Improving Graph Attention Networks with Large Margin-based Constraints

Guangtao Wang^1, Rex Ying^2, Jing Huang^1, Jure Leskovec^2
^{1}JD AI Research, Mountain View, CA
^{2}Department of Computer Science, Stanford University, Stanford, CA
^{1}{guangtao.wang, jing.huang}@jd.com
^{2}{rexying@stanford.edu, jure@cs.stanford.edu}

Abstract

Graph Attention Networks (GATs) are the state-of-the-art neural architecture for representation learning with graphs. GATs learn attention functions that assign weights to nodes so that different nodes have different influences in the feature aggregation stage of the Graph Neural Network. In practice, however, induced attention functions are prone to overfitting due to the increased number of parameters and the lack of direct supervision on attention weights. Moreover, GATs also suffer from over-smoothing the class decision boundary. Here we propose a framework to address the weaknesses of GATs by proposing the use of margin-based constraints on attention weights. We first theoretically analyze the over-smoothing behavior of GATs and then develop an approach that uses constraints on the attention weights according to class boundary and feature aggregation information. Furthermore, to alleviate the overfitting problem, we propose additional constraints on the graph structure. Extensive experiments and ablation studies demonstrate the effectiveness of our method, which leads to significant improvements over the previous state-of-the-art graph attention methods.

1 Introduction

Recently, a novel architecture leveraging attention mechanism in Graph Neural Networks (GNNs) called Graph Attention Networks (GATs) was introduced [1]. GAT was motivated by attention mechanism in natural language processing [2, 3]. It computes representation of each node by attending to its neighbors via a masked self-attention. For each node, different weights are learned for neighboring nodes by attention functions, so that the nodes in the same neighborhood have different weights in the feature aggregation step of GAT. Inspired by such attention-based architecture, several new attention-based GNNs have been proposed, and have achieved state-of-the-art performance on node classification benchmarks [4, 5, 6, 7, 8, 9].

However, attention-based GNNs suffer from problems of overfitting and over-smoothing: (1) the learned attention functions tend to overfit the training data because there is only one source to guide the attention function learning: classification error, which is indirect and limited. (2) The over-smoothing problem arises for nodes that are connected but lie on different sides of the class decision boundary. Due to information exchange over these edges, stacking multiple attention layers causes excessive smoothing of node features and makes nodes from different classes become indistinguishable.

In this paper, we propose a new Constrained Graph Attention Network (C-GAT) that adds constraints over the attention weight computation to solve the problem of overfitting and over-smoothing. We develop an aggregation strategy to further remedy the over-smoothing problem at the class boundary by selecting top k neighbors for feature aggregation. Furthermore, we also design an adaptive layer-wise negative sampling strategy to train C-GAT efficiently and effectively.

We evaluate the proposed approach on four node classification benchmarks: Cora, Citeseer and Pubmed, as well as an inductive protein-protein interaction (PPI) dataset. Our extensive experimental results and analyses demonstrate the benefit of the C-GAT model and show consistent gains over state-of-the-art graph attention models on standard benchmarks for graph node classification.

2 Analysis of GATs

Notation. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ be a graph where $\mathcal{V}$ is the set of N nodes (or vertices), $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of M edges connecting M pairs of nodes in $\mathcal{V}$, and $\boldsymbol{X} \in \mathbb{R}^{N \times d}$ represents the node input features, and each row $\boldsymbol{x}_i = \boldsymbol{X}_{i:}$. In this paper, we consider undirected graphs. Suppose $\boldsymbol{A}_{N\times N}$ is the adjacency/weighted adjacency matrix of $\mathcal{G}$ with $\boldsymbol{A}_{i,j} \geq 0$, $\boldsymbol{D} = \text{diag}(d_1, d_2, \cdots, d_N)$ and $d_i = \sum_{j=1}^{N} \boldsymbol{A}_{i,j}$. And the random walk normalized Laplacian $\boldsymbol{L}_{rw} = \boldsymbol{D}^{-1}\boldsymbol{L}$.
Node classification. Suppose that $\mathcal{V}_l \subset \mathcal{V}$ consists of a set of labeled nodes, the goal of node classification is to predict the labels of the unlabeled nodes.
Attention-based GNN utilizes the following layer-wise attention based aggregate function for node embedding on each node $v_i \in \mathcal{V}$:

$$\boldsymbol{h}_i^{(l+1)} = \sigma(\sum_{j\in\mathcal{N}_i} \alpha_{i,j}^{(l)} \boldsymbol{W}^{(l)} \boldsymbol{h}_j^{(l)}), \alpha_{i,j}^{(l)} = \frac{\exp(\phi_\omega^{(l)}(\boldsymbol{h}_i^{(l)}, \boldsymbol{h}_j^{(l)}))}{\sum_{k\in\mathcal{N}_i} \exp(\phi_\omega^{(l)}(\boldsymbol{h}_i^{(l)}, \boldsymbol{h}_k^{(l)}))} \tag{1}$$

