EXPLORING DEEPER GRAPH CONVOLUTIONS FOR SEMI-SUPERVISED NODE CLASSIFICATION

Ashish Tiwari *Richeek Das* *Shanmuganathan Raman*

CVIG Lab, Indian Institute of Technology Gandhinagar
ashish.tiwari@iitgn.ac.in, das.richeek01@iitb.ac.in, shanmuga@iitgn.ac.in

ABSTRACT

Graph convolutional networks (GCNs) have achieved impressive performance in learning from graph-structured data. Although GCN and its variants have shown promising results, they continue to remain shallow as their performance drops with an increasing number of layers - a problem popularly known as *oversmoothing*. This work introduces a simple yet effective idea of feature gating over graph convolution layers to facilitate deeper graph neural networks and address oversmoothing. The proposed feature gating is easy to implement without changing the underlying network architecture and is broadly applicable to GCN and almost any of its variants. Further, we demonstrate the use of feature gating in assigning importance to node features and the nodes for the node classification task. Quantitative analysis on real-world datasets shows that feature gating paves the way for constructing deeper GCNs.

***Index Terms*—** Graph Convolutional Networks, Oversmoothing, Gated Convolution.

1. INTRODUCTION

Graph Convolution Networks (GCNs) have gained much momentum in recent years in handling graph-structured data that is ubiquitous throughout the natural and the social sciences. Kipf & Welling first introduced convolutions to graph-structured data as GCN [1] for semi-supervised node classification. Since then, several other variants such as GraphSAGE [2], GAT [3], SGC [4], and GMNN [5] have been developed. These networks have been deployed in several exciting applications such as social analysis [6, 7], traffic prediction [8, 9], recommender systems [10], and of course, computer vision [11]. Despite such an enormous success and wide applicability, GNNs have remained shallow to the extent of being just 2-layer deep (GCN [1] and GAT [3]). While CNNs experience performance enhancement by stacking more layers and including non-linearities, surprisingly, a similar treatment to GCNs leads to performance degradation. The performance reduction is generally due to overfitting, vanishing gradients, and oversmoothing or their combination. While the first two factors have been addressed to some extent, oversmoothing still remains a challenge [12, 13]. Drawing inspiration from the developments in CNNs, the first two solutions that arise naturally are increasing depth and including residual connections. As per [14], the underlying node representations in GCN converge either to a certain value, or a smaller subspace, thus get difficult to distinguish as the number of layers increases. ResNet [15] is known to tackle this problem in CNNs using residual connections for several computer vision applications. However, applying residual connections to GCNs leads to prohibitive memory cost [16] and merely slows down oversmoothing [1]. Further, several other methods attempt to tackle oversmoothing by either randomly dropping a fraction of edges [13] or introducing a normalization scheme [12] to prevent feature mixing across different clusters or using identity mapping and residual connection [17]. While [17] is observed to tackle oversmoothing by a considerable amount, others merely slow down the performance drop with increasing network depth.

Inspired by the success of gated convolutions [18] in CNNs, we propose to incorporate a simpler form of feature gating in graph convolutions. We perform a quantitative evaluation on standard datasets to show that feature gating, when incorporated in the existing deep GCN models, helps obtain a better performance. Further, we show how feature gating in a typical graph convolution layer can help assign importance scores to nodes and node features to foster appropriate feature selection while training GCNs. We also provide t-SNE visualization to understand the interpretation of node importance. We empirically show that identity mapping over the learnable weight matrices and a modified form of residual connection along with feature gating enables us to design deeper graph networks and achieve better performance for semi-supervised node classification.

2. RELATED WORK

Several works have proposed exciting ways to reduce the extent of oversmoothing in the recent past. By using dense skip connections, JKNet [19] combines the output of each layer to retain local information at the nodes. DropEdge [13] modifies the underlying graph structure at every epoch during training by randomly dropping a certain fraction of edges from the input graph to reduce oversmoothing and ensure robustness in the model. Zhao *et al.* in [12] introduced a normalization scheme called PairNorm such that the total pairwise feature distances remain constant across different layers. This causes distant pairs to have less similar features and prevents feature mixing across clusters. Graph Neural Diffusion (GRAND) [20] considers learning on graphs as a diffusion process and GNNs as discretizations of an underlying PDE to address oversmoothing. While these methods slow down the performance drop with the increasing number of layers, the accuracy still goes down with increasing depth in general. SGC [4] introduced deep propagation in the existing shallow versions by including the k^{th} power of the graph convolution matrix in one layer. This helps the layer to capture information from higher-order neighborhoods. Klicpera *et al.* [21] replaced the power of graph convolution matrix by Personalised PageRank to handle oversmoothing and extended it to arbitrary graph diffusion process [22]. However, these methods cannot handle deep non-linear architectures as they rely on a linear combination of neighborhood features in every layer. Most GNNs adopt a message-passing scheme to learn node representation using the same coefficients to aggregate

