Graph Neural Networks (with Proper Weights) Can Escape Oversmoothing

Zhijian Zhuo zhijianzhuo@stu.pku.edu.cn
School of Mathematical Sciences, Peking University

Yifei Wang yifei_w@mit.edu
CSAIL, MIT

Jinwen Ma✉ jwma@math.pku.edu.cn
School of Mathematical Sciences, Peking University

Yisen Wang✉ yisen.wang@pku.edu.cn
National Key Lab of General Artificial Intelligence,
School of Intelligence Science and Technology, Peking University

Editors: Vu Nguyen and Hsuan-Tien Lin

Abstract

Graph Neural Networks (GNNs) are known to suffer from degraded performance with more layers. Most prior works explained it from graph propagation, arguing that it inevitably leads to indistinguishable node features under more depth, known as *oversmoothing*. However, we notice that these analyses largely ignore the role of GNN weights either directly or by unrealistically strong assumptions. In this paper, we rediscover the role of GNN weights on oversmoothing with a systematic study. Notably, contrary to previous findings, we show that when learned freely, there always exist ideal weights such that vanilla GNNs completely avoid oversmoothing, even after infinite propagation steps. It indicates that oversmoothing is a problem of learning disabilities instead of the doom of GNNs themselves. To facilitate the learning of proper weights, we propose Weight Reparameterization (**WeightRep**) as a way to adaptively maintain the ideal weights in vanilla GNNs along the learning process. We theoretically show that for linear GNNs, WeightRep can always mitigate oversmoothing (full collapse) as well as dimensional collapse. Extensive experiments on nine benchmark datasets demonstrate its effectiveness and efficiency in practice.

Keywords: Graph neural networks, Oversmoothing, Semi-supervised learning.

1. Introduction

Graph Neural Networks (GNNs) have revolutionized a multitude of fields with their unique ability to learn from graph-structured data, finding widespread application in domains such as node classification (Kipf and Welling, 2017; Hamilton et al., 2017), drug discovery (Xiong et al., 2019), computational chemistry (Gilmer et al., 2017), recommendation systems (Wang et al., 2018), and traffic prediction (Zhao et al., 2019). Generally, most existing GNNs are based on the message-passing paradigm (Kipf and Welling, 2017; Veličković et al., 2018), which includes neighborhood aggregation and feature transformation, to learn representations of given graphs.

© 2024 Z. Zhuo, Y. Wang, J. Ma & Y. Wang.

Despite their rapid development, GNNs face a common issue: the oversmoothing phenomenon (Li et al., 2018; Guo et al., 2023a; Rusch et al., 2023), where node representations become excessively similar across the graph as the depth of networks increases, leading to significant performance deterioration. Hence, the best performance is usually attached when networks are shallow. This limitation constrains GNNs' representation capability, making it crucial to investigate and counter oversmoothing for real-world applications. Empirically, oversmoothing has been observed on many GNNs and datasets (Li et al., 2018; Chen et al., 2020a; Rusch et al., 2023). Theoretically, oversmoothing has been shown to inevitably occur both asymptotically (Li et al., 2018; Oono and Suzuki, 2020; Cai and Wang, 2020; Guo et al., 2023b; Roth and Liebig, 2023) and non-asymptotically (Keriven, 2022).

In this paper, we challenge these theoretical results by proving that common GNNs, such as GCN, can indeed escape oversmoothing, even after infinite propagation steps. Our key insight is to rediscover the potential of weight matrices in GNNs, which were either neglected (Li et al., 2018) or assumed to be small enough (Oono and Suzuki, 2020) in previous studies. Notably, we theoretically prove *the existence of specific weights that ensure representations of GNNs are well distinguished at each layer*. Therefore, our analysis suggests that oversmoothing is not an inherent structural flaw but rather a result of the learning process failing to identify non-smoothing weights. That is, existing optimizers are unable to find the non-smoothing weights that *do* exist. Guided by the principle, we propose **Weight Reparameterization (WeightRep)**, a novel approach that dynamically constructs input-dependent weights that maintain a desirable representation spectrum in vanilla GNNs. This method ensures that weights of GNNs remain ideal (in the sense of escaping from oversmoothing) throughout the training process. Our extensive experiments across various models and benchmark datasets show that GCN employing WeightRep outperforms vanilla GCN as well as with normalization on various depths. Our key contributions are summarized as follows:

- We systematically analyze the role of GNN weights on oversmoothing in three cases: without weights, random weights, and freely learnable weights. Contrary to previous findings, we show that when weights are learnable, there always exist proper weights that can prevent oversmoothing at each GNN layer.
- Inspired by the theoretical analysis, we propose WeightRep which reparameterizes weights to be adaptive to input features. In this way, WeightRep ensures that weights of GNNs stay ideal throughout the training process, as supported by theoretical guarantees.
- We validate the effectiveness of WeightRep through comprehensive experiments on real-world graphs. Results show that our method can effectively mitigate oversmoothing across different types of graph datasets.

2. Related Work

Graph Neural Networks. There are two main families of GNNs, spectral-based methods and spatial-based methods. Spectral-based approaches use graph convolutions which are based on graph spectral theory (Bruna et al., 2013; Klicpera et al., 2019; He et al., 2021).

And spatial-based approaches define graph convolutions by aggregating and transforming feature information from neighbors (Hamilton et al., 2017). It includes a lot of famous variants like GCN (Kipf and Welling, 2017) and GAT (Veličković et al., 2018).

