GSA-GCN: Leveraging Global Information in Self-attention Graph Convolution Networks

Chen Wang
Department of Computer Science
Rutgers University
Piscataway, NJ 08854
`chen.wang@cs.rutgers.edu`

Chengyuan Deng
Department of Computer Science
Rutgers University
Piscataway, NJ 08854
`charles.deng@rutgers.edu`

Abstract

We proposed Global Self-Attention Graph Convolutional Networks based on recent advances in the self-attention mechanism. The new technique allows Graph Convolutional Networks to leverage global information regardless of local edge geometry, and can therefore improve the quality of information propagation by taking advantage of node features. Experimental results on different datasets illustrate favorable properties for the proposed method, and the performance on generalization is especially remarkable.

1 Introduction

In this project, we designed a novel Global Self-attention Graph Convolutional Network (GSA-GCN) to incorporate global feature similarity into graph information propagation. The Graph Convolutional Network (GCN) framework [1] has prompted Graph Networks to be one of the most promising techniques in pursuing Artificial General Intelligence [2], and different attention/self-attention mechanisms have been proposed to improve the quality of information aggregation under this 1-st order method (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 considers *local geometry* from the edge connections of the graph, and exclude possible scenarios when a vertex can have strong correlation/influence with another *without edge connection*. To date, as we can see from a comprehensive survey [5], there has now been any publication considering such issues.

The idea of the Global Self-attention Graph Convolutional Network is based on Self-attention Generative Adversarial Networks (GANs) proposed in [6]. As a direct analogy, each node of a graph can be roughly viewed as a pixel of an image, and as the edge connections dictates information pasing, it can be deemed as an analogy of local convolutional kernels in CNNs. Based on this idea, we designed a method that computes self-attention features based entirely on the similarity of node features (similar to the method using features of pixels in [6]). Of course, if we simply drop the information of local edge connections and formulate layers only with this geometry-irrelevant attention mechanism, the model will no longer be a Graph Neural Network at all. Thus, as the similar mechanism in [6], our model applies the self-attention mechanism to get a 'global attention feature', and performs local geometry-based graph convolution without attention afterwards. It is also noticeable that a (though unproved) local-global information trade-off could exist, thus an interpolation parameter is also introduced.

Figures 1 and 2 illustrate an intuitive comparison between existing attention mechanisms and the global self-attention approach. From the figures, it can be observed that existing self-attention mechanisms rely primarily on local geometry. The idea behind the attention mechanism is that instead of uniformly aggregating information from neighboring nodes, the information of the similar neighboring nodes looks much more helpful, and therefore will be augmented for propagation.

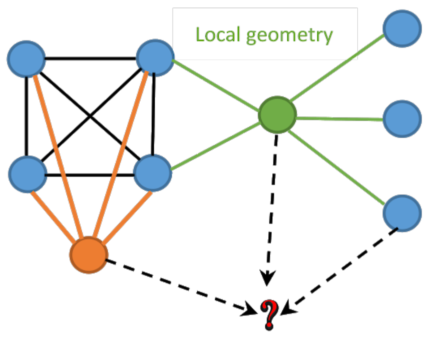

Figure 1: Self-attention in local geometry

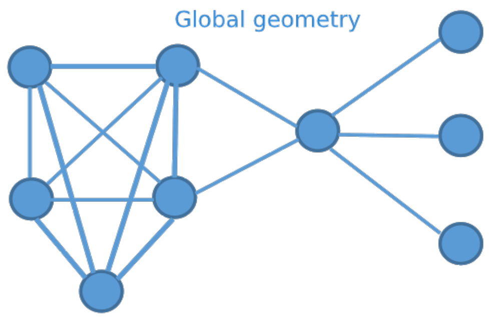

Figure 2: Global self-attention

However, suppose in figure 1, the 3 right-most nodes share a significant similarity with the strongly-connected 4 blue nodes, and they are not geometrically connected for some reason, then the local geometry-based attention approaches will lose its advantages and the network will be prone to converge to a sub-optimal solution. In contrast, for the global self-attention mechanism, we will be able to leverage the information from nodes with similar features regardless of the local geometric properties.

It is worthy to mention that the GCN framework proposed by [1] is equivalent to a 1-st order Chebyshev approximation of the Fourier transformation. As we observed favorable performances from the proposed GSA-GCN, we suspect that the underlying reason behind (in addition to the engineering reason of incorporating more reliable information) is that our approach can serve as a powerful correction of the residuals of this approximation. However, en route of pursuing this direction, we have not shown any rigorous proof yet, and this can be a work in the future.