[1]This work was carried out with the support of SERB MATRICS grant.

information from each feature dimension. Recently, Jin *et al.* [23] introduced Graph Feature Gating Network (GFGN) that shows performance improvement by considering each feature dimension differently during the aggregation. However, the formulation is more complex and not readily applicable to all the GCN variants. Further, Chen *et al.* proposed identity mapping to enhance network performance in GCNII [17]. Inspired by the propositions in [17], we propose the inclusion of a simple feature gating mechanism along with identity mapping and modified residual connections to address oversmoothing.

3. METHOD

This section discusses the effects of different methods to handle the oversmoothing problem and finally introduces feature gating in graph convolutions.

3.1. Discussion on Mathematical Aspects of Oversmoothing

3.1.1. Mathematical Notations

Consider a simple undirected graph $\mathcal{G} = (V, E)$ with N nodes, an adjacency matrix $\mathbf{A}$, and a diagonal degree matrix $\mathbf{D}$. Introducing self-loops to every node in the graph results in $\widetilde{\mathcal{G}} = (V, \widetilde{E})$ with adjacency matrix $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and degree matrix $\widetilde{\mathbf{D}} = \mathbf{D} + \mathbf{I}$, where $\mathbf{I}$ is an $n \times n$ identity matrix. The node feature matrix $\mathbf{X} \in \mathrm{I\!R}^{\mathrm{N} \times \mathrm{d}}$ defines a d-dimensional feature vector (say $\mathbf{x}_v$) for each node v in the graph. Further, the normalized graph laplacian matrix $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ is a symmetric positive semi-definite matrix with the eigen-decomposition $\mathbf{U\Lambda U^T}$. $\mathbf{\Lambda}$ and $\mathbf{U}$ are the diagonal matrix of eigenvalues and eigenvector matrix of $\mathbf{L}$, respectively.

3.1.2. Spectral Graph Convolution

The spectral convolution on graphs between a signal $\mathbf{x} \in \mathrm{I\!R}^{\mathrm{N}}$ (scalar for every node) and a filter $g_\theta = diag(\theta)$ is given by $g_\theta \star \mathbf{x} = \mathbf{U}\mathbf{g}_\theta\mathbf{U}^T$, where parameter $\theta \in \mathrm{I\!R}^{\mathrm{n}}$ is the vector corresponding to the filter coefficients. Here, understand g_θ as a function of the eigenvalues of $\mathbf{L}$, i.e. $g_\theta(\mathbf{\Lambda})$. As per [24], $g_\theta(\mathbf{\Lambda})$ is approximated by an expansion of Chebyshev polynomials $T_k(x)$ up to K^{th} order as $g_\theta(\mathbf{\Lambda}) \approx \sum_{k=0}^{K} \theta_k T_k(\widetilde{\mathbf{\Lambda}})$. Here, $\widetilde{\mathbf{\Lambda}} = \frac{2}{\lambda_{max}}\mathbf{\Lambda} - \mathbf{I}$ and λ_{max} is the largest eigenvalue of $\mathbf{L}$. Therefore, $g_\theta(\mathbf{\Lambda}) \star \mathbf{x} \approx \sum_{k=0}^{K} \theta_k T_k(\widetilde{\mathbf{L}})\mathbf{x}$ with $\widetilde{\mathbf{L}} = \frac{2}{\lambda_{max}}\mathbf{L} - \mathbf{I}$. This expression captures the information from the K-hop neighborhood of the central node.

3.1.3. Graph Convolution Layer

By setting $K = 1$, $\theta_0 = 2\theta$, and $\theta_1 = -\theta$, Kipf and Welling in [1] formulated the graph convolution operation as $g_\theta \star \mathbf{x} = \theta(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{x}$. They further proposed a normalization trick and replaced $\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ with $\widetilde{\mathbf{P}} = \widetilde{\mathbf{D}}^{-1/2}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-1/2}$.Finally, the graph convolutional layer is represented as $\mathbf{H}^{(l+1)} = \sigma(\widetilde{\mathbf{P}}\mathbf{H}^{(l)}\mathbf{W}^{(l)})$, where σ represents ReLU activation. However, stacking up K such layers approximates a K^{th}-order polynomial filter $\widetilde{\mathbf{P}}^K\mathbf{x}$ with fixed coefficients θ in the spectral domain [4]. Essentially, oversmoothing is caused by the fact that such a fixed polynomial filter $\widetilde{\mathbf{P}}^K\mathbf{x}$ converges to a distribution that is distant from the input feature $\mathbf{x}$ and hence, incurs vanishing gradients.