Oversmoothing in GNNs. Oversmoothing is a key issue with GNNs, where increasing the depth of GNNs results in the node features converging to similar values Li et al. (2018). GNNs usually achieve optimal performance when networks are shallow due to oversmoothing. Some methods are proposed to mitigate oversmoothing in GNNs. Inspired by the dropout technique (Hinton et al., 2012), researchers also add a random dropping module to each layer. For instance, DropEdge (Rong et al., 2019) and DropNode (Feng et al., 2020) randomly drop edges and nodes of the underlying graph during training, respectively. And DropMessage (Fang et al., 2023) unified random dropping of edges and nodes for GNNs. A proven way is adding residual connections that change the structure of GNNs, including JKNet (Xu et al., 2018), DeepGCNs (Li et al., 2019) and GCNII (Chen et al., 2020b). Another line of research is utilizing normalization techniques such as PairNorm (Zhao and Akoglu, 2020), NodeNorm (Zhou et al., 2021), ContraNorm (Guo et al., 2023a).

Theoretical Analysis of Oversmoothing. Existing theoretical results are mainly to prove the existence of oversmoothing. The pioneering work of Li et al. (2018) showed that the output of GCNs would converge to the eigenspace associated with the maximum eigenvalue under the linear and without the feature transformation setting. For nonlinear GNNs, Oono and Suzuki (2020) proved that the infinite layer GCN only outputs information of the graph Laplacian with bounded feature transformation. And Cai and Wang (2020) showed that the Dirichlet energy of embeddings will converge to zero. Keriven (2022) also characterized oversmoothing from a non-asymptotic view. We note that Roth and Liebig (2023) claimed that oversmoothing of GNNs occurs independently of feature transformations. Recently, Wu et al. (2023) proved that GAT loses expressive power exponentially under mild assumptions. Different from the previous theory, we prove the existence of proper weights such that GNNs can avoid oversmoothing and such weights can be found by WeightRep.

3. Preliminaries

3.1. Notations and Assumption

We first describe key notations used in this paper. We denote a graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$ consisting of a set of N nodes $\mathcal{V} = \{v_1, v_2, \dots, v_N\}$, a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and the node feature matrix $X \in \mathbb{R}^{N\times d}$. The neighborhood set of node v_i is denoted by $\mathcal{N}_i$. Also, we use the adjacency matrix $A \in \mathbb{R}^{N\times N}$ to describe the graph structure. An $N-$dimension all-ones vector is denoted by $\mathbf{1}$. Let $D = \text{diag}(A\mathbf{1})$ denote the degree matrix of $\mathcal{G}$. The graph Laplacian is defined as $L = D - A$ and two versions of normalized Laplacian are then given by $L_{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$ and $L_{rm} = D^{-1}L$, respectively.

Let $\|\cdot\|_2$, $\|\cdot\|_F$ be the 2-norm and Frobenious norm of matrix, respectively. We use the shorthand $[n] = \{1, 2, \dots, n\}$. $\text{Tr}(\cdot)$ denotes trace of a matrix. The spectral radius of a square matrix A denoted by $\rho(A)$ is defined as $\rho(A) = \max\{|\lambda_1|, |\lambda_2|, \dots, |\lambda_N|\}$, where $\lambda_1, \lambda_2, \dots, \lambda_N$ are eigenvalues of A. For two real matrices with the same dimension B, C, we define $B \geq C$ $(B > C)$ if $B_{ij} \geq C_{ij}$ $(B_{ij} > C_{ij})$ for each i, j.

We make the following assumption on the graph which is a common assumption for the theoretical analysis of graphs Cai and Wang (2020); Wu et al. (2023).

Assumption 1 *The graph $\mathcal{G}$ is connected and each node on the graph has a self-loop.*

For disconnected graphs, we can apply our results to each connected component of graphs. And in practice, we usually add a self-loop to every node, *e.g.,* GCN (Kipf and Welling, 2017) and GAT (Veličković et al., 2018).

3.2. Graph Neural Networks

We consider Message-Passing Graph Neural Networks (MP-GNNs) which iteratively transform the layer-wise forward-propagation operation of the form:

$$X^{(k+1)} = \sigma(PX^{(k)}W^{(k)}), \tag{1}$$

where P is the aggregation operator, $PX^{(k)}$ denotes message aggregation, $W^{(k)} \in \mathbb{R}^{d\times d}$ is the learnable weight matrix used for weight transformation $\sigma(\cdot W^{(k)})$, σ denotes a pointwise nonlinear activation function and $X^{(k)} = [X_1^{(k)}, X_2^{(k)}, \cdots, X_N^{(k)}]^\top$ represents k-th layer node embedding matrix and $X^{(1)} = X$.

Remark. In this paper, we mainly consider normalized aggregation operators that belong to $\mathcal{P} = \{P \in \mathbb{R}^{N\times N} | \exists\ a > 0\ s.t.\ aP \geq A \text{ and } \rho(P) = 1\}$. It is easy to verify that for any $\alpha \in (0, 1]$, both two versions of normalization Laplacian suffice, *i.e.,* $(I - \alpha L_{sym}), (I - \alpha L_{rw}) \in \mathcal{P}$. Many well-known message-passing GNNs can be included as well, such as GCN (Kipf and Welling, 2017), GAT (Veličković et al., 2018), SGC (Wu et al., 2019), and DGC (Wang et al., 2021).

