On the Global Self-Attention Mechanism for Graph Convolutional Networks

Chen Wang
Department of Computer Science
Rutgers University
Piscataway, New Jersey 08854
Email: chen.wang.cs@rutgers.edu

Chengyuan Deng
Department of Computer Science
Rutgers University
Piscataway, New Jersey 08854
Email: charles.deng@rutgers.edu

***Abstract*—Applying Global Self-Attention (GSA) mechanism over features has achieved remarkable success on Convolutional Neural Networks (CNNs). However, it is not clear if Graph Convolutional Networks (GCNs) can similarly benefit from such a technique. In this paper, inspired by the similarity between CNNs and GCNs, we study the impact of the Global Self-Attention mechanism on GCNs. We find that consistent with the intuition, the GSA mechanism allows GCNs to capture feature-based vertex relations regardless of edge connections; As a result, the GSA mechanism can introduce extra expressive power to the GCNs. Furthermore, we analyze the impacts of the GSA mechanism on the issues of overfitting and over-smoothing. We prove that the GSA mechanism can alleviate both the overfitting and the over-smoothing issues based on some recent technical developments. Experiments on multiple benchmark datasets illustrate both superior expressive power and less significant overfitting and over-smoothing problems for the GSA-augmented GCNs, which corroborate the intuitions and the theoretical results.**

I. Introduction

The emerge of Graph Convolutional Network (GCN) framework [1] has prompted graph networks to be one of the most promising techniques in pursuing artificial general intelligence [2]. Inspired by the closely related field of Concolutional Neural Networks (CNNs), different attention and self-attention mechanisms have been proposed to improve the quality of information aggregation under the GCN framework (e.g. [3]). Existing self-attention mechanisms in GCNs usually consider the feature information between neighboring vertices, and assign connection weights to each vertex accordingly [3], [4]. This type of attention considers the local geometry as the edge connections of the graph, and exclude possible scenarios when a vertex could have strong correlations and influences with another without edge connection. To date, as we can see from a comprehensive survey [5], there has now been any significant work considering applying the Global Self-Attention (GSA) mechanism to GCNs.

We notice that despite the absence of the study on the GSA mechanism on the GCNs, such mechanism has achieved remarkable success on the similar Convolutional Neural Networks (notably in [6]). Therefore, in this paper, inspired by the above observations, we study the effect of GSA mechanism on the GCN domain. The similarity between CNNs and GCNs ([5]) makes the implementation of the GSA mechanism on GCNs straightforward. For CNNs operating on image, the GSA mechanism functions in the way to compute the inner products between pixel features. Likewise, for GCNs operating on graphs, we can view each vertex roughly as a pixel, and the local edge-based geometry as an analogy of convolutional kernels; Therefore, the GSA mechanism on GCNs can be achieved by direct product between every pairs of nodes, regardless of edge connection. An intuitive illustration of the above process can be found in Figure 1.

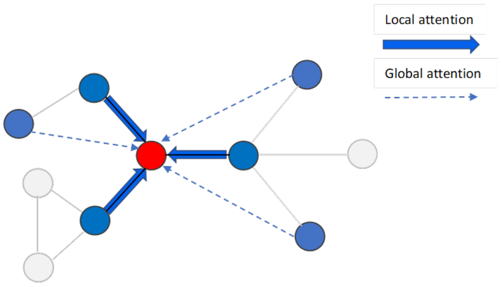

Fig. 1: Local & Global Attention Mechanism in Graph Neural Networks. Local attention is only applied to neighbor nodes of the target node, while global attention includes every node in the graph

Given the celebrated results for the GSA mechanism on CNNs [6], it is reasonable to believe the GSA mechanism can give a performance boost for the GCN model. Similar to the intuition behind the GSA-augmented CNN, the GSA mechanism will give GCN the ability to capture long-range vertex dependencies and feature convolutions. This capacity can give the GCNs additional expressive power, especially when there exist pairs of vertices which are not connected by edges yet share similar features and are of the same class/pattern. We examine this intuition through the experiments on multiple benchmark datasets for node and graph classification tasks, and the empirical results affirm that by simply applying the GSA mechanism to plain GCN, it can outperform multiple carefully-designed advanced methods.

In addition to the above intuition, we study the impacts for the GSA mechanism on overfitting and over-smoothing, two issue closely connected to the graph edge structure. overfitting means the scenario when the testing accuracy decreases while the training accuracy still appears to climb up. Over-smoothing indicates the situation of exponentially-growing training losses as more layers are stacked. We prove that the GSA mechanism can mitigate both issues: On the overfitting problem, we decompose the loss of the GCN with the GSA mechanism into two regularization terms, and we prove that it is possible for the GSA mechanism to simulate the 'edge dropout' process ([7]) to mitigate overfitting; On the front of over-smoothing, we prove that the GSA mechanism is roughly equivalent to interpolating a positive definite matrix to the original feature, and it can therefore reduce over-smoothing according to a recently-established theoretical framework ([8]). These theoretical results are verified by the experiments.

The rest of the paper is arranged as follows. Section II provides a brief review of related work and the context of this paper; Section III outlines the details of applying the GSA mechanism to GCNs; The analysis on overfitting and over-smoothing is presented in Section IV, and experimental results are show in Section V; And finally, section Section VI presents a general conclusion of our work.

II. Related Work

We focus on the Graph Convolutional Networks based on Message-passing, which was initiated by the idea of approximating graph spectral convolution [9], [10], [11] and popularized by the success of the standard GCN model [1]. To date, there have been fruitful outcomes on novel graph networks stemmed from the strategy [12], [13], [14]. The mechanism of aggregating information from neighboring nodes can be deemed as a spatial-based approach [15] which is similar to the procedures in Convolutional Neural Networks (CNNs) [5]. Consequently, the attention techniques in CNNs [16] have been widely examined from the perspective of GCNs [3], [17], [18], [19].

Despite the similarity between the two models, the attention methods in GCNs are mostly applied to local geometry with neighborhood connections, while CNN-based attention techniques are often applied to the global features. For instance, [6] provides a self-attention CNN as the generator of Generative Adversarial Networks (GANs) and shows celebrated capabilities in capturing long-range feature relations. The remarkable success indicates potential advance by applying the global attention mechanism to GCNs to get similar results. Moreover, [20] outlines one significant weakness of Message passing-based Graph Networks is the lack of 'the ability to capture long-range dependencies'. Consequently, it can be reasonably assumed that introducing global attention mechanism to GCNs will provide positive outcomes.