Observation 1: *The fixed coefficients tend to limit the expressive power of a multi-layer GCN model and leads to oversmoothing* [17].

To understand this, let us consider for $M \leq N$, let V be an M-dimensional subspace of $\mathrm{I\!R}^{\mathrm{N}}$ denoting the eigenspace associated with the smallest eigenvalue of normalised graph laplacian $\mathbf{L}$. Let $\mathcal{M}$ be the subspace of $\mathrm{I\!R}^{\mathrm{N} \times \mathrm{C}}$ such that $\mathcal{M} : V \otimes \mathrm{I\!R}^{\mathrm{C}}$ and $\otimes$ denotes the Kronecker product. Further, let $d_{\mathcal{M}}(\mathbf{H}) := inf\{||\mathbf{H} - \mathbf{Y}||_F | \mathbf{Y} \in \mathcal{M}\}$ denote the distance between $\mathbf{H}$ and $\mathcal{M}$. Given that s is the maximum singular value of $\mathbf{W}^{(l)}$ and λ is the largest eigenvalue of normalised graph laplacian, the following holds, as per [25].

Observation 2: *For any initial value $\mathbf{H}^{(0)}$, the output of l^{th} layer $\mathbf{H}^{(l)}$ satisfies $d_{\mathcal{M}}(\mathbf{H}^{(l)}) \leq (s\lambda)^l d_{\mathcal{M}}(\mathbf{H}^{(0)})$.*

This can be interpreted as the exponential loss of information in GCNs with increasing depth. At any stage, if two nodes $i, j \in V$ belong to the same connected component and their degrees are identical, the corresponding column vectors of $\mathbf{H}^{(l)}$ will be near identical thus, making the nodes indistinguishable. Due to this, the GCN output carries no information other than the node degrees and connected components, especially when the layer size grows to infinity [25]. One way to address this is to carry the input feature $\mathbf{H}^{(0)}$ along with the graph structure to ensure that the GCN does not suffer from oversmoothing even if the number of layers grows to infinity. Further, keeping $s\lambda \geq 1$ can help by slowing down the convergence to $\mathcal{M}$. For $\lambda = 1$, we can have $s \geq 1$ and for $\lambda \neq 1$, the sweet spot falls in the range $1 \leq s \leq \lambda^{-1}$ [25].

3.1.4. Information Aggregation

Shallow networks fail to span enough neighbourhood around a given node and hence, lag in understanding the entire structure of the graph. Recent work on Personalised PageRank [22] demonstrates the aggregation of information from multi-hop neighbourhood without actually increasing the number of layers in the network. It defines the graph convolution as shown in Equation 1.

$$\mathbf{H}^{(l+1)} = (1 - \alpha_l)\widetilde{\mathbf{P}}\mathbf{H}^{(l)} + \alpha_l\mathbf{H}^{(0)} \tag{1}$$

Here, $0 \leq \alpha_l \leq 1$ represents the fraction of initial features $\mathbf{H}^{(0)}$ available at the l^{th} layer. $\mathbf{H}^{(0)}$ need not always be the initial feature matrix $\mathbf{X}$. In case the feature dimension d is large, a lower-dimensional feature representation $\mathbf{H}^{(0)}$ is obtained by applying a fully-connected neural network on $\mathbf{X}$ before the actual forward propagation. In addition to spanning a multi-hop neighbourhood, this formulation has other key benefits. (i) The final representation of each node contains some fraction of information from the input layer as we continue to stack many layers [17]. (ii) The learned node representations of the output of each layer tend to be separated in (nearly) the same sense as the initial node features. This is required such that the total pair-wise squared distance (TPSD) remains constant across all layers [12] to slow down the convergence to $\mathcal{M}$ (*Observation 2).*

3.1.5. Residual Connection

The residual connections have led to an increased performance in CNNs. However, they merely slow down the rate of performance drop when applied to GCNs [16]. This motivates the search for a proper formulation to apply residual connections to GCNs. A possible solution arises from linear ResNet [26, 4] of the form $\mathbf{H}^{(l+1)} = \mathbf{H}^{(l)}(\mathbf{W}^{(l)} + \mathbf{I})$. The idea here is to add the identity matrix $\mathbf{I}$ to the learnable weight matrix $\mathbf{W}^{(l)}$ at the l^{th} layer. Instead of simple addition, a weighted addition $\beta_l\mathbf{W}^{(l)} + (1 - \beta_l)\mathbf{I}$ enables weight matrices $\mathbf{W}^{(l)}$ to have smaller norms by imposing regularization on them to avoid overfitting. The rate of convergence to the subspace $\mathcal{M}$ is proportional to s^K ($\lambda = 1$) for a K-layer GCN. The identity mapping ensures the maximum singular value s to remain closer to 1, thereby keeping s^K large and relieving the information loss.