3.3. Metrics for Oversmoothing

Before we are devoted into the formal discussion, it is necessary to define oversmoothing mathematically. Firstly, we introduce two metrics to quantify the degree of oversmoothing: the Dirichlet energy and Mean Average Distance. The Dirichlet energy which measures the smoothness of a function on a graph (Cai and Wang, 2020) is defined as:

$$E(X^{(k)}) = \mathrm{Tr}(X^{(k)\top} \Delta X^{(k)}), \tag{2}$$

where Δ is some matrix, *e.g.,* L_{sym} and L_{rw}. To mitigate the impact of feature amplitude, the normalized Dirichlet energy $E(\frac{X^{(k)}}{\|X^{(k)}\|})$ is proposed by Giovanni et al. (2023) and Maskey et al. (2023).

For two node vectors $x, y \in \mathbb{R}^d$, we define $\cos(x, y) = \frac{x^T y}{\|x\|_2\|x\|_2}$. And if $x = 0$ or $y = 0$, $\cos(x, y) = 0$. Then Mean Average Distance (MAD) (Chen et al., 2020a) which calculates the average distance betweeen node representations can be defined as:

$$\mathrm{MAD}(X^{(k)}) = \frac{1}{N(N-1)} \sum_{i,j\in[N]} \left(1 - \cos(X_i^{(k)}, X_j^{(k)})\right). \tag{3}$$

Note that MAD is defined on normalized features, hence $\mathrm{MAD}(X^{(k)}) = \mathrm{MAD}(\frac{X^{(k)}}{\|X^{(k)}\|})$. We can define oversmoothing of GNNs with respect to metric $\mu(\cdot)$ ($E(\cdot)$ or $\mathrm{MAD}(\cdot)$) as:

Definition 1 *Graph Neural Networks (GNNs) are oversmoothing for metric $\mu(\cdot)$ if*

$$\lim_{k\to+\infty} \mu\left(\frac{X^{(k)}}{\|X^{(k)}\|}\right) = 0. \tag{4}$$

4. Revisiting Oversmoothing from a Weight Perspective

As introduced in Section 3.2, each layer of Message-Passing GNNs has two parts: message aggregation and weight transformation. So what role do weights play in the oversmoothing problem? In this section, we systematically analyze the role of weights on oversmoothing in three cases: without weights, with random weights, and with freely learnable weights.

4.1. Oversmoothing without Weights

We begin our discussion by considering simply GNNs which only contain message aggregation without weight transformation:

$$X^{(k+1)} = \sigma(PX^{(k)}), k = 1, 2, \ldots \tag{5}$$

Based on Assumption 1, we establish the following proposition which reveals that the representation $X^{(k)}$ exponentially converges to the same value as the depth of the model increases.

Proposition 2 *Under Assumption 1 and without weigths, i.e., $X^{(k+1)} = \sigma(PX^{(k)})$ and $\sigma(\cdot)$ is ReLU or the identity map. For any $P \in \mathcal{P} = \{P \in \mathbb{R}^{N\times N} | \exists a > 0 \text{ s.t., } aP \geq A \text{ and } \rho(P) = 1\}$, we have that P has eigenpair $(1, v)$ where all components of v are positive and any other eigenvalue λ of P satisfies $|\lambda| < 1$. Furthermore,*

$$\lim_{k\to+\infty} X^{(k)} = vc^\top, \quad \textit{for some } c \in \mathbb{R}^d, \tag{6}$$

$$\|X^{(k)} - vc^\top\|_F = O(|\lambda_2|^k), \tag{7}$$

where v is a positive dominant vector of P and λ_2 is the eigenvalue of P with the second maximal module.

We present the proof in Appendix A. The above proposition implies that

$$\lim_{k\to+\infty} E(X^{(k)}) = \lim_{k\to+\infty} E\left(\frac{X^{(k)}}{\|X^{(k)}\|}\right) = \lim_{k\to+\infty} \mathrm{MAD}(X^{(k)}) = 0. \tag{8}$$

Therefore, only considering the graph propagation process without weights, we prove that oversmoothing is indeed inevitable. In addition, the convergence rate of oversmoothing is exponential with the increased depth of GNNs.

4.2. Oversmoothing with Random Weights

In the previous subsection, our analysis reveals that oversmoothing happens under graph propagation without weights. However, another question is what would happen if we consider weight transformations. There are two cases: random weights and learnable weights. Next, let us first turn our attention to the effect of multi-layer random weights. Mathematically, we are interested in the asymptotic behavior of the following system:

$$X^{(k+1)} = X^{(k)}W^{(k)}, \quad k = 1, 2, \ldots, \tag{9}$$

where $\{W^{(k)} \in \mathbb{R}^{d\times d}\}_{k=1}^{\infty}$ are random weight matrices in which each element is independently and identically distributed from the Gaussian distribution $\mathcal{N}(0, \sigma^2)$ or the uniform distribution Uniform$[-a, a]$ for some positive numbers $\sigma, a > 0$. The following proposition tells us that the representation will collapse to a rank one matrix asymptotically.

Proposition 3 *Suppose $X^{(k+1)} = X^{(k)}W^{(k)}$ and all entries of the weight matrices $\{W^{(k)}\}$ are drawn i.i.d. from the Gaussian distribution or the uniform distribution, then $\frac{X^{(k)}}{\|X^{(k)}\|}$ converges to a rank one matrix almost surely.*

Influence of Initialization Schemes. It is worth noting that commonly used Xavier initialization (Glorot and Bengio, 2010) and Kaiming initialization (He et al., 2015) satisfy the condition that all entries of the weight matrices are independently drawn from the Gaussian distributions or the uniform distribution. Therefore, the above proposition reveals a critical fact that such initialization will make the presentation of neural networks degraded. It follows that random weights result in the collapse representation of GNNs. And this makes training deep networks more difficult.