From a more theoretical perspective, the association between the GSA mechanism and the over-smoothing issue is significant. It has been long noticed that GCNs cannot be stacked as deep as CNNs without invoking negative effects ([21], [22]). [23] provides an insight of this by showing the linear GCN is a special form of Laplacian smoothing and will converge to an feature-invariant point as the network goes deeper. Sub-sequentially, [8] proves GCNs with Relu activation will converge to a feature-invariant space with a rate exponential to the maximum singular value of the convolutional filter. Our theorem for the GSA mechanism on over-smoothing is inspired by the above-mentioned work. The link between the GSA mechanism and overfitting is not as significant, but we notice that the GSA mechanism can simulate different methods ([12], [7]) that are proved to alleviate overfitting. Therefore, we established our theoretical result on the GSA and overfitting based on this idea.

III. Implementing Global Self-Attention Mechanism on GCNs

A. Graph Convolution Networks

Given a graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, a Graph Convolutional Networks takes as input a set of features $\boldsymbol{X}_i$ for every node (vertex) $v_i \in \boldsymbol{V}$, resulting in a node-feature matrix representation $\boldsymbol{X}$ of a graph $n \times d$ where n is number of nodes/vertices and d is number of features. We use the graph adjacency matrix $\boldsymbol{A}$ model with the size of $n \times n$ to represent the edge connections. The network layers are connected using a convolutional projection of input graph $\boldsymbol{X}$ with adjacency matrix $\boldsymbol{A}$. In practice, we add self-loop on each node to get $\hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ with $\boldsymbol{I}$ as the identity matrix. Therefore, the feed-forward layer of a GCN can be expressed as

$$\begin{aligned} \boldsymbol{H}^{(l+1)} &= f_{\boldsymbol{W}}(\boldsymbol{H}^{(l)}, \boldsymbol{A}) \\ &= \sigma(\boldsymbol{D}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}) \\ &= \sigma(\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}), \end{aligned} \tag{1}$$

where $\boldsymbol{W}^{(l)}$ is the weight matrix for convolution of the l-th layer and $\sigma(\cdot)$ is the activation function. Matrix $\boldsymbol{D} = \texttt{diag}\left(\sum_{j=1}^{N} \hat{\boldsymbol{A}}_{\cdot,j}\right)$ is the normalization matrix of $\hat{A}$, where $\texttt{diag}(\cdot)$ means the operation of expanding an n-length vector to a $n \times n$ matrix with the main diagonal filled by the elements of the vector.

B. Global Self-attention Mechanism

The Global Self-Attention mechanism on images with Convolutional Neural Networks (CNNs) is discussed thoroughly in [6]. Transferring to the GCNs, on each layer, the self-attention layer takes the output of the previous layer $\boldsymbol{H}^{(l)}$ and calculate the influence of node j on node i with

$$\begin{gathered} \beta_{i,j} = \texttt{softmax}_{j \in \{1,2,\cdots,n\}}[s_{i,j}] \\ \text{where } s_{i,j} = (\hat{\boldsymbol{H}}_i^{(l)}\boldsymbol{W}_l)(\hat{\boldsymbol{H}}_j^{(l)}\boldsymbol{W}_r)^T, \end{gathered} \tag{2}$$

where the calculation of $s_{i,j}$ is essentially a pair-wise production by summing over all the channels/features of the node. Matrices $\boldsymbol{W}_l$ and $\boldsymbol{W}_r$ serve the purpose of dimension-reduction to reduce the computational load and to provide additional flexibility to the trainable variables. We denote the result of the attention importance map as matrix $\boldsymbol{B}$, which is a $n \times n$ matrix. With the attention importance mask, the attention feature can be calculated with

$$\boldsymbol{o}_i^{(l)} = (\sum_{j=1}^{N} \beta_{i,j}\boldsymbol{H}_j^{(l)}\boldsymbol{W}_h)\boldsymbol{W}_g, \tag{3}$$

where $\boldsymbol{W}_h$ is the matrix to transform the input $\boldsymbol{H}^{(l)}$ to a lower dimension and $\boldsymbol{W}_h$ is the matrix to project the feature size back to the original. The operations of the above equation can be efficiently paralleled by re-writing it in the matrix production formula.

C. GSA Mechanism Augmented GCNs

The GSA mechanism correctly captures the feature information on a global level. Nevertheless, for graph inputs and networks, local geometry denoted by edge connections is also crucial and it is the very characteristic that makes a graph network. Thus, we perform an interpolation similar first introduce by [6]. The resulting Global Self-attention GCN layer goes as

$$\boldsymbol{H}^{(l+1)} = \sigma((\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l)} + \gamma\boldsymbol{O}^{(l)})\boldsymbol{W}^{(l)}), \tag{4}$$

where $\boldsymbol{O}^{(l)}$ is the output of the global self-attention and γ is a non-negative trainable parameter ($\gamma > 0$) with initial value 0 and $\boldsymbol{W}^{(l)}$ is the convolution/filter matrix of layer l. Notice that the attention feature will *not* be processed by graph adjacency-based aggregation in the above operation. The necessity of $\boldsymbol{W}^{(l)}$ is questioned in [24] and one can drop this matrix if the computational resource is limited. Nevertheless, we keep the matrix here to serve a general purpose of applications and analysis.

IV. Theoretical Analysis for the GSA mechanism on overfitting and Over-smoothing

In this section, we discuss the impact of GSA mechanism on the issues of overfitting and over-smoothing in Graph Convolutional Networks (GCNs). We argue that the GSA mechanism can alleviate overfitting and over-smoothing simultaneously. Specifically, we provide the intuition for the GSA mechanism to prevent GCNs from overfitting the local edge geometry in Section IV-A, and show that the GSA mechanism could affect the GCN in the same way of DropEdge [7] under certain assumptions and model variations. Furthermore, we prove that with mild approximations and assumptions, the GSA mechanism is guaranteed to mitigate over-smoothing in section IV-B.

In the interest of the analysis, we re-formulate a simplified version of the layer in Eq (4). Notice that the linear attention transformation matrices $\boldsymbol{W}_l, \boldsymbol{W}_r, \boldsymbol{W}_h, \boldsymbol{W}_g$ are employed mainly for the purpose of reducing computational complexity and accelerating training. Thus, we simplify them to the identity matrix $\boldsymbol{I}$ in this section.

A. overfitting

We first discuss the impact for GSA on remedy the overfitting problem. We consider the overfitting noise from the following source: In Graph Convolutional Network, we assume the vertices connected by an edge share the similar information, and the neighbors of a vertex should contain homogeneous features. However, in reality, this is not necessarily true. Therefore, if a vertex learn from an neighboring vertex with irrelevant or mixed information, it cause an overfitting problem to the local edge geometry.