3.2. Feature Gating

Feature gating has been extensively used in the area of speech [27], language [28], and also in vision [29, 30, 31] processing. Yu et al. [18] first introduced this formulation as gated convolution for the image inpainting task. Gated convolutions have been applied to GNNs to produce sequential outputs [32, 33]. However, their ability to foster deeper GNNs has not been explored much, to the best of our knowledge. A simple modification can incorporate feature gating in a typical GCN, as described by Equation 2.

$$\mathbf{H}^{(l+1)} = \widetilde{\mathbf{P}} \left(\phi(\mathbf{H}^{(l)}\mathbf{W}^{(l)}) \odot \sigma(\mathbf{H}^{(l)}\mathbf{G}^{(l)}) \right) \tag{2}$$

Here, $\mathbf{W}$ and $\mathbf{G}$ are the learned weight matrices. ϕ denotes the ReLU activation function, and σ denotes the sigmoid function, which provides gating values for node features between 0 and 1. $\odot$ denotes the element-wise Hadamard product. We extend this formulation to other GCN variants as well and demonstrate the performance in Section 4.

Although the formulation described in Equation 1 considers the linear combination of neighbourhood features, the inclusion of non-linearity (like ReLU) can make it suitable for handling non-linearities in deep networks. However, the repeated application of non-linearity will lead to overfitting [25, 22]. To avoid overfitting, a deep network must focus on the most relevant parts of the input and dynamically select relevant node features that are actually essential for making decisions. This is facilitated in CNNs using gated convolutions, and on the same lines, we propose to use feature gating in the graph convolutions. Motivated by the *Observation 1 & 2* discussed in Section 3.1 and propositions in [17], we now introduce a formulation of graph convolution that incorporates the benefits of identity mapping and a modified form of residual connection and denote the formulation by GCN-IR, as described in Equation 3.

$$\mathbf{H}^{(l+1)} = \beta_l \left(\widetilde{\mathbf{P}}\mathbf{Z}^{(l)}\mathbf{W}^{(l)} \right) + (1-\beta_l)\widetilde{\mathbf{P}}\mathbf{Z}^{(l)}\mathbf{I} \tag{3}$$

Here, $\mathbf{Z}^{(l)} = (1-\alpha_l)\mathbf{H}^{(l)} + \alpha_l \mathbf{H}^{(0)}$. α_l and β_l are weights assigned to initial features $\mathbf{H}^{(0)}$ and the identity mapping, respectively. The proposed formulation enables GCN to preserve the information from input feature $\mathbf{H}^{(0)}$ along with the graph structure. It is essential for retaining the discriminative power of GCN and avoid oversmoothing. The decay of the weight matrix must increase as we keep stacking the layers for better regularization. Hence, in practice, $\beta_l \approx \frac{\gamma}{l}$, where $\gamma = 0.5$ is a hyper-parameter. We extend Equation 3 to introduce our formulation: GCN-IR-FG, which includes feature gating to enable the design of deeper GCNs in Equation 4.

$$\begin{aligned}\mathbf{H}^{(l+1)} = \ & \beta_l \left[\widetilde{\mathbf{P}} \left(\phi(\mathbf{Z}^{(l)}\mathbf{W}^{(l)}) \odot \sigma(\mathbf{Z}^{(l)}\mathbf{G}^{(l)}) \right) \right] + \\ & (1-\beta_l)\widetilde{\mathbf{P}}\mathbf{Z}^{(l)}\mathbf{I}\end{aligned} \tag{4}$$

3.3. Feature Selection via Gating

The whole idea of feature gating revolves around appropriate feature selection. Interestingly, feature gating in graph convolutions can help identify a set of node features that are crucial for making a class prediction for a node and assigning an importance score to the node itself. Let us understand this mathematically by considering GCN with feature gating (GCN-FG), as described in Equation 2. Taking $l = 0$ for simplicity, Equation 2 can be rewritten as shown in Equation 5.

$$\mathbf{H}^{(1)} = \widetilde{\mathbf{P}} \left(\phi(\mathbf{XW}) \odot \sigma(\mathbf{XG}) \right) \tag{5}$$

