\pm 0.94}^{\uparrow}$	$80.66_{\pm 1.63}^{\uparrow}$	$80.62_{\pm 0.84}^{\uparrow}$
	PairNorm	$78.40_{\pm 1.88}$	$78.10_{\pm 1.05}$	$77.76_{\pm 1.70}$	$77.46_{\pm 1.22}$
	PairNorm+LGT	$79.50_{\pm 0.56}^{\uparrow}$	$77.50_{\pm 1.49}$	$78.88_{\pm 0.24}^{\uparrow}$	$79.68_{\pm 1.33}^{\uparrow}$
Citeseer	SGC	$69.66_{\pm 0.61}$	$69.56_{\pm 1.15}$	$69.06_{\pm 1.63}$	$67.90_{\pm 1.37}$
	IresGCN	$65.20_{\pm 2.21}$	$66.30_{\pm 1.76}$	$66.38_{\pm 1.46}$	$67.26_{\pm 2.13}$
	PSNR	$66.60_{\pm 2.23}$	$65.00_{\pm 2.45}$	$65.64_{\pm 1.47}$	$64.38_{\pm 2.80}$
	GCN	$65.54_{\pm 1.80}$	$50.70_{\pm 3.64}$	$25.58_{\pm 1.13}$	$25.70_{\pm 3.65}$
	GCN+LGT	$69.58_{\pm 1.23}^{\uparrow}$	$69.46_{\pm 1.35}^{\uparrow}$	$69.74_{\pm 1.07}^{\uparrow}$	$69.58_{\pm 0.53}^{\uparrow}$
	ContraNorm	$66.52_{\pm 1.41}$	$66.42_{\pm 1.07}$	$66.34_{\pm 1.92}$	$66.28_{\pm 1.57}$
	ContraNorm+LGT	$68.18_{\pm 2.12}^{\uparrow}$	$68.48_{\pm 1.35}^{\uparrow}$	$68.50_{\pm 1.28}^{\uparrow}$	$68.52_{\pm 1.16}^{\uparrow}$
	PairNorm	$67.66_{\pm 2.40}$	$67.64_{\pm 2.12}$	$66.66_{\pm 2.84}$	$65.40_{\pm 2.00}$
	PairNorm+LGT	$69.06_{\pm 1.90}^{\uparrow}$	$68.48_{\pm 1.44}^{\uparrow}$	$65.12_{\pm 2.12}$	$65.80_{\pm 2.50}^{\uparrow}$
Pubmed	SGC	$74.30_{\pm 1.22}$	$73.74_{\pm 2.05}$	$72.44_{\pm 0.97}$	$70.08_{\pm 0.73}$
	IresGCN	$77.24_{\pm 1.13}$	$77.70_{\pm 2.03}$	$76.92_{\pm 0.43}$	$77.48_{\pm 1.79}$
	PSNR	$76.94_{\pm 1.65}$	$77.24_{\pm 1.33}$	$77.24_{\pm 0.71}$	$77.42_{\pm 1.07}$
	GCN	$77.04_{\pm 2.12}$	$70.48_{\pm 3.43}$	$44.24_{\pm 3.49}$	$47.70_{\pm 4.11}$
	GCN+LGT	$77.94_{\pm 2.12}^{\uparrow}$	$77.34_{\pm 1.48}^{\uparrow}$	$77.82_{\pm 1.43}^{\uparrow}$	$77.70_{\pm 1.09}^{\uparrow}$
	ContraNorm	$77.94_{\pm 2.04}$	OOM	OOM	OOM
	ContraNorm+LGT	$77.88_{\pm 1.22}$	$77.40_{\pm 1.81}$	OOM	OOM
	PairNorm	$77.02_{\pm 1.37}$	$77.92_{\pm 2.33}$	$77.90_{\pm 1.41}$	$77.40_{\pm 0.65}$
	PairNorm+LGT	$77.08_{\pm 0.71}^{\uparrow}$	$76.96_{\pm 0.94}$	$77.14_{\pm 0.52}$	$77.32_{\pm 0.98}$
AmazonPhoto	SGC	$89.18_{\pm 0.54}$	$87.50_{\pm 1.42}$	$84.84_{\pm 2.07}$	$75.80_{\pm 4.41}$
	IresGCN	$90.26_{\pm 1.40}$	$90.48_{\pm 2.05}$	$90.76_{\pm 1.50}$	$91.68_{\pm 1.18}$
	PSNR	$90.98_{\pm 0.61}$	$90.72_{\pm 1.02}$	$91.14_{\pm 0.31}$	$91.26_{\pm 1.25}$
	GCN	$90.22_{\pm 0.61}$	$88.40_{\pm 0.96}$	$64.24_{\pm 3.18}$	$69.94_{\pm 5.36}$
	GCN+LGT	$91.10_{\pm 0.76}^{\uparrow}$	$91.02_{\pm 0.52}^{\uparrow}$	$91.26_{\pm 0.72}^{\uparrow}$	$91.34_{\pm 0.25}^{\uparrow}$
	ContraNorm	$91.14_{\pm 1.25}$	$91.48_{\pm 0.88}$	$91.64_{\pm 0.45}$	$91.56_{\pm 0.75}$
	ContraNorm+LGT	$91.74_{\pm 0.89}^{\uparrow}$	$91.52_{\pm 0.33}^{\uparrow}$	$91.56_{\pm 0.70}$	$91.80_{\pm 0.91}^{\uparrow}$
	PairNorm	$91.14_{\pm 0.59}$	$90.54_{\pm 0.28}$	$90.76_{\pm 0.53}$	$91.00_{\pm 1.00}$
	PairNorm+LGT	$91.40_{\pm 0.71}^{\uparrow}$	$90.94_{\pm 0.81}^{\uparrow}$	$89.90_{\pm 1.46}$	$91.38_{\pm 0.40}^{\uparrow}$