4.3. Oversmoothing with Freely Learnable Weights

In the above, our focus is to consider message aggregation without weights or with random weight transformations respectively, and we have shown that oversmoothing does occur in these cases. Now, let us consider graph propagation with learnable weights:

$$X^{(k+1)} = PX^{(k)}W^{(k)}, \quad k = 1, 2, \dots, \tag{10}$$

where $\{W^{(k)} \in \mathbb{R}^{d\times d}\}_{k=1}^{\infty}$ are learnable paremeters. However, would oversmoothing happen under graph propagation with freely learnable weights as well? Our answer is **NO**, since we find that oversmoothing actually can be avoided with proper weight transformations. In fact, in the following Theorem 4, we reveal that there exists proper weights such that GNNs representations are well distinguished at each layer.

Theorem 4 *Suppose that $\sigma(\cdot)$ is the identity map, i.e., $X^{(k+1)} = PX^{(k)}W^{(k)}$. For $P \in \mathcal{P}$ and $\mathrm{Rank}(X) > 1$, there exist proper feature transformations $\{W^{(k)}\}_{i=1}^{\infty}$ and a constant $a > 0$ such that*

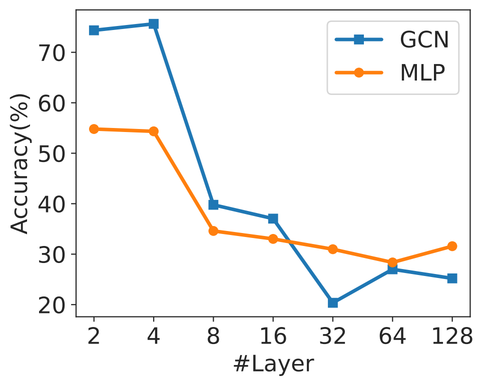
$$E(X^{(k)}),\ E(\frac{X^{(k)}}{\|X^{(k)}\|_F}),\ \mathrm{MAD}(X^{(k)}) > a, \quad \forall k \in \mathbb{N}. \tag{11}$$

Theorem 4 tells us that there exist such weight matrices such that oversmoothing can be avoided in the infinite layer GCN: the oversmoothing metric is always larger than a constant number and does not converge to zero with the increased depth of the model. And we can get similar results for GAT-like dynamic message-passing $X^{(k+1)} = P^kX^{(k)}W^{(k)}$ as bellow:

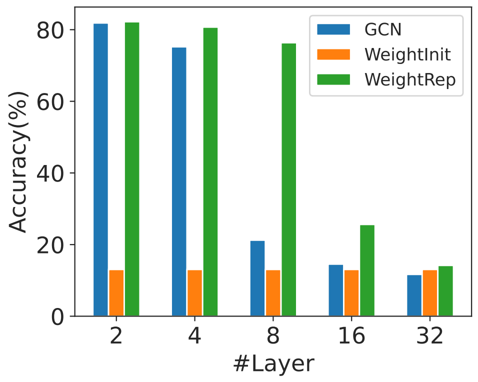
Corollary 5 *Suppose that $\sigma(\cdot)$ is the identity map, i.e., $X^{(k+1)} = P^kX^{(k)}W^{(k)}$. For $P^k \in \mathcal{P}$ and $\mathrm{Rank}(X) > 1$, there exist proper feature transformations $\{W^{(k)}\}_{i=1}^{\infty}$ and a constant $a > 0$ such that*

$$E(X^{(k)}),\ E(\frac{X^{(k)}}{\|X^{(k)}\|_F}),\ \mathrm{MAD}(X^{(k)}) > a, \quad \forall k \in \mathbb{N}. \tag{12}$$

Remark. Above, we theoretically prove the existence of weights that make GNNs not oversmoothing. However, we need to point out that the optimizer cannot find such weights in practice since the optimization of deep networks has always been a difficult problem. Here,



(a) GCN v.s. MLP



(b) Models Performance

Figure 1: Experiments on Cora dataset. (a): The impact of message aggregation and weight transformation when varying the depth of models on Cora dataset. MLP has the same structure as GCN except for the absence of message aggregation. (b): Performance comparison of GCN, WeightInit and WeightRep on Cora dataset with various depths.

we empirically examine the impacts of message aggregation and weight transformation on trained deep models. To control variables, we compare MLP and GCN models on Cora dataset, where MLP has the same structure as GCN except for the absence of message aggregation. From Figure 1(a), we can see that when the performance of the MLP declines, the performance of the GCN also declines, and the trend is consistent. This indicates that the optimization of weights may be the main reason that the performance drops of GCN as the depth increases. For more experiment evidence on other datasets, please see Figure 3 in Appendix C.

5. Building Non-oversmoothing GNNs with Weight Reparameterization

From the theoretical analysis in the previous section, we know that the proper weight transformation which maintains the linearly independent of each layer's presentation is critical in mitigating oversmoothing. In this section, our goal is to leverage this insight to mitigate the oversmoothing problem.

5.1. Failure of Weight Initialization

It seems natural to directly initialize weights (dubbed WeightInit) as the construction in Theorem 4. However, there are some problems with this method. First, since the number of nodes in graph datasets is usually large (see Table 4), the calculation cost of eigenvalue decomposition of the aggregation matrix is very expensive. Second, the existence of the non-linear activation function. Third, weights are constantly updated by the optimizer during the training. As a result, the performance of WeightInit is poor (see Figure 1(b). Next, we explore an input-dependent approach to keep the linearly independent of the representation during the training.