1.1 Related Work

The research line of generalizing convolutions in graph domain can date back to [7], since then, considerable work have been concentrating on reducing the computational complexity by optimizing convolution filters [8, 9, 10, 11] until [1] simplified the computation by approximation with Chebyshev polynomials in Fourier domain. More extensions and variations based on [1] have been proposed [12, 13, 14]. A comprehensive survey [5] provides a overall reveiw on the rapid development of GCNs recently.With increasing notice on attention mechanism, leveraging attention with GNNs also illustrates promising performance[3], particularly on graph classification tasks[15, 16].

Transformer architectures have triumphed traditional recurrent and convolutional models on various tasks[17, 18, 19]. For Generative Adversial Networks, AttnGAN [20] uses attention over word embeddings within an input sequence and SAGAN [6] is first-of-its-kind to leverage self-attention over internal model states.

2 Method

2.1 Graph Convolution Networks

Graph Convolutional Networks, first proposed in [1], introduce a scalable approach based on approximation of spectral convolutions on graphs. Given a graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, a Graph Convolutional Networks takes as input a set of features x_i for every note $V_i \in \boldsymbol{V}$, resulting in a node-feature matrix representation $\boldsymbol{X}$ of a graph $n \times d$ where n is number of nodes and d is number of features. Additionally, a Graph Convolutional Network input will include a graph adjacency matrix $\boldsymbol{A}$ with the size of $n \times n$ to represent the local geometry as edge connections. Finally, the network layers are connected using a convolutional projection of input graph $\boldsymbol{X}$ with adjacency matrix $\boldsymbol{A}$, therefore we have the following non-linear function for each layer:

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

where $\hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ with $\boldsymbol{A}$ as the original adjacency matrix and $\boldsymbol{I}$ as the identity matrix of the main diagonal. $\boldsymbol{W}^{(l)}$ is the weight matrix for convolution of the l-th layer and $\sigma(\cdot)$ is the activation function. Matrix $\boldsymbol{D}$ serves as the normalization matrix of $\hat{A}$:

$$\boldsymbol{D} = \texttt{diag}\left(\sum_{j=1}^{N} \hat{\boldsymbol{A}}_j\right) \tag{2}$$

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.

2.2 Global Self-attention Mechanism

The self-attention mechanism on image data with Convolutional Neural Networks is discussed thoroughly in [6]. In this section, we discuss the self-attention mechanism in GCN following the method in [6] with modification to the graph structure. 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 the following equation:

$$\beta_{i,j} = \texttt{softmax}_{j\in\{1,2,\cdots,n\}}[s_{i,j}], \qquad \text{where } s_{i,j} = (\hat{\boldsymbol{H}}_i^{(l)}\boldsymbol{W_l})(\hat{\boldsymbol{H}}_j^{(l)}\boldsymbol{W_r})^T \tag{3}$$

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 $\boldsymbol{\beta}$, and it will be a $n \times n$ matrix. With the attention importance mask, the attention feature can be calculated with:

$$\boldsymbol{o}_i^{(l)} = (\sum_{j=1}^{N} \beta_{i,j}\boldsymbol{H}_j^{(l)}\boldsymbol{W}_h)\boldsymbol{W}_g \tag{4}$$

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.

2.3 Global Self-Attention Graph Convolutional Network

The global self-attention layer correctly captures the feature information on a global level. Nevertheless, as we have discussed before, 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 graph convolution after obtaining the self-attention features:

$$\boldsymbol{H}^{(l+1)} = \sigma(\hat{\boldsymbol{D}}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{O}^{(l)}) \tag{5}$$

where $\boldsymbol{O}^{(l)}$ is the collection of n self-attention outputs. To trade-off local and global geometries, [6] introduces an interpolation. And here, we can apply a similar procedure:

$$\boldsymbol{H}^{(l+1)} = \sigma(\hat{\boldsymbol{D}}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-\frac{1}{2}}(\gamma\boldsymbol{O}^{(l)} + \boldsymbol{H}^{(l)})) \tag{6}$$

where γ is assigned with an initial value of 0 and it can be trained over the process. In addition, one can apply linear feature transformation to $\boldsymbol{H}^{(l)}$ before proceeding it to the self-attention layer (as in the standard GCN setup), although this step has been argued as unnecessary in [21]. Taking together all the above operations, a single layer of Global Self-Attention Graph Convolutional Network can be given as algorithm 1.