surpasses SGC at moderate depths (4 layers) on Cora, Pubmed, and AmazonPhoto, demonstrating that *proper training can balance expressiveness and smoothness*, preserving GCN's advantages.

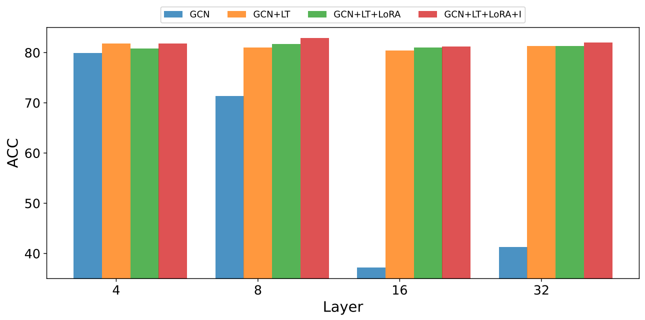


Fig. 4: **Results of ablation study (Cora).**

Compatibility with Existing Anti-over-smoothing Methods. When applied to ContraNorm, LGT enhances its performance on Cora, Citeseer, and AmazonPhoto. On Pubmed, where standard ContraNorm suffers from out-of-memory (OOM) issues at 8 layers, LGT improves efficiency, producing competitive results at 4 layers and valid outputs at 8 layers. For PairNorm, LGT improves performance in 10 out of 16 settings, further confirming its broad applicability to existing anti-over-smoothing techniques.

In summary, LGT (1) effectively mitigates over-smoothing for deep GCNs, (2) improves the deep model performance, and (3) complements existing anti-over-smoothing methods, offering a general and scalable training strategy.

5.3 Ablation Study

To systematically assess the individual contributions of each component in LGT, we perform ablation studies on four GCN-based variants: (1) standard GCN as baseline, (2) GCN with Layer-wise Training (GCN+LT), (3) GCN with LT and Low-Rank Adaptation (GCN+LT+LoRA), and (4) full LGT incorporating LT, LoRA, and Identity Initialization (GCN+LT+LoRA+I). Figure 4 reports the comparison results on the Cora dataset.

As shown in Figure 4, the introduction of LT significantly improves GCN performance in all depths (4 to 32 layers), with especially notable gains in 16 layers. Furthermore, adding LoRA and identity initialization brings additional improvements for deeper models (8–32 layers), demonstrating their roles in enhancing representation capacity and stabilizing optimization. These results confirm the positive and complementary contributions of all three components in alleviating over-smoothing and improving deep GCN performance.