5.2. Proposed Weight Reparameterization

As analyzed in Section 5.1, direct weight initialization fails to preserve the weight structure since the hidden representation $X^{(k)}$ starts to shift once the training begins. To ensure that weights always remain linearly independent of the features during the training, our approach is to dynamically construct **input-dependent weights** from the representation of each layer. Concretely, suppose that $\hat{X}^{(k)} = PX^{(k-1)}W^{(k-1)}$ and $X^{(k)} = \hat{X}^{(k)}\hat{W}^{(k)}$, where $W^{(k-1)}$ is the learnable weight and $\hat{W}^{(k)}$ is the constructed weight such that keeping the linearly independent of the representation $X^{(k)}$.

Construction of the weight $\hat{W}^{(k)}$. Now let us describe in detail how to get the weight $\hat{W}^{(k)}$. Denote $\Sigma = \hat{X}^{(k)\top}\hat{X}^{(k)} \in \mathbb{R}^{d\times d}$ as the covariance matrix of $\hat{X}^{(k)}$. Our goal is making $X^{(k)\top}X^{(k)} = I_d$, *i.e.*, $\hat{W}^{(k)\top}\Sigma\hat{W}^{(k)} = I_d$. Since Σ is a symmetric positive definite matrix [1], suppose Σ has the spectral decomposition $\Sigma = U\Lambda U^\top$, where Λ is a diagonal matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d > 0$ and U is an orthogonal matrix. It is straightforward to verify that we can get an explicit solution

$$\hat{W}^{(k)} = \Sigma^{-1/2} = U\Lambda^{-1/2}U^\top.$$

Then we have the new node representation, *i.e.*, $X^{(k)} = \hat{X}^{(k)}\Sigma^{-1/2}$.

In addition, we can extend the construction of the weight $\hat{W}^{(k)}$ to more generalized cases that can be applied to various scenarios. Given the spectral decomposition of the covariance matrix Σ, the generalized construction of $\hat{W}^{(k)}$ with the function $f(\cdot)$ is defined as:

$$\hat{W}^{(k)} = \Sigma_f = Uf(\Lambda)U^T, \tag{13}$$

where $f(\Lambda)$ is denoted as the function $f(\cdot)$ is applied on every diagonal element of the digonal marix Λ, *i.e.*, $f(\Lambda) = \text{diag}\{f(\lambda_i), f(\lambda_2), \ldots, f(\lambda_d)\}$. And the new node feature is calculated by $X^{(k)} = \hat{X}^{(k)}\hat{W}^{(k)} = \hat{X}^{(k)}\Sigma_f$.

The choices of $f(\cdot)$ can be varied widely, in this paper we mainly consider the power function family, *i.e.*, $f(\lambda) = \lambda^p$. When the power exponent $p = -\frac{1}{2}$, this construction becomes the whitening transformation. In this view, our method WeightRep is a special normalization layer like LayerNorm (Ba et al., 2016) and PairNorm (Zhao and Akoglu, 2020).

Reparameterized Weight. In fact, we can reformulate the representation $X^{(k)}$ as:

$$X^{(k)} = PX^{(k-1)}W^{(k-1)}\hat{W}^{(k)} = PX^{(k-1)}\tilde{W}^{(k)}, \tag{14}$$

where $\tilde{W}^{(k)} = W^{(k-1)}\hat{W}^{(k)}$ is the reparameterized weight. Based on the reformulation in Eq. (14), we can understand the construction of new weights as the weight reparameterization (dubbed **WeightRep**). It follows that reparameterized weights play the same role as in Theorem 4 that reparameterized weights can help keep the linearly independent of the representation $X^{(k)}$. Under this perspective, this implicitly helps us to get such proper weight transformations.

1. In practice, we calculate Σ by $\Sigma = \hat{X}^{(k)\top}\hat{X}^{(k)} + \epsilon I_d$ for the numerical stability, where ϵ is a small positive number (default: $1e-6$). Hence, we can consider Σ as the symmetric positive definite matrix.

5.3. Theoretical Guarantees

In this subsection, we theoretically analyze the oversmoothing behaviors of WeighRep.

Proposition 6 *For WeightRep as in Eq (14) with $p = -\frac{1}{2}$, there exists $a > 0$ such that*

$$E(X^{(k)}),\ E(\frac{X^{(k)}}{\|X^{(k)}\|_F}),\ \mathrm{MAD}(X^{(k)}) > a.$$

Proposition 6 shows that WeightRep can guarantee non-oversmoothing throughout the training. Besides, Guo et al. (2023a) pointed out that features may still suffer from dimensional collapse, whether features lie in a low-dimensional subspace while not fully collapsed, which may be regarded as subtle oversmoothing as well. The effective rank can be used to measure such dimension collapse. As defined below, a lower effective rank means that feature eigenvalues are more concentrated, indicating a higher degree of dimensional collapse.

Definition 7 (Effective Rank (Roy and Vetterli, 2007)) *Suppose $X \in \mathbb{R}^{N\times d}$ has singular value decomposition $X = U\Sigma V^\top$, where Σ is a diagonal matrix with singular values $\sigma_1, \sigma_2, \ldots, \sigma_m \geq 0, m = \min\{N, d\}$. The singular value distribution is $p_i = \frac{\sigma_i}{\sum_{j=1}^m \sigma_j}, \quad j = 1, 2, \ldots, m$. Then the effective rank of the matrix X, denoted as $\mathrm{Erank}(X)$, is defined as $\mathrm{Erank}(X) = \exp\{H(p_1, p_2, \ldots, p_m)\}$, where $H(p_1, p_2, \ldots, p_m)$ is the Shannon entropy which defined as $H(p_1, p_2, \ldots, p_m) = -\sum_{i=1}^m p_i \log(p_i)$.*

Proposition 8 *Suppose $X = \hat{X}\Sigma_f$ with $f(\lambda) = \lambda^p$ and $-\frac{1}{2} \leq p < 0$, then we have*

$$\mathrm{Erank}(X) > \mathrm{Erank}(\hat{X}). \tag{15}$$

The proposition above characterizes the impact of WeightRep on the effective rank of the representation. The rise of the effective rank after WeightRep indicates that representations of different nodes will be more dilated, thus avoiding dimensional collapse.