Here, $\mathbf{H}^{(0)} = \mathbf{X}$, the initial node feature matrix such that $\mathbf{X} \in \mathbb{R}^{\mathrm{N}\times\mathrm{d}}$, $\mathbf{W} = \mathbf{W}^{(0)} \in \mathbb{R}^{\mathrm{d}\times\mathrm{F}}$, and $\mathbf{G} = \mathbf{G}^{(0)} \in \mathbb{R}^{\mathrm{d}\times\mathrm{F}}$. Let us start by looking at the product $\mathbf{B} = \mathbf{XW}$. Let $\mathbf{x}_i \in \mathbb{R}^{\mathrm{N}\times 1}$ denotes the column vector corresponding to i^{th} column of $\mathbf{X}$ and w_{ij} is the $(i,j)^{th}$ entry of $\mathbf{W}$. Hence, $\mathbf{b}_j \in \mathbb{R}^{\mathrm{N}\times 1}$, the j^{th} column of $\mathbf{B}$ is given by Equation 6.

$$\mathbf{b}_j = \sum_{i=1}^{d} w_{ij}\mathbf{x}_i \quad , \quad j = 1, 2, ..., F \tag{6}$$

Every node is associated with a d-dimensional node feature vector (atleast in the initial representation). How can w_{ij} explain the importance of the i^{th} entry (feature) in that d-dimensional feature vector for all the nodes? A careful observation reveals that w_{ij} effectively indicates the weight of i^{th} feature in forming the j^{th} feature of all the nodes in the next layer, and subsequently the final representation. Further, since Equation 4 involves taking Hadamard product with $\mathbf{S} = \sigma(\mathbf{XG})$, each entry s_{ij} of $\mathbf{S}$ will be in the range [0,1]. Therefore,

$$\mathbf{b}_j \odot \mathbf{s}_j = \sum_{i=1}^{d} w_{ij}(\mathbf{x}_i \odot \mathbf{s}_j) \tag{7}$$

Here, $\mathbf{s}_j$ is the j^{th} column of $\mathbf{S}$. Collectively, $w_{ij}s_{kj}$ represents the importance of i^{th} feature of the k^{th} node in forming the j^{th} feature of the same node after aggregation in the next layer.

In general, for the l^{th} layer with $\mathbf{H}^{(l)} \in \mathbb{R}^{\mathrm{N}\times\mathrm{d}}$ and $\mathbf{W}^{(l)}, \mathbf{G}^{(l)} \in \mathbb{R}^{\mathrm{d}\times\mathrm{F}}$ we note the following.

(i) **Node feature importance.** The importance of the m^{th} feature of the n^{th} node ($\mathcal{I}_{n,m}$) can be obtained as shown in Equation 8.

$$\mathcal{I}_{n,m} = \sum_{j=1}^{F} w_{mj}s_{nj} \tag{8}$$

(ii) **Node importance.** Since the features of a node are representative of the node itself, the importance of a node can be obtained using the importance scores of its features. The importance of an n^{th} node ($\mathcal{I}_n$) can be obtained as shown in Equation 9.

$$\mathcal{I}_n = \sum_{j=1}^{F} \parallel \mathbf{w}_j \parallel_2 s_{nj} \tag{9}$$

Here, $\mathbf{w}_j = [w_{1j}, w_{2j}, ..., w_{dj}]^T$. We normalize the importance scores over different features of a node (Equation 8) and over different nodes (Equation 9) to be in the range $[0, 1]$.

Visual interpretation. Fig. 1 (top) shows the colormap visualization obtained by randomly choosing 64 features across 16 randomly selected nodes from the first hidden layer of a GCN with feature gating. It indicates the node feature importance. Further, in t-Distributed Stochastic Neighbor Embedding (t-SNE) plot [34] (Fig. 1 (bottom)) the nodes are represented as circles of different radius (node importance) and different classes as clusters of a different color.

4. RESULTS

This section evaluates the efficacy of the proposed framework against the state-of-the-art GNN models for semi-supervised node

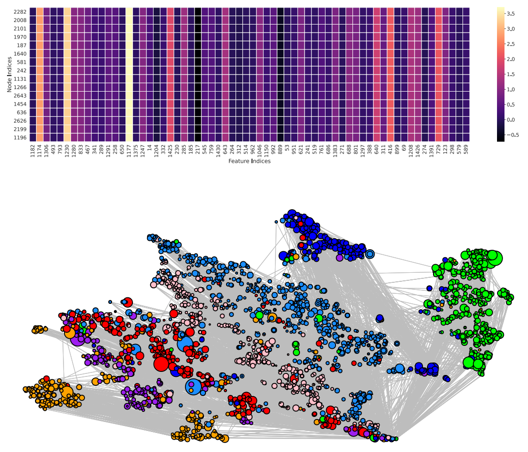

Fig. 1. Colormap (top) and t-SNE visualization (bottom) on the Cora dataset.

classification tasks. We evaluate the proposed framework over Cora, Citeseer, PubMed [35], Coauthor CS, Coauthor Physics [36], Amazon Photo, and Amazon Computer [37] datasets. We split the seven datasets into training, validation, and testing sets by randomly selecting 20 nodes per class for training, 500 nodes for validation, and 1,000 nodes for testing. The learning rate of 0.001, $\alpha_l = 0.2$ (Cora and Pubmed) and 0.5 (others), and a dropout of 0.5 is used training. For all the methods (except ours) the hyper parameters values are as mentioned in the respective papers.