Where $\boldsymbol{W}^{(l)} \in \mathbb{R}^{d^{(l+1)} \times d^{(l)}}$ is a trainable weight matrix shared by l-th layer. σ is the activation function. $\boldsymbol{h}_i^{(l)} \in \mathbb{R}^{d^{(l)}}$ is the node embedding in l-th layer; $\boldsymbol{h}_i^{(0)} = \boldsymbol{x}_i$. $\mathcal{N}_i$ is the set of v_i's one-hop neighboring nodes and also includes v_i (i.e. there is a self-loop on each node). $\alpha_{i,j}^{(l)}$ is the l-th attention weight between the target node v_i and the neighboring node v_j, which is generated by applying softmax to the values computed by attention function $\phi_\omega^{(l)}$, and ω is the trainable parameters of the attention function. In this paper, we use GATs to refer to all attention-based GNN models.
The overfitting problem of GATs. The attention functions in GAT compute the attention weights based on the features of pairs of connected nodes (see Eq. 1). To train such attention functions in GATs, there is only one source to learn their parameters: the classification error. In other words, the supervision of GATs to learn attention parameters is limited by the number of edges in graphs, and indirect since the supervision signal only comes from node labels for node classification. In general, smaller number of supervisions leads to overfitting [10]. The experimental study in Section 4 of robustness analysis demonstrates that the overfitting easily occurs in GATs, especially for noisy data.
The over-smoothing problem of GATs. To facilitate the analysis, we focus on the attention aggregation and simplify Eq. 1 in terms of matrix operation as $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}$[1], where $\boldsymbol{A}_{N\times N}$ is the attention matrix, $\boldsymbol{A}_{ij}^{(l)} = \alpha_{i,j}^{(l)}$ if $j \in \mathcal{N}_i$ otherwise $\boldsymbol{A}_{i,j}^{(l)} = 0$, and $\sum_{i=1}^{N} \alpha_{i,j}^{(l)} = 1$. Then we have the following proposition that indicates that a **single GAT layer** acts as a random walk Laplacian smoothing.

Proposition 1. *Let matrix ($\boldsymbol{I} - \boldsymbol{A}$) be a random walk normalized Laplacian of the graph $\mathcal{G}$. Then a single attention layer is equivalent to the Laplacian smoothing operation. (Proof in Appendix A)*
Based on the cluster assumption in node classification that the nodes in same connected component tend to share same labels [11], the smoothing results in an easier classification problem. This contributes to the better performance of GAT in node classification, compared to other GNN models.
Deeper GATs. To understand the behavior of GATs with deep layers, we first state Theorem 1 as follows: let P be a transition probability matrix of a connected undirected graph $\mathcal{G}$ with N nodes, $P^{(t)}(v_i, v_j)$ be the probability of being at node v_j after t-step random walks on $\mathcal{G}$ starting at v_i, and d_v be the degree of node v.

Theorem 1. *If graph G has no bipartite components, there exists a random walk on $\mathcal{G}$ with transition probability matrix P, that converges to a unique stationary distribution π. That is, for any pair of nodes $\{v_i, v_j\}$, $\lim_{t\to\infty} P^{(t)}(v_i, v_j) = \pi(v_j) = \frac{d_{v_j}}{\sum_k^N d_k}$. (Proof in Appendix B)*
We can view attention weight matrix $\boldsymbol{A}$ as a random walk transition probability matrix since $\boldsymbol{A}_{i,j} \geq 0$ and $\sum_{j=1}^{N} \alpha_{i,j} = 1$. In practice, attention weight matrices vary in different layers. Stacking multiple GAT layers is equivalent to a matrix chain multiplication with different attention weight matrices. Based on Theorem 1, we further derive Theorem 2 (proof in Appendix C), to show that GATs suffer from over-smoothing when they go deep, even with different attention weights at every layer.