The proposed GAS-GCN is a general framework that can be applied to various tasks in the regime of graph machine learning, such as graph node classification, graph classification, and graph embedding. For node-level tasks, one can directly retrieve information from GAS-GCN and perform prediction and back-propagation. On the other hand, for graph-level tasks, one can apply a read-out layer before retrieving the final output. It is noticeable that such framework can be applied to other related tasks like hidden feature relation inference (there is a paper by us earlier this year [22]). However, we do not pursue this direction in this project.

```
Algorithm 1: Single Layer of GSA-GCN
Input  : The feature matrix $\boldsymbol{H}^{(l)}$ of the previous layer
Output : The feature matrix $\boldsymbol{H}^{(l+1)}$ of the current layer
(Optional) Compute linear feature transformation with $\boldsymbol{H}^{(l)} \leftarrow \boldsymbol{H}^{(l)}\boldsymbol{W}_t$;
Compute attention feature $\boldsymbol{O}^{(l)}$ with equations 3-4.;
Compute graph convolution output $\boldsymbol{H}^{(l+1)}$ with equation 6.
```

3 Experiments

To evaluate the performance of GSA-GCN, we conducted experiments with multiple benchmark datasets on two tasks: semi-supervised node classification and supervised graph classification. For the node classification task, one node label is provided for each class. Considering the overall large amount of nodes, the supervision is very weak and a good Graph Convolutional Network should have a sufficiently satisfactory generalization ability. For the graph classification task, we apply a simple read-out layer to first sum among the nodes, and then project the dimension of features to the number of classes to get the result.

Limited by time and computational resources, the datasets tested in this project are of small sizes. For the node classification problem, we simply retrieve the largest graph from the corresponding dataset, and provide the label of the first available node (indexing order) for each class. For the graph classification problem, we set the graph size as the largest size among all the graphs, and pad other graphs with 0 rows for the node feature matrix and unit diagonal matrix for the adjacency matrix.

3.1 Node Classification

Experiments on node classification are conducted on three benchmark graph kernel datasets: Karate with 34 nodes and no given feature; Enzymes with 126 nodes and 18-dim given features; Protein with 620 nodes and 29-dim given features. For Karate dataset, since there is no given node features, the node features are calculated by passing a random initialization through 3 GCN layers; for the other two datasets, there are feature-engineering-based node features provided.

Table 1 demonstrates train and test accuracy obtained from GCNs and GSA-GCNs on the mentioned datasets. Notice that since these datasets are quite small and the features are not hard to learn, both plain and GSA-based GCNs are able to converge to an optimal if given sufficient time. Thus, the results are provided by limiting the number of epochs and the learning rate. As it has been observed, GSA-GCN can outperform plain GCN in terms of test accuracy, and the performance gap is more significant on reliable (engineering-based) features. Also, the generalization ability for GSA-GCN is consistently better tha plain GCN. This observation matches our intuition of the model, which leverages the global self-attention advantage to overcome potential overfitting of local geometries.

Table 1: Node Classification Performance of GCNs and GSA-GCNs

	Performance	Karate	Enzymes	Protein
Plain GCN	Train accuracy	0.7500	1.0	1.0
	Test accuracy	0.1667	0.7963	0.4061
GSA-GCN	Train accuracy	1.0000	1.0000	0.5
	Test accuracy	**0.2667**	**0.8145**	**0.7087**

3.2 Graph Classification

For the graph classification task, we use Enzymes and Coil-rag datasets. Notice that unlike the situation in node classification, where we only use a single graph, here we accumulate all the graphs to form a collection of matrices. Table 2 illustrates the graph classification results on the given data. From the table, it can be observed that GSA-GCN can outperform plain GCN on this type of task, and the generalization performance is again the most significant improvement. Despite the run-of-the-mill performances for both GSA- and plain GCN on the Enzymes dataset, results on Coil-rag dataset gives us a similar training accuracy but a much better test accuracy on GSA-GCN.

Table 2: Graph Classification Performance of GCNs and GSA-GCNs

	Performance	Enzymes	Coil-rag
Plain GCN	Train accuracy	0.563	0.8281
	Test accuracy	0.266	0.7032
GSA-GCN	Train accuracy	0.484	0.8283
	Test accuracy	**0.328**	**0.8435**

To further illustrate the favorable performance of GSA-GCN, we illustrate the training and testing accuracy curves with respect to the number of epochs in figures 3 and 4. From the figures, it can be observed that the curves of the training accuracy of plain and GSA-augmented GCNs are in the same pattern. For the plain GCN, the testing accuracy failed to catch up with the training performance; while for the GSA-GCN, the test accuracy can roughly stay in the same range with it training counterpart upon convergence. The lower row shows a zoom-in version of the above comparison, and the advantage is more obvious.