Dataset	# Classes	# Nodes	# Edges	# Features	Average Degree
Cora	7	2,708	5,429	1,433	3.90
Citeseer	6	3,327	4,732	3,703	2.77
PubMed	3	19,717	44,338	500	4.50
Coauthor CS	15	18,333	81,894	6,805	8.93
Coauthor Physics	5	34,493	2,47,962	8,415	14.38
Amazon Photo	8	7,650	1,19,081	745	31.13
Amazon Computer	10	13,752	2,45,861	767	35.76

Table 1. Dataset Statistics

Table 2 shows that feature gating enhances the performance over vanilla and DropEdge variants of GCN and GAT. However, the increase in performance for GCN is higher than that of GAT. This is primarily attributed to the fact that the attention mechanism in GAT is able to perform appropriate feature selection (though in a relatively complex manner), which otherwise would have been handled by feature gating. Overall, the proposed frameworks (GCN-IR and GCN-IR-FG) report the best test accuracy across different datasets.

Further, we show the performance variation with network depth when evaluated over Cora, Citeseer, and PubMed datasets in Table 3. Although GCN-FG and GAT-FG show better performance when compared with GCN and GAT alone, they lag in handling oversmoothing when compared to their respective DropEdge variants because the accuracy continues to drop with increasing depth. This indicates that feature gating alone might not be sufficient (though better than their vanilla version) to address oversmoothing. However, as depicted by the performance of GFGN, it can potentially slow down the rate of convergence. The performance of GRAND peaks at eight layers and either saturates or drops with a further increase in the number of layers. Interestingly, GCN-IR and GCN-IR-FG exhibit an increase in accuracy with the network depth and hence, address oversmoothing across different datasets. While GCNII also shows similar trends, the proposed formulation reports higher accuracy until 64 layers. Hence, the combination of identity mapping and modified residual along with feature gating can address oversmoothing to foster deeper GCNs.

Method	Cora	Citeseer	PubMed	Coauthor CS	Coauthor Physics	Amazon Photos	Amazon Computers
SGC	80.7 ± 0.8	72.8 ± 0.6	77.3 ± 1.1	78.72 ± 0.7	76.28 ± 1.2	75.14 ± 0.5	79.85 ± 0.7
GCN	81.1 ± 0.7 (2)	70.9 ± 0.9 (2)	78.1 ± 0.5 (2)	81.87 ± 0.8 (2)	83.99 ± 0.7 (2)	81.63 ± 0.7 (2)	84.67 ± 0.8 (2)
GCN-DE	82.3 ± 0.6 (2)	71.1 ± 0.5 (2)	78.4 ± 0.3 (2)	82.28 ± 0.6 (2)	85.08 ± 0.6 (2)	82.07 ± 0.8 (2)	84.91 ± 0.6 (2)
GCN-FG	82.8 ± 0.6 (2)	73.1 ± 0.6 (2)	79.1 ± 0.5 (2)	83.05 ± 0.6 (2)	85.56 ± 0.5 (2)	83.86 ± 0.5 (2)	85.29 ± 0.5 (2)
GAT	81.9 ± 0.5 (2)	70.6 ± 0.6 (2)	77.4 ± 0.9 (2)	82.01 ± 0.5 (2)	84.57 ± 1.1 (2)	82.53 ± 0.9 (2)	85.09 ± 0.7 (2)
GAT-DE	82.1 ± 0.6 (2)	70.8 ± 0.4 (2)	77.5 ± 0.8 (2)	82.84 ± 0.7 (2)	85.22 ± 0.6 (2)	83.09 ± 0.7 (2)	85.57 ± 0.5 (2)
GAT-FG	82.0 ± 0.5 (2)	70.9 ± 0.7 (2)	77.8 ± 0.9 (2)	82.56 ± 0.4 (2)	84.98 ± 0.5 (2)	82.84 ± 0.6 (2)	84.11 ± 0.6 (2)
JKNet	81.1 ± 0.8 (4)	69.8 ± 0.6 (16)	78.1 ± 0.5 (16)	84.54 ± 0.7 (8)	85.52 ± 0.5 (16)	81.67 ± 0.6 (16)	82.14 ± 0.6 (16)
APPNP	83.3 ± 0.7	71.8 ± 0.8	79.8 ± 0.7	85.37 ± 0.5	85.98 ± 0.6	82.62 ± 0.4	83.81 ± 0.8
GRAND*	84.3 ± 0.6 (8)	72.8 ± 0.5 (8)	78.5 ± 0.6 (4)	88.31 ± 0.5 (8)	87.89 ± 0.6 (8)	85.31 ± 0.5 (8)	86.66 ± 0.7 (4)
GCNII	84.8 ± 0.5 (32)	72.9 ± 0.3 (64)	79.8 ± 0.3 (16)	88.83 ± 0.8 (16)	88.51 ± 0.7 (32)	88.27 ± 0.6 (32)	87.25 ± 0.6 (32)
GFGN*	84.9 ± 0.6 (4)	73.4 ± 0.4 (4)	80.4 ± 0.4 (8)	89.03 ± 0.6 (4)	89.45 ± 0.3 (4)	89.13 ± 0.8 (8)	87.92 ± 0.5 (4)
GCN-IR	85.3 ± 0.7 (32)	73.2 ± 0.6 (32)	**80.1 ± 0.5** (32)	89.98 ± 0.7 (32)	89.75 ± 0.6 (32)	89.16 ± 0.7 (64)	**88.91 ± 0.6** (32)
GCN-IR-FG	**85.7 ± 0.5** (32)	**73.6 ± 0.4** (32)	80.0 ± 0.6 (64)	**90.79 ± 0.6** (64)	**90.21 ± 0.4** (32)	**89.94 ± 0.4** (64)	88.49 ± 0.5 (32)