5.4 Training Efficiency Analysis

We evaluate the training efficiency of LGT on Cora and AmazonPhoto datasets, as shown in Figure 5. Applying LGT to both GCN and PairNorm, we observe that LGT not only improves GCN's classification accuracy (e.g., from 70% to

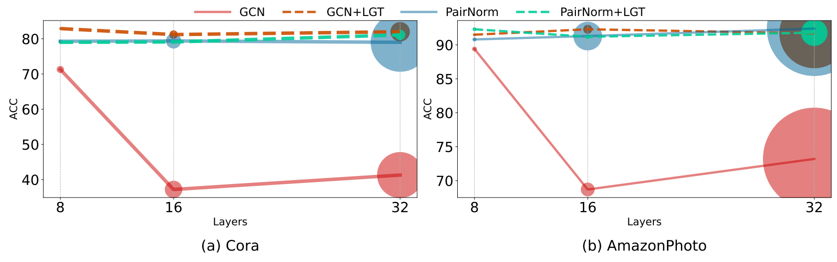


Fig. 5: **Training efficiency and accuracy comparison.** Line plots show classification accuracy across different network depths, and circle sizes indicate relative training time. LGT significantly improves GCN's accuracy while reducing training time. For PairNorm, LGT enhances training efficiency without sacrificing accuracy, highlighting its complementary effect.

Table 3: Training Time Comparison

Models	Cora			AmazonPhoto		
	Layer 8	Layer 16	Layer 32	Layer 8	Layer 16	Layer 32
GCN	115s	293s	784s	72s	228s	1760s
GCN+LGT	28s	135s	326s	41s	1452s	1194s
PairNorm	46s	252s	1006s	61s	485s	1627s
PairNorm+LGT	42s	62s	170s	55s	69s	447s

over 80% on Cora) but also significantly reduces training time by avoiding full-model optimization through progressive layer-wise updates, as shown in Table 3. Although PairNorm already addresses over-smoothing, integrating LGT further reduces its training time without compromising accuracy. These results indicate that LGT complements existing normalization methods by enhancing training efficiency while maintaining or improving model performance.

5.5 Effect of Rank in LoRA

We analyze the impact of the rank parameter in LoRA on model performance under varying network depths on AmazonPhoto. As shown in Figure 6, rank influences classification accuracy, particularly in deeper networks.

Overall, **rank=10** achieves the most stable and consistently high performance across 4, 8, and 16 layers, striking a balance between expressiveness and computational efficiency. Conversely, while **rank=32** exhibits comparatively over rank=10 in shadow networks (8 layers), it generally underperforms compared to 4 and 16 layers, likely due to over-parameterization and reduced generalization.

These results highlight the critical role of rank selection: overly small ranks limit model capacity, while excessively large ranks introduce redundancy and

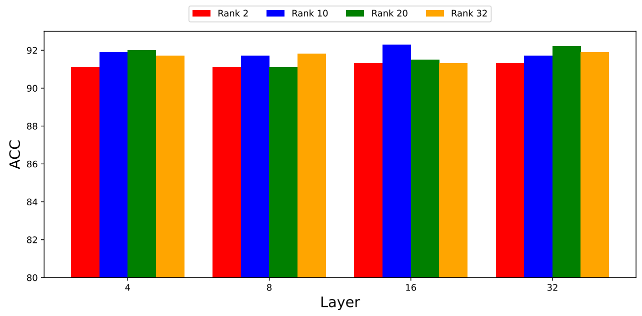


Fig. 6: **Results of rank changing over layers (AmazonPhoto).**

unnecessary computational overhead. Thus, **rank=10** is recommended for practical use, offering robust performance across depths with controlled complexity.

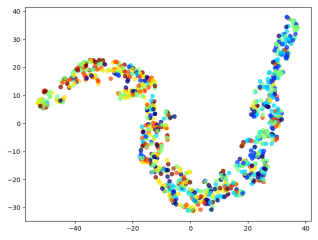
5.6 Node Embedding Visualization

We visualize the 32-layer node embeddings on the Cora dataset using t-SNE to assess feature separability under deep architectures. As shown in Figure 7, vanilla GCN suffers from severe over-smoothing, with highly mixed node features and indistinct class boundaries. In contrast, GCN+LGT produces well-separated and compact clusters, demonstrating improved feature discrimination and effective mitigation of over-smoothing. Moreover, combining LGT with ContraNorm (ContraNorm+LGT) further enhances class separability compared to ContraNorm alone, confirming LGT's compatibility and complementary effect. These results highlight that LGT enables deep GCNs to preserve expressive and discriminative node representations.