5.4. Implementation and Complexity Analysis

Empirically, we think of WeightRep as a gradientless operation in which the constructed weight $\hat{W}^{(k)}$ does not require the gradient. So we disable the gradient calculation of the operator in Eq (13). This is crucial in saving training time and reducing memory consumption for computations. Therefore, the calculation cost of WeightRep is mainly caused by the spectral decomposition which does not require the gradient. It follows that the additional computational complexity is $O(Nd^2 + d^3)$, which is linearly dependent on the number of nodes. And if $N > d$, the additional computational complexity is the same as the linear transformation $O(Nd^2)$.

For GCN, we add WeightRep to every layer except the last layer like PairNorm (Zhao and Akoglu, 2020) and ContraNorm (Guo et al., 2023a). Therefore, suppose the depth of GCN is L, the total computational complexity is $O(LNd^2 + Ld^3)$.

6. Experiments

In this section, we design extensive experiments to evaluate the effectiveness of our proposed method WeightRep. We mainly focus on node classification including homophily graphs and heterophily graphs. Moreover, we conduct ablative studies on hyperparameters sensitivity analysis. For more details experiment setting and adding experiments can be found in Appendix B & C

6.1. Experiment Setup

Datasets. We evaluate our method WeightRep on nine common graph node classification datasets varying in graph size and feature type, including homophily graphs and heterophily graphs. For the homophily benchmark, we choose the citation datasets Cora, CiteSeer, and Pubmed (Yang et al., 2016), the Amazon co-purchase graphs Computers and Photo (Shchur et al., 2018), and the Coauthor datasets CS and Physics (Shchur et al., 2018). For the heterophily benchmark, we choose Wikipedia graphs Chameleon and Squirrel (Rozemberczki et al., 2021). Their data statistics are summarized in Table 4.

Setup. For all methods, we use standard GCNs (Kipf and Welling, 2017) as the backbone. Following the previous settings of Zhao and Akoglu (2020), the hidden dimension of GCNs and dropout rate are set to 32 and 0.6, respectively. We initialize the model parameters randomly and use the Adam optimizer (with weight decay $5e-4$) to train the encoder (Kingma and Ba, 2015). We run each experiment within 200 epochs. All experiments are run with 5 random seeds and the average performance and standard deviation are reported. For three citation networks data, we use the public and standard splits as in Kipf and Welling (2017). For Amazon co-purchase graph Computers and Photo, and the Coauthor datasets CS and Physic, we use the same setting as He et al. (2021), who randomly split the node sets into train, validation and test set ratio 60%, 20% and 20%. For Wikipedia graphs Chameleon and Squirrel, we use pre-processed data introduced in Pei et al. (2019), so train, validation, and test splits are available. All our experiments are performed on a single 24GB Nvidia GeForce RTX 3090.

Hyperparameters. For each classification task and model with different depths, we select the best power exponent p in the range of $\{-0.2, -0.3, -0.4, -0.5\}$. For a fair comparison, we select the best configuration of hyperparameters for all methods only based on the accuracy of the validation set.

6.2. Experimet Results

The test accuracies of GCNs with various depths, 2, 4, 8, and 16, are shown in Table 1 (The results of 64 hidden dimensions and the MAD metric are shown in Appendix C). The results show that our method WeightRep significantly outperforms the baseline across the nine datasets, both homophily graphs and heterophily graphs. First, we can see that our method can improve the performance of 2-layer GCN which usually has the best performance. For instance, our WeightRep boosts the performance of 2-layer GCN 10.04% and 14.87% on Computers and Photo datasets, respectively. Furthermore, one can see that our method can also improve deeper GCNs. On Cora dataset, for example, the proposed method can improve the vanilla GCNs with 4 layers and 8 layers by a margin of 5.50 % and 55.12%,

Table 1: Node classification accuracies (%) on nine datasets. We use GCN as the backbone with various depths: 2, 4, 8, 16. The hidden dimension is set to 32. Reported results are averaged over 5 runs. For every layer setting, the highest accuracy is in bold.