Table 2. Test accuracy and standard deviation over 50 random initializations. (DE: DropEdge, FG: Feature Gating) *For each dataset, we present the best performing variants of GRAND and GFGN.

Dataset	Method	Layers					
		2	4	8	16	32	64
Cora	GCN	**81.1**	80.4	69.5	64.9	60.3	28.7
	GCN-DE	**82.3**	82.0	75.8	75.7	62.5	49.5
	GCN-FG	**82.8**	78.6	69.9	72.7	59.3	47.4
	GAT	**81.9**	79.8	69.5	47.8	45.1	25.3
	GAT-DE	**82.1**	80.8	72.9	66.3	51.3	43.4
	GAT-FG	**82.0**	68.4	51.3	59.7	45.1	29.2
	GRAND	72.8	79.6	**84.3**	80.1	77.2	73.4
	GCNII	82.1	82.4	83.6	83.9	**84.8**	84.6
	GFGN	83.7	**84.9**	82.3	79.6	65.2	61.9
	GCN-IR	82.4	83.9	84.6	84.8	**85.3**	85.2
	GCN-IR-FG	83.2	82.9	85.1	85.4	**85.7**	85.5
Citeseer	GCN	**70.9**	67.6	30.2	18.3	25.0	20.0
	GCN-DE	**71.1**	69.6	61.4	57.2	41.6	34.4
	GCN-FG	**73.1**	61.5	57.2	55.5	39.7	29.1
	GAT	**70.6**	61.5	38.4	25.5	17.3	20.3
	GAT-DE	**70.8**	63.7	46.6	31.6	22.2	21.5
	GAT-FG	**70.9**	60.8	41.3	27.6	11.8	18.7
	GRAND	68.8	71.6	**72.8**	72.6	70.1	69.5
	GCNII	68.3	68.6	69.1	70.2	**72.9**	**72.9**
	GFGN	71.8	**73.4**	72.7	68.9	61.4	58.7
	GCN-IR	66.1	67.9	70.6	72.0	**73.2**	73.1
	GCN-IR-FG	68.1	68.7	71.2	71.7	**73.6**	72.8
PubMed	GCN	**78.1**	75.5	62.2	41.7	21.4	34.2
	GCN-DE	**78.4**	77.9	78.1	78.2	77.0	61.5
	GCN-FG	**79.1**	74.7	65.1	54.5	41.7	35.1
	GAT	**77.4**	69.4	48.3	39.5	41.3	40.7
	GAT-DE	**77.5**	76.8	76.4	67.1	54.7	68.2
	GAT-FG	**77.8**	68.7	51.8	48.2	49.1	41.6
	GRAND	72.4	74.7	**78.5**	76.2	71.9	68.6
	GCNII	74.9	76.2	77.5	**79.8**	78.1	74.2
	GFGN	**80.4**	79.7	77.6	72.4	67.6	62.8
	GCN-IR	78.1	78.3	79.1	79.6	**80.1**	79.5
	GCN-IR-FG	77.8	78.1	79.7	79.9	**80.0**	**80.0**

Table 3. Performance variation (%) with increasing depth until 64 layers owing to memory constraints.