Intuitively, the Global Self-Attention mechanism can alleviate this issue in two ways. Firstly, from a geometry perspective, the GSA mechanism can aggregate information from 'faraway' vertices regardless of the edge connections, and its effect can be viewed as a regularization to the graph local geometries; And secondly, from a feature perspective, the GSA mechanism can force the vertices with similar features to share information, which mitigates the noise from neighboring vertices with feature of different patterns.

The above intuition can be mathematically grounded as follows. Denoting the output of the last hidden layer as $\boldsymbol{H}^{(L)} = \boldsymbol{H}^{(L-1)}\boldsymbol{W}$ (without considering the activation function), and $\boldsymbol{H}^{(L-1)}$ is the hidden features before the last layer. The loss function can be denoted as:

$$\mathcal{L}(\tilde{\boldsymbol{A}}\boldsymbol{H}^{(L)} + \gamma\boldsymbol{B}\boldsymbol{H}^{(L)}, \boldsymbol{y}), \tag{5}$$

where $\boldsymbol{y}$ is the target (the labels of nodes or graph). Let $\bar{\boldsymbol{A}}$ be the complement of the unnormalized $\boldsymbol{A}$, which means $\bar{\boldsymbol{A}}_{i,j} = 1 \leftrightarrow \boldsymbol{A}_{i,j} = 0$ (i.e. the adjacency matrix of 'no connections'). Also, denote $\boldsymbol{J}$ as the 'all ones' matrix and $\boldsymbol{J} = \boldsymbol{I} + \boldsymbol{L}$, where $\boldsymbol{L}$ is the 'all-ones except the main diagonal' matrix. The first argument (denoting as $\hat{\boldsymbol{y}}$) of the loss function of Eq (5) can be decomposed as the function of $\bar{\boldsymbol{A}}$:

$$\begin{aligned}
\hat{\boldsymbol{y}} &= \tilde{\boldsymbol{A}}\boldsymbol{H}^{(L)} + \gamma\bar{\boldsymbol{A}} \circ \boldsymbol{B}\boldsymbol{H}^{(L)} + \gamma(\boldsymbol{J} - \bar{\boldsymbol{A}}) \circ \boldsymbol{B}\boldsymbol{H}^{(L)} \\
&= \tilde{\boldsymbol{A}}\boldsymbol{H}^{(L)} + \gamma\bar{\boldsymbol{A}} \circ \boldsymbol{B}\boldsymbol{H}^{(L)} + \gamma(\boldsymbol{I} + \boldsymbol{L} - \bar{\boldsymbol{A}}) \circ \boldsymbol{B}\boldsymbol{H}^{(L)} \\
&= \tilde{\boldsymbol{A}}\boldsymbol{H}^{(L)} + \gamma(\bar{\boldsymbol{A}} + \boldsymbol{L}) \circ \boldsymbol{B}\boldsymbol{H}^{(L)} + \gamma(\boldsymbol{I} - \bar{\boldsymbol{A}}) \circ \boldsymbol{B}\boldsymbol{H}^{(L)} \\
&= \big(\tilde{\boldsymbol{A}} + \gamma(\bar{\boldsymbol{A}} + \boldsymbol{L}) \circ \boldsymbol{B}\big)\boldsymbol{H}^{(L)} + \gamma(\boldsymbol{I} - \bar{\boldsymbol{A}}) \circ \boldsymbol{B}\boldsymbol{H}^{(L)},
\end{aligned}$$

where $\circ$ represents the element-wise multiplication. In the above equation, let $\boldsymbol{A}' = \tilde{\boldsymbol{A}} + \gamma(\bar{\boldsymbol{A}} + \boldsymbol{L}) \circ \boldsymbol{B}$, and this represents the *geometric penalization* of the edge connections to avoid overfitting. Furthermore, for any super-additive loss function with property $\mathcal{L}(x+y, z) \gtrsim \mathcal{L}(x, z) + \mathcal{L}(y, 0)$, the loss function in Eq (5) can be decomposed into

$$\mathcal{L}\big((\tilde{\boldsymbol{A}} + \gamma(\bar{\boldsymbol{A}} + \boldsymbol{L}) \circ \boldsymbol{B})\boldsymbol{H}^{(L)}, \boldsymbol{y}\big) + \mathcal{L}\big(\gamma(\boldsymbol{I} - \bar{\boldsymbol{A}}) \circ \boldsymbol{B}\boldsymbol{H}^{(L)}, \boldsymbol{0}\big).$$

For the linear solution of the second term (i.e. $\gamma(\boldsymbol{I} - \bar{\boldsymbol{A}}) \circ \boldsymbol{B}\boldsymbol{H}^{(L)} = \boldsymbol{0}$), it is equivalent to the minimization of the quadratic form:

$$\frac{\gamma}{2} \cdot (\boldsymbol{H}^{(L)})^T(\boldsymbol{I} - \bar{\boldsymbol{A}}) \circ \boldsymbol{B}(\boldsymbol{H}^{(L)}).$$

This term is proportional to

$$\mathcal{L}_{\text{reg}} = \frac{\gamma}{2} \cdot \sum_{i,j} \mathbb{I}\big(\boldsymbol{A}_{i,j} = 0\big) \cdot \langle \boldsymbol{h}_i^{(L-1)}, \boldsymbol{h}_i^{(L-1)} \rangle \cdot \|\boldsymbol{h}_i^{(L)} - \boldsymbol{h}_j^{(L)}\|_2^2,$$

which means the GSA mechanism will penalize the situation where disconnected vertices v_i and v_j share similar features in the $(L-1)$ layer but are processed to considerably different features. Putting the above together, by minimizing the target in Eq (5) with vanilla gradient descent, the GSA mechanism will encourage the upper bound of the following target:

$$\mathcal{L}\big(\underbrace{(\tilde{\boldsymbol{A}} + \gamma(\bar{\boldsymbol{A}} + \boldsymbol{L}) \circ \boldsymbol{B})}_{\text{geometry regularization}}\boldsymbol{H}^{(L)}, \boldsymbol{y}\big) \quad + \quad \underbrace{\mathcal{L}_{\text{reg}}}_{\text{feature regularization}} \quad , \tag{6}$$

which supports the intuition for the GSA mechanism to alleviate overfitting.

Beyond the above insights, one can also understand the impact of the Global Self-Attention mechanism on GCNs from the perspective of vertex subsampling [12] and DropEdge [7]. These methods adopt techniques similar to Dropout, and they have been shown to alleviate overfitting. The DropEdge method follows a simple strategy to delete edges during convolution; and to see how the GSA mechanism can simulate this method, we provide an analysis that, under certain assumptions and simplifications, the impact for the GSA mechanism on a significant portion of the vertices is essentially to cancel the impact of another vertex (thus 'drop an edge').