Datasets	Model	#L=2	#L=4	#L=8	#L=16
Cora	GCN	81.86 ± 0.43	75.21 ± 2.71	21.23 ± 7.99	14.54 ± 8.67
	WeightRep	**82.21 ± 0.11**	**80.71 ± 0.35**	**76.35 ± 2.31**	**25.62 ± 5.89**
CiteSeer	GCN	68.96 ± 0.51	53.74 ± 4.13	19.65 ± 0.00	19.65 ± 0.00
	WeightRep	**69.04 ± 0.33**	**65.56 ± 0.24**	**57.63 ± 3.16**	**34.38 ± 2.72**
Pubmed	GCN	**77.94 ± 0.12**	**75.96 ± 0.51**	20.8 ± 0.00	32.02 ± 9.16
	WeightRep	77.68 ± 0.12	75.58 ± 0.36	**75.23 ± 0.80**	**73.41 ± 1.71**
Computers	GCN	75.99 ± 3.47	38.24 ± 27.21	25.79 ± 25.64	9.46 ± 3.98
	WeightRep	**86.03 ± 0.19**	**81.95 ± 0.38**	**76.78 ± 0.86**	**59.71 ± 2.34**
Photo	GCN	77.50 ± 4.65	48.99 ± 28.49	26.74 ± 16.30	13.05 ± 13.61
	WeightRep	**92.37 ± 0.15**	**90.73 ± 0.39**	**88.30 ± 0.42**	**69.69 ± 5.84**
CoauthorCS	GCN	93.25 ± 0.33	72.30 ± 14.99	2.58 ± 2.32	1.42 ± 0.00
	WeightRep	**93.81 ± 0.13**	**92.49 ± 0.07**	**91.24 ± 0.14**	**84.47 ± 0.57**
CoauthorPhysics	GCN	95.74 ± 0.10	95.97 ± 0.11	91.53 ± 3.71	84.3 ± 0.00
	WeightRep	**96.62 ± 0.09**	**96.15 ± 0.07**	**95.7 ± 0.12**	**94.76 ± 0.18**
Chameleon	GCN	47.89 ± 1.50	36.89 ± 1.03	22.37 ± 0.00	22.37 ± 0.00
	WeightRep	**63.38 ± 0.95**	**53.42 ± 1.52**	**45.79 ± 1.76**	**35.22 ± 1.04**
Squirrel	GCN	28.66 ± 0.83	19.31 ± 0.00	19.31 ± 0.00	19.31 ± 0.00
	WeightRep	**45.90 ± 0.44**	**38.27 ± 1.02**	**34.33 ± 0.77**	**26.07 ± 1.45**

respectively. Notably, although WeightRep resolves oversmoothing at each layer and helps improve the performance, we still observe some degradation under more depth. It indicates that oversmoothing might not explain all sources of performance degradation. For example, since the training becomes harder under more depth, the optimizer might not be able to find good discriminative features for classification, which is not fully captured in the oversmoothing notion. We leave more in-depth study on this problem to future work.

6.3. Comparison with other Designs

In this subsection, we compare our method with the following normalization techniques which tackle the oversmoothing problem: LayerNorm (Ba et al., 2016), PairNorm (Zhao and Akoglu, 2020) and ContraNorm (Guo et al., 2023a). For a fair comparison, for PairNorm and ContraNorm, we tune the hyperparameter scaler s from $\{0.2, 0.5, 0.8, 1.0\}$ as in the paper (Guo et al., 2023a). The comparison results are presented in Table 2. From the

Table 2: Test accuracy (%) comparison of different normalizations. The highest accuracy is in bold and the second one is underlined. OOM: out of memory.

Datasets	Model	#L=2	#L=4	#L=8	#L=16
CiteSeer	GCN	<u>68.96</u>	53.74	19.65	19.65
	LayerNorm	64.89	61.74	25.4	19.65
	PairNorm	46.38	40.17	39.57	24.09
	ContraNorm	63.67	<u>60.69</u>	**60.18**	**43.63**
	WeightRep	**69.04**	**65.56**	<u>57.63</u>	<u>34.38</u>
Pubmed	GCN	**77.94**	**75.96**	20.8	32.02
	LayerNorm	75.04	73.82	43.51	35.97
	PairNorm	72.81	67.99	<u>64.05</u>	<u>72.27</u>
	ContraNorm	75.42	OOM	OOM	OOM
	WeightRep	<u>77.68</u>	<u>75.58</u>	**75.23**	**73.41**

table, we can see that our proposed WeightRep outperforms LayerNorm and PairNorm on CiteSeer and Pubmed datasets. Compared to ContraNorm, our method performs better in shallow layers and can scale to larger graph datasets. More results can be found in Table 8.

6.4. Computational Time and GPU Memory

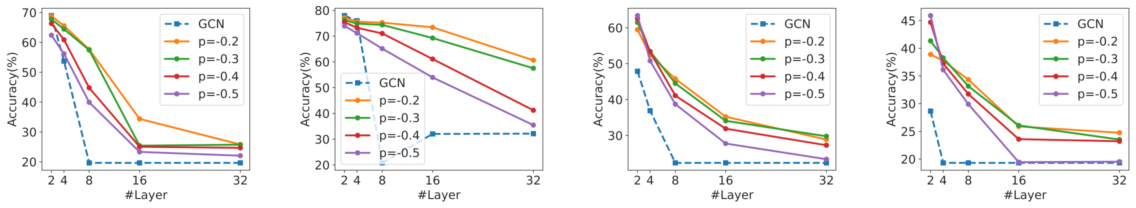
Here we compare different models' training speed and GPU memory usage under the same setting. The models are set to 8 layers and are trained in 200 epochs. All models are evaluated on Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz and a single 24GB Nvidia GeForce RTX 3090. The comparison results are listed in Table 3. From the table, one can see that our WeightRep is slightly slower than vanilla GCN and PairNorm, and the GPU memory usage of our method only increases a little. Compared to ContraNorm, our method performs better on both the training time and GPU memory. In other words, the cost of adding WeightRep into GCN is acceptable.

6.5. Hyperparameter Analysis on Power Exponent

To understand the impacts of the hyperparameter p on different layer GCNs (2, 4, 8, 16 and 32) and datasets (CiteSeer, Pubmed, Chameleon, and Squirrel), we conduct experiments with different power exponent p. For all datasets and models, we vary the power exponent p in the range of $\{-0.2, -0.3, -0.4, -0.5\}$, and other parameters remain consistent. We present the hyperparameter analysis in Figure 2. From the figure, we observe that when $p = -0.2$ or -0.3, models usually achieve the best performance for all layers. Therefore, our WeigtRep is robust to the power exponent p. In general, we find that $p = -0.2$ is a good choice for most GCN with various depths and datasets.