5. CONCLUSION

In this work, we establish the efficacy of feature gating in graph convolution and explore its applicability in designing deeper graph convolutional networks. We propose the use of identity mapping and a modified form of residual connection along with feature gating to create deep GCN models that achieve the best results for the semi-supervised node classification task. We establish the idea of feature selection facilitated through feature gating by assigning importance scores to nodes and node features. We describe how to interpret these importance scores to understand their effect on node classification through necessary visualizations. Some interesting directions in the future would be to understand the effect of feature gating on even larger graphs, their use as an attention mechanism, and provide better interpretability and explainability of GCNs.

6. REFERENCES

[1] Thomas N Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[2] William L Hamilton, Rex Ying, and Jure Leskovec, "Inductive representation learning on large graphs," *arXiv preprint arXiv:1706.02216*, 2017.

[3] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[4] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger, "Simplifying graph convolutional networks," in *International conference on machine learning*. PMLR, 2019, pp. 6861–6871.

[5] Meng Qu, Yoshua Bengio, and Jian Tang, "Gmnn: Graph markov neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 5241–5250.

[6] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang, "Deepinf: Social influence prediction with deep learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2110–2119.

[7] Chang Li and Dan Goldwasser, "Encoding social information with graph convolutional networks forpolitical perspective detection in news media," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 2594–2604.

[8] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, vol. 33, pp. 922–929.

[9] Jia Li, Zhichao Han, Hong Cheng, Jiao Su, Pengyun Wang, Jianfeng Zhang, and Lujia Pan, "Predicting path failure in time-evolving graphs," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1279–1289.

[10] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 974–983.

[11] Long Zhao, Xi Peng, Yu Tian, Mubbasir Kapadia, and Dimitris N Metaxas, "Semantic graph convolutional networks for 3d human pose regression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3425–3435.

[12] Lingxiao Zhao and Leman Akoglu, "Pairnorm: Tackling oversmoothing in gnns," *arXiv preprint arXiv:1909.12223*, 2019.

[13] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang, "Dropedge: Towards deep graph convolutional networks on node classification," *arXiv preprint arXiv:1907.10903*, 2019.

[14] Qimai Li, Zhichao Han, and Xiao-Ming Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018, vol. 32.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[16] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem, "Deepgcns: Can gcns go as deep as cnns?," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9267–9276.

[17] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li, "Simple and deep graph convolutional networks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 1725–1735.

[18] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang, "Free-form image inpainting with gated convolution," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4471–4480.

[19] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka, "Representation learning on graphs with jumping knowledge networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5453–5462.

[20] Benjamin Paul Chamberlain, James Rowbottom, Maria Gorinova, Stefan Webb, Emanuele Rossi, and Michael M Bronstein, "Grand: Graph neural diffusion," *arXiv preprint arXiv:2106.10934*, 2021.

[21] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," *arXiv preprint arXiv:1810.05997*, 2018.

[22] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann, "Diffusion improves graph learning," *arXiv preprint arXiv:1911.05485*, 2019.

[23] Wei Jin, Xiaorui Liu, Yao Ma, Tyler Derr, Charu Aggarwal, and Jiliang Tang, "Graph feature gating networks," *arXiv preprint arXiv:2105.04493*, 2021.

[24] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[25] Kenta Oono and Taiji Suzuki, "Graph neural networks exponentially lose expressive power for node classification," *arXiv preprint arXiv:1905.10947*, 2019.

[26] Moritz Hardt and Tengyu Ma, "Identity matters in deep learning," *arXiv preprint arXiv:1611.04231*, 2016.

[27] A Van den Oord, S Dieleman, H Zen, K Simonyan, O Vinyals, A Graves, N Kalchbrenner, A Senior, and K Kavukcuoglu, "Wavenet: A generative model for raw audio, arxiv," *arXiv preprint arXiv:1609.03499*, 2016.

[28] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier, "Language modeling with gated convolutional networks," in *International conference on machine learning*. PMLR, 2017, pp. 933–941.

[29] Jie Hu, Li Shen, and Gang Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[30] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu, "Conditional image generation with pixelcnn decoders," *arXiv preprint arXiv:1606.05328*, 2016.

[31] Hongzhen Wang, Ying Wang, Qian Zhang, Shiming Xiang, and Chunhong Pan, "Gated convolutional neural network for semantic segmentation in high-resolution images," *Remote Sensing*, vol. 9, no. 5, pp. 446, 2017.

[32] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[33] Luana Ruiz, Fernando Gama, and Alejandro Ribeiro, "Gated graph recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 68, pp. 6303–6318, 2020.

[34] Laurens Van der Maaten and Geoffrey Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[35] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[36] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel, "Image-based recommendations on styles and substitutes," in *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 2015, pp. 43–52.

[37] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia, "Microsoft academic graph: When experts are not enough," *Quantitative Science Studies*, vol. 1, no. 1, pp. 396–413, 2020.