To begin with, let us first define the feature *relations* and feature *influence* for the analysis:

Definition IV.1. We define the *feature influence* from v_i to v_j as the value (or vector) that v_i can change on the feature of v_j without normalization; Similarly, we define the *feature relations* from v_i to v_j as the weight ('extent') of the influence from v_i to v_j (analogy to 'attention weights').

The exact process of the GSA mechanism is difficult to analyze since every vertex can affect each other (too 'chaotic' to get anything more than trivial). Therefore, we make some assumptions and simplifications:

Assumption 1. We study d-regular graphs with following properties:

1) The feature *influence* between two vertices are either $+1$ (*positive*) or -1 (*negative*). That is to say, on the graph convolution operation, the influence of each vertex to another is $\frac{1}{d}$ or $-\frac{1}{d}$.
2) For any vertex v, the feature influence from its neighbors $\mathcal{N}(v)$ are not homogeneous (there exist both *positive* and *negative*).
3) For any vertex v, when computing the Global Self-Attention, only r ($d < r < n$) other vertices will have non-zero *feature relations* towards v (only the influence of r vertices will account).

Remark 1. We can see that the above simplifications and assumptions are realistic in applications. The first assumption quantifies the influence on vertices to scalar and restricted them to $+1$ and -1; And usually, the influence on vertices are indeed 'augment' or 'weaken' the feature pattern. The second assumption is only to make sure the source of overfitting exists. The final assumption seems strong; Nevertheless, since the `softmax()` in Eq (2) will suppress the influence of a majority of vertices, the non-zero relations of the GSA mechanism on most vertices should be indeed a small fraction.

We present the following theorem for the graph with the aforementioned assumptions:

Theorem IV.1. *Let the interpolation parameter γ in Eq (4) be $\gamma = \frac{1}{d}$. There exists a way to arrange the feature relations of the vertices, such that for at least $C \cdot (r+1)^{\frac{3}{2}} \cdot \frac{n}{4^r}$ (C is a constant) vertices, the Global Self-Attention mechanism will eliminate the influence of one of its neighboring vertices.*

Proof. This is a natural result following the well-renowned Ramsey theorem. Recall that for every pair of vertices, the feature influence is either $+1$ or -1. According to Ramsey theorem, for every $R(r+1, r+1)$ vertices, there should be $(r+1)$ vertices with feature influences all positive $(+1)$ or negative (-1). Therefore, we can arrange the feature relations in a way that for any of these $(r+1)$ vertices, the *rest of them* will be the r vertices with non-zero feature relations in the Global Self-Attention process.

Define the set of these $(r+1)$ vertices as $\boldsymbol{V}'$, and focus on any of $v' \in \boldsymbol{V}'$. Since the feature influences of vertices in $\boldsymbol{V}'/\{v'\}$ are entirely positive or negative, the cumulative influence to v' from the Global Self-Attention mechanism is either $+1$ or -1 (with the normalization). With the interpolation parameter $\gamma = \frac{1}{d}$, the overall impact from the the Global Self-Attention mechanism to v' is either $+\frac{1}{d}$ or $-\frac{1}{d}$. Furthermore, since the feature influences from $\mathcal{N}(v')$ (following the edge geometry) are *not* entirely positive or negative, there must be one vertex in $\mathcal{N}(v')$ with its the feature influence eliminated.

For all the vertices in $\boldsymbol{V}'$, we have at least $(r+1)$ vertices undergoing the 'DropEdge' procedure. Now, suppose in the worst case, the $\boldsymbol{V}'$ sets with mutually positive or negative influences vertices do not overlap; in this case, we can find at least $(r+1)$ qualified vertices for every $R(r+1, r+1)$ vertices. A well-establish upper bound for the $R(s,s)$ Ramsey number (see [25]) is $[1+o(1)]\frac{4^{s-1}}{\sqrt{\pi s}}$. Plugging the number of $(r+1)$, this will give us $\frac{n}{4^r} \cdot \sqrt{\pi}[1+o(1)] \cdot (r+1)^{\frac{1}{2}}$ sets of $\boldsymbol{V}'$ vertices. Finally, since we have at least $(r+1)$ vertices in each of these sets with the 'edge elimination' process, the overall number of vertices to undergo the process is at least $\pi[1+o(1)] \cdot (r+1)^{\frac{3}{2}} \cdot \frac{n}{4^r}$. □

Remark 2. We remark that when the third assumption does not hold, we can modify the Global Self-Attention mechanism to a 'refined' version that sub-samples r vertices to compute feature influences. In this way, it is possible to design an algorithm that with *any* arrangement of the relations between features, a large portion of vertices will undergo the edge dropout process with high (constant) probability. However, as we focus on understanding the GSA mechanism in this paper, we leave the above idea as a future direction to pursue.

B. Over-smoothing

Our result on over-smoothing is based on the theory in [8], which analyzes the over-smoothing problem as the convergence distance to an invariant subspace. Under the ReLu activation, the distance between the layer output and the space is upper-bounded by $O((s\lambda)^L)$, where s is the maximum singular value of the weight matrix, λ is a parameter related to the Graph Laplacian, and L is the number of layers. We demonstrate that, as long as the features of vertices are *independent*, applying the Global Self-attention mechanism is equivalent to performing convolution with a substituting weight matrix with a larger maximum singular value. We remark that our proof follows an idea similar to [7]. The difference is that the proof in [7] established a notion of the ϵ-smoothing layer, while our proof is underpinned by the explicit increment of s.

In the below sections, unless explicitly specifying, we use shorted-handed notations $\boldsymbol{H} := \boldsymbol{H}^{(l)}$ and $\boldsymbol{W} := \boldsymbol{W}^{(l)}$ to denote the output matrix at any layer. The convolutional result of each layer can therefore be denoted as

$$(\boldsymbol{H} + \gamma \tilde{\boldsymbol{A}}^{-1}\boldsymbol{B}\boldsymbol{H})\boldsymbol{W}, \tag{7}$$

where $\tilde{\boldsymbol{A}}$ invertible as it is an approximation of the graph Laplacian. We assume the output matrices $\boldsymbol{H}$ satisfy the following property:

Assumption 2. $\boldsymbol{H}$ is of full column ranks. Notice that this implies $(\boldsymbol{H}^T\boldsymbol{H})$ is invertible, Hermitian and positive definite. We further assume $(\boldsymbol{H}\boldsymbol{H}^T)$ is invertible.