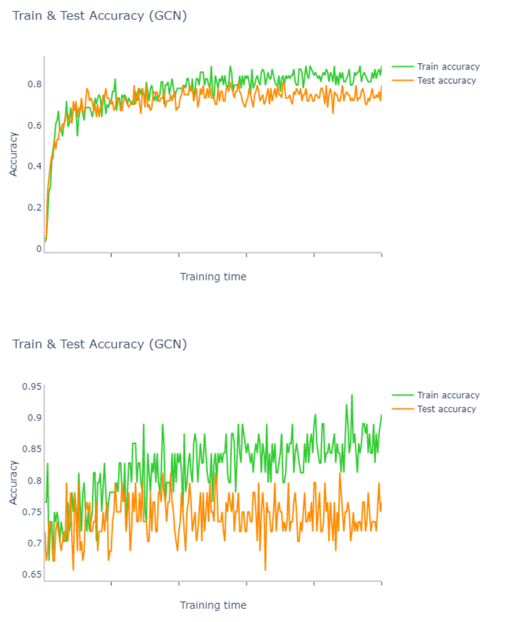

Figure 3: GCN train & test accuracy **top**: overall curves **bottom**: zoom-in of the curves near convergence

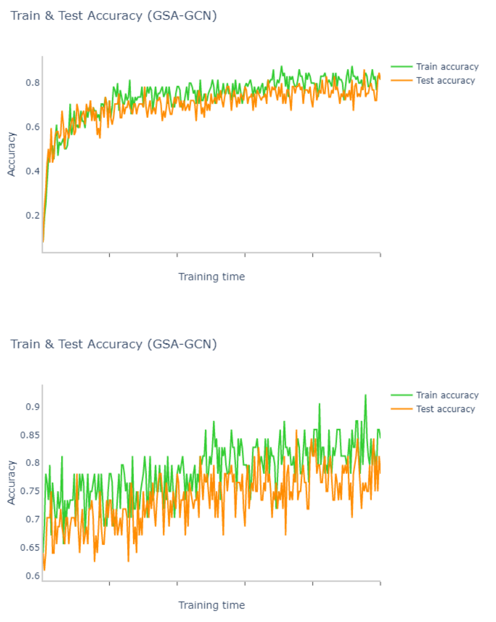

Figure 4: GSA-GCN train & test accuracy **top**: overall curves **bottom**: zoom-in of the curves near convergence

4 Conclusion

In this project, motivated by recent cutting-edge development on Graph Convolution Networks and applications of self-attention mechanism, we proposed Global Self-attention Graph Convolution Networks (GSA-GCN). To the best of the authors' knowledge, GSA-GCN is first-of-its-kind in graph machine learning to leverage global self-attention mechanism for improved information aggregation. Experimental results on two classical tasks illustrate compelling performance for GSA-augmented GCNs over their vanilla counterparts with the same network architecture. Especially, consistent with the intuition of the method, GSA-GCN illustrates favorable robustness against overfitting.

References

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

[2] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

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

[4] John Boaz Lee, Ryan Rossi, and Xiangnan Kong. Graph classification using structural attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1666–1674. ACM, 2018.

[5] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

[6] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

[7] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[8] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[9] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[10] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive graph convolutional neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[11] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2018.

[12] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1993–2001, 2016.

[13] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023, 2016.

[14] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[15] Fengwen Chen, Shirui Pan, Jing Jiang, Huan Huo, and Guodong Long. Dagcn: Dual attention graph convolutional networks. *arXiv preprint arXiv:1904.02278*, 2019.

[16] Dai Quoc Nguyen, Tu Dinh Nguyen, and Dinh Phung. Unsupervised universal self-attention network for graph classification. *arXiv preprint arXiv:1909.11855*, 2019.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[18] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. *arXiv preprint arXiv:1802.05751*, 2018.

[19] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018.

[20] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1316–1324, 2018.

[21] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153*, 2019.

[22] Chen Wang, Chengyuan Deng, and Vladimir Ivanov. Sag-vae: End-to-end joint inference of data representations and feature relations. *arXiv preprint arXiv:1911.11984*, 2019.

A Contribution

Chen Wang: initiated the idea and implemented part of the codes; formulated the mathematical derivations and expressions; pre-processing the datasets; part of the report (introduction, method)
Chengyuan Deng: Implemented part of the codes; Conducted part of experiments; Formulated background motivation; Searched for datasets; Completed part of the report (Related Work, Experiments);