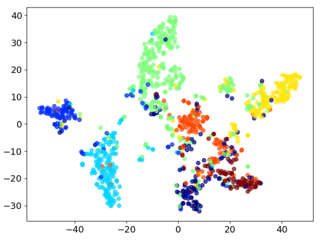
6 Conclusion

This paper revisits the over-smoothing problem in deep GCNs and identifies trainable linear transformations, rather than just the graph Laplacian, as a key factor exacerbating feature collapse. To address this, we propose Layer-wise Gradual Training (LGT), a novel training strategy that progressively deepens GCNs while preserving feature expressiveness. LGT integrates incremental layer-wise training for stabilized optimization, low-rank adaptation for efficient fine-tuning, and identity initialization for smooth layer expansion. Extensive experiments demonstrate that LGT significantly improves deep GCN performance, mitigating over-smoothing while enhancing training efficiency. Moreover, LGT is highly compatible with existing anti-over-smoothing techniques, such as PairNorm and ContraNorm, extending their scalability to deeper architectures. The efficiency analysis highlights faster convergence and improved stability.

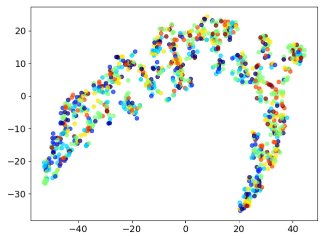
Despite these advantages, two key areas warrant further exploration. (1) *Theoretical insights into LGT's optimization benefits.* The over-smoothing can be



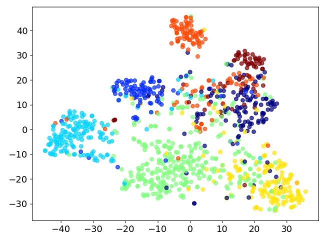
(a) GCN



(b) GCN+LGT



(c) ContraNorm



(d) ContraNorm+LGT

Fig. 7: The node embedding visualization of GCN, ContraNorm, and their respective variants GCN+LGT and ContraNorm+LGT under the 32th layer.

viewed as a degenerate optimization issue. LGT preserves well-initialized shallow layers and prevents unstable gradient interference from randomly initialized deep layers. A deeper theoretical analysis—possibly through saddle-point theory or local optimization landscapes—could explain how LGT helps GCNs converge toward desirable optimization regions, thereby preventing over-smoothing. (2) *Interaction with existing anti-over-smoothing methods*. While LGT enhances several approaches, its impact varies across datasets (e.g., PairNorm on Pubmed). Future work will explore the relationship between training-based and structure-based regularization to refine deep GCN optimization.

Overall, LGT provides a general, training-centric solution for deep GCNs, shifting the focus from architectural design to optimization strategies and paving the way for more effective and scalable graph learning models.

7 Acknowledgments

This work was supported by the National Natural Science Foundation of China (Nos. 62276162, 62272286), the Fundamental Research Program of Shanxi Province (No. 202203021222016), the Central guidance for Local scientific and technological development funds (No. YDZJSX20231B001), and the Key Laboratory of Evolutionary Science Intelligence of Shanxi Province.

References

1. Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., Sun, X.: Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 3438–3445 (2020)
2. Chen, M., Wei, Z., Huang, Z., Ding, B., Li, Y.: Simple and deep graph convolutional networks. In: III, H.D., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 1725–1735. PMLR (13–18 Jul 2020)
3. Chen, Y., Tang, X., Qi, X., Li, C.G., Xiao, R.: Learning graph normalization for graph neural networks. Neurocomputing **493**, 613–625 (2022). `https://doi.org/https://doi.org/https://doi.org/10.1016/j.neucom.2022.01.003`
4. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems **29** (2016)
5. Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., Yin, D.: Graph neural net works for social recommendation. In: The World Wide Web Conference. p. 417–426. WWW '19, Association for Computing Machinery, New York, NY, USA (2019)
6. Feng, W., Zhang, J., Dong, Y., Han, Y., Luan, H., Xu, Q., Yang, Q., Kharlamov, E., Tang, J.: Graph random neural networks for semi-supervised learning on graphs. Advances in neural information processing systems **33**, 22092–22103 (2020)
7. Giles, C.L., Bollacker, K.D., Lawrence, S.: Citeseer: an automatic citation indexing system. In: Proceedings of the Third ACM Conference on Digital Libraries. p. 89–98. DL '98, Association for Computing Machinery, New York, NY, USA (1998). `https://doi.org/https://doi.org/10.1145/276675.276685`
8. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Proceedings of the 34th International Conference on Machine Learning - Volume 70. p. 1263–1272. ICML'17, JMLR.org (2017)
9. Guo, X., Wang, Y., Du, T., Wang, Y.: Contranorm: A contrastive learning perspective on oversmoothing and beyond. In: The Eleventh International Conference on Learning Representations (2023)
10. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: LoRA: Low-rank adaptation of large language models. In: International onference on Learning Representations (2022)
11. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (2017)
12. Li, G., Muller, M., Thabet, A., Ghanem, B.: Deepgcns: Can gcns go as deep as cnns? In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 9267–9276 (2019)
13. Li, H., Zhao, Y., Mao, Z., Qin, Y., Xiao, Z., Feng, J., Gu, Y., Ju, W., Luo, X., Zhang, M.: A survey on graph neural networks in intelligent transportation systems. arXiv preprint arXiv:2401.00713 (2024)
14. Li, Q., Han, Z., Wu, X.M.: Deeper insights into graph convolutional networks for semi-supervised learning. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence. AAAI'18/IAAI'18/EAAI'18, AAAI Press (2018)
15. Lin, X., Quan, Z., Wang, Z.J., Ma, T., Zeng, X.: Kgnn: Knowledge graph neural network for drug-drug interaction prediction. In: Bessiere, C. (ed.) Proceedings of