Also, we employ the following approximation of $\boldsymbol{B}$ and $\tilde{\boldsymbol{A}}$ in the analysis:

Assumption 3. We use $\hat{\boldsymbol{B}}$ as an Hermitian and positive definite approximation of $\boldsymbol{B}$. Moreover, we use $\tilde{\boldsymbol{A}}' = (1+\epsilon)\boldsymbol{I} + \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$ ($\epsilon > 0$) as a variation of graph Laplacian to approximate $\tilde{\boldsymbol{A}}$. Notice that $\tilde{\boldsymbol{A}}'$ is also Hermitian and positive definite as it is strictly diagonally dominant.

Remark 3. The above assumptions and approximation are easy to satisfy in real-world applications. Assumption 2 almost merely assumes the features of different dimensions are *independent*. The $\hat{\boldsymbol{B}}$ matrix can be viewed as a pre-Softmax version of $\boldsymbol{B}$, which is indeed Hermitian and the main diagonal of $\boldsymbol{B}$ is at least as large as any other element. Also, $\tilde{\boldsymbol{A}}'$ without the ϵ term will be Hermitian and positive semi-definite itself – and the input corresponds to the 0 eigenvalue is unique.

Denote the output of equation 7 as $\boldsymbol{H}\tilde{\boldsymbol{W}}$, with some simple algebraic manipulation, the expression of $\tilde{\boldsymbol{W}}$ will be:

$$\begin{aligned}\tilde{\boldsymbol{W}} &= (\boldsymbol{I} + \gamma(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\tilde{\boldsymbol{A}}^{-1}\boldsymbol{B}\boldsymbol{H})\boldsymbol{W} \\ &\approx (\boldsymbol{I} + \gamma(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\tilde{\boldsymbol{A}}'^{-1}\hat{\boldsymbol{B}}\boldsymbol{H})\boldsymbol{W}.\end{aligned} \tag{8}$$

To simplify the notations, we define $\hat{\boldsymbol{C}} := \tilde{\boldsymbol{A}}'^{-1}\hat{\boldsymbol{B}}$, which is a Hermitian and positive definite matrix. Finally, we also set $\gamma > 0$ in the analysis as it will be otherwise without the GSA mechanism.

Now we are ready to introduce the definitions and results in [8] as our preliminaries. The main concepts of [8] are the invariant subspace and its distance between a matrix. Formally, they are defined as follows:

Definition IV.2 ([8])**.** For $M, N \in \mathbb{N}^+$ and $M < N$, suppose there exists a subspace $U \subseteq \mathbb{R}^M$ for $\mathbb{R}^N$ such that:

- U has orthonormal basis $\boldsymbol{E}$ consisting of non-negative vectors;
- U is invariant under P;

then we define $\mathcal{M} := U \otimes R^C = \{\sum_{i=1}^M \boldsymbol{e}_i \otimes \boldsymbol{k}_i | \boldsymbol{k}_i \in \mathbb{R}^C\} = \{\boldsymbol{E}\boldsymbol{K} | \boldsymbol{K} \in \mathbb{R}^{M\times C}\}$ as the subspace of $\mathbb{R}^{N\times C}$. Furthermore, for any matrix $\boldsymbol{H} \in \mathbb{R}^{N\times C}$, we define the distance between $\boldsymbol{H}$ and $\mathcal{M}$ as $d_{\mathcal{M}}(\boldsymbol{H}) := \inf_{\boldsymbol{Y}\in\mathcal{M}}\{\|\boldsymbol{H} - \boldsymbol{Y}\|_F\}$.

Following the notions in the literature, we denote the maximum singular value of $\boldsymbol{W}^l$ as s_l, and let $s := \sup_{l\in\mathbb{N}^+} s_l$. We recall Theorem 2 and Corollary 2 of [8] for a multi-layer GCN to form the following lemma:

Lemma IV.2 ([8])**.** *For any initial value $\boldsymbol{H}^{(0)}$ and the output of the l-th layer as $\boldsymbol{H}^{(l)}$, the convergence rate to $\mathcal{M}$ is $d_{\mathcal{M}}(\boldsymbol{H}^{(l)}) = O((s\lambda)^l)$, where λ is the second-largest eigenvalue of the normalized graph Laplacian. Furthermore, $\boldsymbol{H}^{(l)}$ satisfies $d_{\mathcal{M}}(\boldsymbol{H}^{(l)}) \leq (s\lambda)^l d_{\mathcal{M}}(\boldsymbol{H}^{(0)})$.*

In the above lemma we assume $s\lambda < 1$, which means $d_{\mathcal{M}}(\boldsymbol{H}^{(l)})$ will exponentially converge to 0. We are now ready to demonstrate our main theorem for over-smoothing:

Theorem IV.3. *$\forall\, l \in \mathbb{N}^+$, let $\boldsymbol{H}^{(l)}$ and $\hat{\boldsymbol{H}}^{(l)}$ denote the output of plain GCNs and GSA mechanism-augmented GCNs. The convergence rate to $\mathcal{M}$ for GSA mechanism-augmented GCNs is asymptotically slower than the former. i.e. $d_{\mathcal{M}}(\hat{\boldsymbol{H}}^{(l)}) = \omega(d_{\mathcal{M}}(\boldsymbol{H}^{(l)}))$. Furthermore, there exists $\boldsymbol{H}^{(l)}$ on layer l and the choice of γ such that $d_{\mathcal{M}}(\hat{\boldsymbol{H}}^{(l+1)}) > d_{\mathcal{M}}(\boldsymbol{H}^{(l+1)})$.*

The proof of theorem IV.3 crucially relies on the fact that for each layer, the maximum singular value of $\tilde{\boldsymbol{W}}$ is larger than its counterpart in $\boldsymbol{W}$. En route to proving theorem IV.3, we will show the following lemmas. We use $\lambda_{\min}(\cdot)$ and $\sigma_{\min}(\cdot)$ to denote the minimum eigenvalue and singular value.