Table 3: Training time and memory comparison of different methods on CiteSeer and Pubmed datasets. OOM: out of memory.

Datasets	Model	Training time	GPU memory
CiteSeer	GCN	5.59s	1582MiB
	PairNorm	6.99s	1624MiB
	ContraNorm	8.91s	3168MiB
	WeightRep	8.27s	1702MiB
Pubmed	GCN	5.93s	1640MiB
	PairNorm	7.06s	1830MiB
	ContraNorm	/	OOM
	WeightRep	8.33s	1758MiB



(a) CiteSeer (b) Pubmed (c) Chameleon (d) Squirrel

Figure 2: Performance when varying the power exponent p on CiteSeer, Pubmed, Chameleon and Squirrel datasets. Best viewed in color.

7. Conclusions

In this paper, we proposed a new understanding of oversmoothing from a weight perspective. Different from the common conclusions of existing theories, we first showed that if learned freely, vanilla GNNs (e.g., GCN, GAT, SGC) can provably escape oversmoothing with the existence of ideal weights. Inspired by the theoretical analysis, we then proposed WeightRep to mitigate oversmoothing and provided theoretical guarantees on WeightRep. Experimental results demonstrated that GCN employing WeightRep can outperform the vanilla GCN even with normalization on various depth settings. Our analysis challenged common beliefs of GNN oversmoothing, and it paved a new understanding of oversmoothing from the learning perspective, opening up new avenues for future research.

Acknowledgments

Yisen Wang was supported by National Key R&D Program of China (2022ZD0160300), National Natural Science Foundation of China (92370129, 62376010), Beijing Nova Program (20230484344), and CCF-Baichuan-EB Fund. Jinwen Ma was supported by the Natural

Science Foundation of China under grant 62071171 and the high-performance computing platform of Peking University.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Philippe Bougerol et al. *Products of random matrices with applications to Schrödinger operators*, volume 8. Springer Science & Business Media, 2012.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *ICLR*, 2013.

Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *AAAI*, 2020a.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *ICML*, 2020b.

Hadi Daneshmand, Jonas Kohler, Francis Bach, Thomas Hofmann, and Aurelien Lucchi. Batch normalization provably avoids ranks collapse for randomly initialised deep networks. In *NeurIPS*, 2020.

Taoran Fang, Zhiqing Xiao, Chunping Wang, Jiarong Xu, Xuan Yang, and Yang Yang. Dropmessage: Unifying random dropping for graph neural networks. In *AAAI*, 2023.

Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural networks for semi-supervised learning on graphs. In *NeurIPS*, 2020.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

Francesco Di Giovanni, James Rowbottom, Benjamin Paul Chamberlain, Thomas Markovich, and Michael M. Bronstein. Understanding convolution on graphs via energies. *Transactions on Machine Learning Research*, 2023.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

Xiaojun Guo, Yifei Wang, Tianqi Du, and Yisen Wang. Contranorm: A contrastive learning perspective on oversmoothing and beyond. In *ICLR*, 2023a.

Xiaojun Guo, Yifei Wang, Zeming Wei, and Yisen Wang. Architecture matters: Uncovering implicit mechanisms in graph contrastive learning. In *NeurIPS*, 2023b.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

Mingguo He, Zhewei Wei, Hongteng Xu, et al. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. In *NeurIPS*, 2021.

IN Herstein. A note on primitive matrices. *The American Mathematical Monthly*, 61(1): 18–20, 1954.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Nicolas Keriven. Not too little, not too much: a theoretical analysis of graph (over)smoothing. In *LoG*, 2022.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*, 2019.

Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. DeepGCNs: Can gcns go as deep as cnns? In *CVPR*, 2019.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, 2018.

Sohir Maskey, Raffaele Paolino, Aras Bacho, and Gitta Kutyniok. A fractional graph laplacian approach to oversmoothing. In *NeurIPS*, 2023.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *ICLR*, 2020.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *ICLR*, 2019.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. DropEdge: Towards deep graph convolutional networks on node classification. In *ICLR*, 2019.

Andreas Roth and Thomas Liebig. Rank collapse causes over-smoothing and over-correlation in graph neural networks. In *LoG*, 2023.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *EUSIPCO*, 2007.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 2021.

T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *Relational Representation Learning Workshop, NeurIPS*, 2018.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *KDD*, 2018.

Yifei Wang, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. Dissecting the diffusion process in linear graph convolutional networks. In *NeurIPS*, 2021.

Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. In *ICML*, 2019.

Xinyi Wu, Amir Ajorlou, Zihui Wu, and Ali Jadbabaie. Demystifying oversmoothing in attention-based graph neural networks. In *NeurIPS*, 2023.

Zhaoping Xiong, Dingyan Wang, Xiaohong Liu, Feisheng Zhong, Xiaozhe Wan, Xutong Li, Zhaojun Li, Xiaomin Luo, Kaixian Chen, Hualiang Jiang, et al. Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of medicinal chemistry*, 2019.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *ICML*, 2018.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, 2016.

Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE transactions on intelligent transportation systems*, 2019.

Lingxiao Zhao and Leman Akoglu. PairNorm: Tackling oversmoothing in gnns. In *ICLR*, 2020.

Kuangqi Zhou, Yanfei Dong, Kaixin Wang, Wee Sun Lee, Bryan Hooi, Huan Xu, and Jiashi Feng. Understanding and resolving performance degradation in graph convolutional networks. In *CIKM*, 2021.

Zhijian Zhuo, Yifei Wang, Jinwen Ma, and Yisen Wang. Towards a unified theoretical understanding of non-contrastive learning via rank differential mechanism. In *ICLR*, 2023.