the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20. pp. 2739–2745. International Joint Conferences on Artificial Intelligence Organization (7 2020). https://doi.org/https://doi.org/10.24963/ijcai.2020/380, main track

16. van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research **9**(86), 2579–2605 (2008)
17. McCallum, A.K., Nigam, K., Rennie, J., Seymore, K.: Automating the construction of internet portals with machine learning. Information Retrieval **3**, 127–163 (2000)
18. Namata, G., London, B., Getoor, L., Huang, B., Edu, U.: Query-driven active surveying for collective classification. In: 10th international workshop on mining and learning with graphs. vol. 8, p. 1 (2012)
19. Oono, K., Suzuki, T.: Graph neural networks exponentially lose expressive power for node classification. In: International Conference on Learning Representations (2020)
20. Park, M., Heo, J., Kim, D.: Mitigating oversmoothing through reverse process of gnns for heterophilic graphs. In: Proceedings of the 41st International Conference on Machine Learning. ICML'24, JMLR.org (2024)
21. Peng, F., Liu, K., Lu, X., Qian, Y., Yan, H., Ma, C.: Tsc: A simple two-sided constraint against over-smoothing. In: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 2376–2387 (2024)
22. Rong, Y., Huang, W., Xu, T., Huang, J.: Dropedge: Towards deep graph convolutional networks on node classification. arXiv preprint arXiv:1907.10903 (2019)
23. Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T.: Collective classification in network data. AI Mag. **29**(3), 93–106 (Sep 2008). https://doi.org/https://doi.org/10.1609/aimag.v29i3.2157
24. Shchur, O., Mumme, M., Bojchevski, A., Günnemann, S.: Pitfalls of graph neural network evaluation. arXiv preprint arXiv:1811.05868 (2018)
25. Song, Y., Zhou, C., Wang, X., Lin, Z.: Ordered GNN: Ordering message passing to deal with heterophily and over-smoothing. In: The Eleventh International Conference on Learning Representations (2023)
26. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (2018)
27. Wang, K., Li, G., Wang, S., Zhang, G., Wang, K., You, Y., Peng, X., Liang, Y., Wang, Y.: The snowflake hypothesis: Training deep gnn with one node one receptive field. arXiv preprint arXiv:2308.10051 (2023)
28. Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., Weinberger, K.: Simplifying graph convolutional networks. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 6861–6871. PMLR (09–15 Jun 2019)
29. Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.i., Jegelka, S.: Representation learning on graphs with jumping knowledge networks. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 5453–5462. PMLR (10–15 Jul 2018)
30. Zhao, L., Akoglu, L.: Pairnorm: Tackling oversmoothing in gnns. In: International Conference on Learning Representations (2020)
31. Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., Sun, M.: Graph neural networks: A review of methods and applications. AI Open 1, 57–81 (2020)

32. Zhou, J., Du, Y., Zhang, R., Xia, J., Yu, Z., Zang, Z., Jin, D., Yang, C., Zhang, R., Li, S.Z.: Deep graph neural networks via posteriori-sampling-based node-adaptative residual module. In: Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., Zhang, C. (eds.) Advances in Neural Information Processing Systems. vol. 37, pp. 68211–68238. Curran Associates, Inc. (2024)
33. Zhou, K., Huang, X., Zha, D., Chen, R., Li, L., Choi, S.H., Hu, X.: Dirichlet energy constrained learning for deep graph neural networks. In: Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems (2021)