Lemma IV.4. *The minimum eigenvalue and singular value of matrix $(\boldsymbol{I} + \gamma(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\hat{\boldsymbol{C}}\boldsymbol{H})$ in equation 8 are greater than 1. I.e. let $\boldsymbol{P} := (\boldsymbol{I} + \gamma(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\hat{\boldsymbol{C}}\boldsymbol{H})$, we have $\lambda_{\min}(\boldsymbol{P}) > 1$ and $\sigma_{\min}(\boldsymbol{P}) > 1$.*

Proof. We first prove the matrix $\boldsymbol{Q} := \gamma(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\hat{\boldsymbol{C}}\boldsymbol{H}$ is Hermitian and positive definite. According to Assumption 2, since $\boldsymbol{H}^T\boldsymbol{H}$ is Hermitian and positive definite, $(\boldsymbol{H}^T\boldsymbol{H})^{-1}$ is Hermitian and positive definite; and since $\hat{\boldsymbol{C}}$ is Hermitian and positive definite, $\boldsymbol{H}^T\hat{\boldsymbol{C}}\boldsymbol{H}$ is Hermitian and positive definite. Thus, $\boldsymbol{Q}$ is positive definite by multiplication. We can show $\boldsymbol{Q}$ is Hermitian by showing $\bar{\boldsymbol{Q}}$ is Hermitian:

$$\begin{aligned}\bar{\boldsymbol{Q}} &= (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\hat{\boldsymbol{C}}\boldsymbol{H} \\ &= (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{H}^T)(\boldsymbol{H}\boldsymbol{H}^T)^{-1}\hat{\boldsymbol{C}}\boldsymbol{H} \\ &= \boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{H}^T)^{-1}\hat{\boldsymbol{C}}\boldsymbol{H} \\ &= \boldsymbol{H}^T\hat{\boldsymbol{C}}\hat{\boldsymbol{C}}^{-1}(\boldsymbol{H}\boldsymbol{H}^T)^{-1}\hat{\boldsymbol{C}}\boldsymbol{H} \\ &= \boldsymbol{H}^T\hat{\boldsymbol{C}}(\boldsymbol{H}\boldsymbol{H}^T\hat{\boldsymbol{C}})^{-1}\hat{\boldsymbol{C}}\boldsymbol{H} \\ &= \boldsymbol{H}^T\hat{\boldsymbol{C}}(\boldsymbol{H}\boldsymbol{H}^T\hat{\boldsymbol{C}})^{-1}\hat{\boldsymbol{C}}\boldsymbol{H}(\boldsymbol{H}^T\boldsymbol{H})(\boldsymbol{H}^T\boldsymbol{H})^{-1} \\ &= \boldsymbol{H}^T\hat{\boldsymbol{C}}(\boldsymbol{H}\boldsymbol{H}^T\hat{\boldsymbol{C}})^{-1}(\boldsymbol{H}\boldsymbol{H}^T\hat{\boldsymbol{C}})\boldsymbol{H}(\boldsymbol{H}^T\boldsymbol{H})^{-1} \\ &= \boldsymbol{H}^T\hat{\boldsymbol{C}}\boldsymbol{H}(\boldsymbol{H}^T\boldsymbol{H})^{-1} = \bar{\boldsymbol{Q}}^T.\end{aligned}$$

Since $\gamma > 0$, we can conclude that $\boldsymbol{Q}$ is Hermitian, and its minimum eigenvalue $\lambda_{\min}(\boldsymbol{Q}) > 0$. Now we introduce a theory in [26], which says that if $\boldsymbol{A}$ and $\boldsymbol{B}$ are Hermitian matrices with eigenvalues denoted in descending order. i.e. $\lambda_1(\boldsymbol{A}) >$

$\lambda_2(\boldsymbol{A}) > \cdots > \lambda_n(\boldsymbol{A})$ and $\lambda_1(\boldsymbol{B}) > \lambda_2(\boldsymbol{B}) > \cdots > \lambda_n(\boldsymbol{B})$, and if $\boldsymbol{A} + \boldsymbol{B} = \boldsymbol{C}$ with the eigenvalue of $\boldsymbol{C}$ denoted in descending order. i.e. $\lambda_1(\boldsymbol{C}) > \lambda_2(\boldsymbol{C}) > \cdots > \lambda_n(\boldsymbol{C})$, then there is inequality:

$$\lambda_{n-i-j}(\boldsymbol{C}) \geq \lambda_{n-i}(\boldsymbol{A}) + \lambda_{n-j}(\boldsymbol{B}).$$

If we set $i = 0, j = 0$ for the above inequality, and consider the condition that identity matrix is Hermitian and with eigenvalue 1, then we will have:

$$\lambda_{\min}(\boldsymbol{P}) \geq \lambda_{\min}(\boldsymbol{I}) + \lambda_{\min}(\boldsymbol{Q}) > 1.$$

And since $\boldsymbol{P}$ is obviously Hermitian, which means its smallest singular value should be the absolute value of its smallest eigenvalue. Therefore, we have $\sigma_{\min}(\boldsymbol{P}) > 1$. □

Now we give the following proposition to help compare the singular values:

Proposition 1. *Let $\|\cdot\|$ denotes the 2-norm, and given two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ where $\boldsymbol{A}$ is a square matrix, then the inequalities $\sigma_{\min}(\boldsymbol{A})\|\boldsymbol{B}\| \leq \|\boldsymbol{A}\boldsymbol{B}\| \leq \sigma_{\max}(\boldsymbol{A})\|\boldsymbol{B}\|$ hold.*

The proof of proposition 1 can be found in most matrix analysis textbooks. Now we can demonstrate the relation between the singular values of $\tilde{\boldsymbol{W}}$ and $\boldsymbol{W}$.

Lemma IV.5. *$\forall\ l \in \mathbb{N}^+$, let $s^{(l)}$ and $\tilde{s}^{(l)}$ be the maximum singular values of $\boldsymbol{W}^{(l)}$ and $\tilde{\boldsymbol{W}}^{(l)}$, respectively. Then there is $\tilde{s}^{(l)} > s^{(l)}$.*

Proof. The proof follows a simple combination of the results of lemma IV.4 and proposition 1. Let $||\cdot||$ be a 2-norm, then one will have:

$$\begin{aligned}
\tilde{s}^{(l)} &= \|\tilde{\boldsymbol{W}}^{(l)}\| \\
&= \|(\boldsymbol{I} + \gamma(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\hat{\boldsymbol{C}}\boldsymbol{H})\boldsymbol{W}^{(l)}\| \\
&\geq \sigma_{\min}(\boldsymbol{I} + \gamma(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\hat{\boldsymbol{C}}\boldsymbol{H})\|\boldsymbol{W}^{(l)}\| \\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{(proposition 1)} \\
&> \|\boldsymbol{W}^{(l)}\| = s^{(l)} \qquad\qquad\qquad \text{(lemma IV.4)}
\end{aligned}$$

□

Proof of Theorem IV.3

The asymptotic relation on convergence rate can be shown straightforwardly by taking the limit of the rate in lemma IV.2. Let $s = \sup_{l\in\mathbb{N}^+} s_l$ and $\hat{s} = \sup_{l\in\mathbb{N}^+} \hat{s}_l$ be the largest singular values among all layers, one can get:

$$\lim_{l\to\infty} \frac{d_{\mathcal{M}}(\hat{\boldsymbol{H}}^{(l)})}{d_{\mathcal{M}}(\boldsymbol{H}^{(l)})} = \lim_{l\to\infty} (\frac{\hat{s}}{s})^l = \infty \qquad (\tfrac{\hat{s}}{s} > 1)$$

The proof of the second part of the theorem relies on the Proposition 4 of [8], which stated that for some $\boldsymbol{H}$, if $s^{(l)}\lambda > 1$, we will have $d_{\mathcal{M}}(\boldsymbol{H}^{(l+1)}) > d_{\mathcal{M}}(\boldsymbol{H}^{(l)})$. Thus, for layer l, if we choose γ so that $s^{(l)}\lambda < 1$ and $\hat{s}^{(l)}\lambda > 1$, then there exists input $\boldsymbol{H}^{(l)} = \hat{\boldsymbol{H}}^{(l)}$ so that:

$$\begin{aligned}
d_{\mathcal{M}}(\boldsymbol{H}^{(l+1)}) &\leq (s^{(l)}\lambda)d_{\mathcal{M}}(\boldsymbol{H}^{(l)}) \\
&< (\hat{s}^{(l)}\lambda)d_{\mathcal{M}}(\hat{\boldsymbol{H}}^{(l+1)}) \\
&< d_{\mathcal{M}}(\hat{\boldsymbol{H}}^{(l+1)})
\end{aligned}$$

V. Empirical Analysis of the GSA Mechanism

In this section, we empirically evaluate the impact of the GSA mechanism on GCNs in terms of the expressive power and the abilities to mitigate overfitting and over-smoothing. We organize the layers in the same way of Eq (4), and simply put it over the plain GCNs. For a convinient notation, we name the above model as *GSA-GCN* in this section. We evaluate its performance with multiple benchmark datasets on two tasks: (I) node classification with citation network datasets: Cora, Citeseer and Pubmed [27], aiming to classify academic papers into various subjects, (II) graph classification on the COIL-RAG dataset. We split the experiments into three parts to show the respective properties of the GSA-GCN. For the node classification task, we firstly apply the GSA-GCN to both supervised and semi-supervised node classifications. The results demonstrate enhanced accuracy and overfitting resistance for the GSA-GCN. Secondly, to corroborate the theorem for the GSA mechanism to mitigate over-smoothing, we compare the training accuracy for GCNs and GSA-GCNs as the number of layers goes deep. And finally, we extend the comparison between GSA-GCNs and plain GCNs to graph classification task with the COIL-RAG dataset to illustrate the preferable expressive power of GSA-GCNs beyond classical node classification. The metadata of aforementioned datasets are illustrated in table I.

TABLE I: Datasets used in the Experiments

Dataset	Graphs	Nodes	Edges	Classes	Features
Cora	1	2708	5429	7	1433
Citeseer	1	3327	4732	6	3703
Pubmed	1	19717	44338	3	500
COIL-RAG	3900	3.01	3.02	100 (graph)	64

A. Semi- and Full-supervised Node Classification

The semi-supervised experiment follows the setup in [1], and the full-supervised experiment follows the practice in [7], [28]. We compare the performance of GSA-GCN with several recently-proposed compelling models based on GCNs, and the two-layer GCN backbone is considered as baseline unless otherwise specified. For semi-supervised learning, only a marginal portion of labeled instances are used in training process, thus it yields weaker performance than full-supervised learning. Table II demonstrates test accuracy on three datasets by GCNs [1], Graph Attention Networks [3], Jumping Knowledge Networks [29] and DropEdge-GCN [7]. We also summarize previous state-of-the-art testing accuracy on three datasets with full-supervised learning in Table III. The performances of comparison models are either as reported in original paper or obtained by fine-tuning to the best performance as we are able to achieve.

TABLE II: Semi-supervised Node Classification Accuracy(%)

Model	Cora	Citeseer	Pubmed
GCN	81.5	70.3	79.0
GAT	83.0	72.5	79.0
JK-Net (4)	80.2	68.7	78.0
DropEdge-GCN	82.8	72.3	79.6
GSA-GCN	**83.3**	**72.9**	**80.1**

TABLE III: Full-supervised Node Classification Accuracy(%)

Model	Cora	Citeseer	Pubmed
GCN	86.1	75.9	90.2
GAT	86.4	76.6	OOM
JK-Net	86.9	78.3	90.5
DropEdge-GCN	86.5	78.7	**91.2**
GSA-GCN	**88.2**	**79.1**	89.4

As illustrated in Table II and Table III, the GSA-GCN model outperforms other advanced methods on almost all the datasets for both tasks. The only exception is the full-supervised classification on Pubmed dataset. Notice some other state-of-the-art methods apart from the ones listed in the tables *may* have reported a higher accuracy than GSA-GCN. Nevertheless, our intention is *not* to show the superiority of the GSA-GCN model, but to verify the *merits of the GSA mechanism*. To this end, the results are sufficiently convincing as the simple GSA mechanism on plain GCNs can lead to performances surpassing the listed advanced methods, which are carefully-designed and complicated. That is to say, consistent with the intuition, the GSA mechanism indeed introduce compelling expressive power to the GCNs. Furthermore, the significant performance improvement for the semi-supervised classification task ratifies the capacities for the GSA mechanism to mitigate overfitting.

B. Over-smoothing

To provide empirical evidences for Theorem IV.3, we show the changes of training accuracy with an increasing number of layers on plain GCN and GSA-GCN. Two fully-supervised node classification datasets (Cora and Citeseer) are employed in this part of the experiment. Notice that since over-smoothing will obstruct the accuracy optimization on both training and testing sets, demonstrating the results on the *training* accuracy alone is sufficient for our purpose.

The results can be summarized in Figure 2 and Figure 3. From the figures, it can be observed that GSA-GCN can consistently achieve a higher training accuracy than its plain counterpart. Moreover, as the graph network deepens to a further extent, the gap becomes more significant. We observe from the training dynamic that the training accuracy of plain GCN for deep models typically stuck at a low level after few epochs, while the curves in GSA-GCN illustrate a fluctuating pattern although it does not converge to a high value.

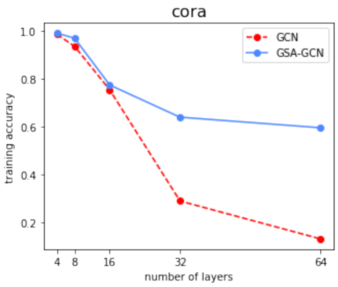

Fig. 2: Over-smoothing with model depth on Cora

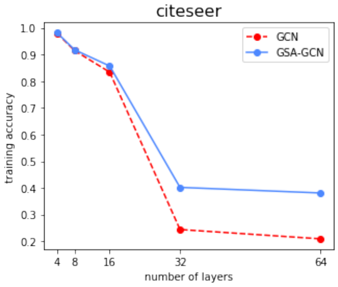

Fig. 3: Over-smoothing with model depth on Citeseer

C. Graph Classification

Limited by space, for the graph classification task, we only illustrate the results on the COIL-RAG dataset. We tested the results on plain and GSA-GCN, and referred an external result from [30]. Our read-out layer for graph classification is based on summation pooling, and the weight regularization is set to $1e-4$. The results are summarized in Table IV.

TABLE IV: Graph Classification Performance of GCNs and GSA-GCNs

Model	Training Accuracy	Test Accuracy
GCN	95.35	85.15
SPI-GCN [30]	N.A.	75.72
GSA-GCN	**97.34**	**88.28**

From the table, it can be observed that the GSA-GCN model can provide the optimal performance among the methods implemented. This positive result indicates that the gain on the expressive power from the GSA mechanism is not limited to node classification tasks. To further illustrate the insights of the GSA-GCN, we plot the training and testing accuracy curves with respect to the number of epochs in Figure 4. From the figure, it can be observed that the curves of both the training and testing accuracy of GSA-GCN stand to a higher level than their plain GCN counterpart. The curve also supports the overfitting-resistance property of the GSA mechanism: we can observe that in the later epochs, the testing accuracy of the plain GCN starts to decline, while the test accuracy for the GCN-GSA continues to climb.

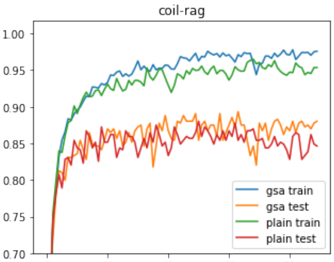

Fig. 4: Graph Classification Accuracy for Plain and GSA GCNs

VI. Conclusion

In this paper, we study the impact of the Global Self-Attention mechanism on Graph Convolutional Networks. To the best of the our knowledge, this is the first concrete attempt to understand such mechanism on GCNs. We first provide a straightforward way to implement the GSA mechanism to GCNS. Furthermore, we theoretically prove that GSA mechanism can mitigate overfitting and over-smoothing problems in GCN-based models based on some recent results. Experiments on two classical tasks illustrate superior expressive power of GSA mechanism against advanced (and much more complicated) variations of the GCN. Additionally, the theoretical results on the alleviation of overfitting and over-smoothing are reflected in the empirical results.

Acknowledgment

The authors would like to thank Mr. Shixiong Wang at National University of Singapore for constructive discussions.

References

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

[2] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, “Relational inductive biases, deep learning, and graph networks,” *arXiv preprint arXiv:1806.01261*, 2018.

[3] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” *arXiv preprint arXiv:1710.10903*, 2017.

[4] J. B. Lee, R. Rossi, and X. Kong, “Graph classification using structural attention,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 1666–1674.

[5] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, “A comprehensive survey on graph neural networks,” *arXiv preprint arXiv:1901.00596*, 2019.

[6] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, “Self-attention generative adversarial networks,” *arXiv preprint arXiv:1805.08318*, 2018.

[7] Y. Rong, W. Huang, T. Xu, and J. Huang, “Dropedge: Towards deep graph convolutional networks on node classification,” in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=Hkx1qkrKPr

[8] K. Oono and T. Suzuki, “Graph neural networks exponentially lose expressive power for node classification,” in *International Conference on Learning Representations*, 2020.

[9] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral networks and locally connected networks on graphs,” *arXiv preprint arXiv:1312.6203*, 2013.

[10] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[11] M. Henaff, J. Bruna, and Y. LeCun, “Deep convolutional networks on graph-structured data,” *arXiv preprint arXiv:1506.05163*, 2015.

[12] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *Advances in Neural Information Processing Systems*, 2017, pp. 1024–1034.

[13] R. Li, S. Wang, F. Zhu, and J. Huang, “Adaptive graph convolutional neural networks,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[14] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, “Cayleynets: Graph convolutional neural networks with complex rational spectral filters,” *IEEE Transactions on Signal Processing*, vol. 67, no. 1, pp. 97–109, 2018.

[15] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1263–1272.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[17] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung, “Gaan: Gated attention networks for learning on large and spatiotemporal graphs,” in *UAI*, 2018.

[18] C. Wang, C. Deng, and V. Ivanov, “Sag-vae: End-to-end joint inference of data representations and feature relations,” *arXiv preprint arXiv:1911.11984*, 2019.

[19] M. Xu, W. Dai, C. Liu, X. Gao, W. Lin, G.-J. Qi, and H. Xiong, “Spatial-temporal transformer networks for traffic flow forecasting,” *ArXiv*, vol. abs/2001.02908, 2020.

[20] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, “Geom-gcn: Geometric graph convolutional networks,” in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=S1e2agrFvS

[21] Y. Rong, W. Huang, T. Xu, and J. Huang, “The truly deep graph convolutional networks for node classification,” *arXiv preprint arXiv:1907.10903*, 2019.

[22] G. Li, M. Müller, A. K. Thabet, and B. Ghanem, “Deepgcns: Can gcns go as deep as cnns?” in *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[23] Q. Li, Z. Han, and X.-M. Wu, “Deeper insights into graph convolutional networks for semi-supervised learning,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[24] F. Wu, T. Zhang, A. H. d. Souza Jr, C. Fifty, T. Yu, and K. Q. Weinberger, “Simplifying graph convolutional networks,” *arXiv preprint arXiv:1902.07153*, 2019.

[25] B. Bollobás, *Graph theory: an introductory course*. Springer Science & Business Media, 2012, vol. 63.

[26] A. Knutson and T. Tao, “Honeycombs and sums of hermitian matrices,” 2001.

[27] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, “Collective classification in network data,” *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[28] J. Chen, T. Ma, and C. Xiao, “Fastgcn: fast learning with graph convolutional networks via importance sampling,” *arXiv preprint arXiv:1801.10247*, 2018.

[29] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, “Representation learning on graphs with jumping knowledge networks,” *arXiv preprint arXiv:1806.03536*, 2018.

[30] A. Atamna, N. Sokolovska, and J.-C. crivello, “SPI-GCN: A Simple Permutation-Invariant Graph Convolutional Network,” Apr 2019, preprint, HAL hal-02093451. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02093451