[1]Similar to [11], we omit the non-linearity activation function σ. In fact [12] shows evidence that similar performance is observed in the case when there is no nonlinearity after the aggregation step.

Theorem 2. *Let $P^{(t)}$ $(t{\geq}1)$ be a transition probability matrix of the connected undirected graph $\mathcal{G}$, corresponding to attention scores of layer t in the GNN, then $\lim_{t\to\infty} \Pi_{i=1}^{t} P^{(t)} = \pi$, where π is the unique stationary distribution in Theorem 1.*

In practice, most graphs contain bridge nodes that connect different components with different labels. Theorem 2 states that if we increase the depth of GAT, due to the boundary nodes, the aggregated node features of different components would become indistinguishable, leading to worse performance of deep GATs (See the observation of over-smoothing in Appendix F). We call this phenomenon **over-smoothing** of graph attention. We also extend the analysis to the **multi-head** attention-based GATs in Appendix E.

Residual connection is an effective way to ensure good performance when increasing the depth of Convolutional Neural Networks [13]. It has also been employed in GATs [1, 4]. Unfortunately, this does not solve the over-smoothing problem in GATs as shown in the following.

Let $\boldsymbol{A}$ be a random walk transition probability matrix, we first define the transition probability matrix of a lazy random walk as $\boldsymbol{P} = \frac{\boldsymbol{I}+\boldsymbol{A}}{2}$. At every step, the lazy random walk has 50% probability of staying at the current node, and 50% probability of moving away from it. Hence the residual connection $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{X} = (\boldsymbol{I} + \boldsymbol{A})\boldsymbol{X}$ is a lazy random walk based smoothing up to a constant factor[2]. Therefore if the lazy random walk $\boldsymbol{P}$ is viewed as a transition probability matrix, by Theorem 2, the features of all nodes in a connected component converge to the same values if more GAT layers are stacked (Appendix D gives the analysis of a lazy random walk).

3 Constrained Graph Attention Networks

To address the problem of overfitting and over-smoothing of GATs, we propose a new framework, the constrained graph attention networks (C-GATs), via imposing constrains on both the attention function and the feature aggregate function. With these constraints, we improve the generalization ability and alleviate the problem of overfitting of GAT. In the following, we first introduce two constraints on the attention computation, which involves two margin-based losses to guide the training of graph attention. Then, based on the constrained attentions, we propose a new aggregation function, which chooses a subset of neighboring nodes based on attention weights. We show that this new scheme for feature aggregation reduces the over-smoothing of GAT.

Margin based Constraints on Attention. For a given node v_i, let $\mathcal{N}_i$ be the set of its one-hop neighboring nodes, $\mathcal{N}_i^- \subset \mathcal{N}_i$ and $\mathcal{N}_i^+ \subseteq \mathcal{N}_i$ [3] are the neighbors with different and same class labels to v_i. We propose the following two constraints on attention weights computation.

(i) *Loss from Graph Structure based Constraint* requires that attention weights between one-hop neighboring nodes are greater than the rest of the nodes.

$$\mathcal{L}_g = \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i\setminus\mathcal{N}_i^-} \sum_{k\in(\mathcal{V}\setminus\mathcal{N}_i)} \max(0, \phi(v_i, v_k) + \zeta_g - \phi(v_i, v_j)) \tag{2}$$

(ii) *Loss from Class Boundary Constraint* requires that attention weights between nodes that share the same labels are greater than those weights between nodes with different labels.

$$\mathcal{L}_b = \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i^+} \sum_{k\in\mathcal{N}_i^-} \max(0, \phi(v_i, v_k) + \zeta_b - \phi(v_i, v_j)) \tag{3}$$

Where $\zeta_g{\geq}0$ and $\zeta_b{\geq}0$ are slack variables which control the margin between attention values. $\phi(v_i, v_j)$ is the attention function. Let $\boldsymbol{h}_i{\in}\mathbb{R}^f$ and $\boldsymbol{h}_j{\in}\mathbb{R}^f$ be the features of nodes v_i and v_j, we use $\phi(\boldsymbol{h}_i, \boldsymbol{h}_j) = \text{LeakyReLU}(\text{MLP}(\boldsymbol{W}_r(\boldsymbol{h}_i || \boldsymbol{h}_j))$ to compute the attention between two nodes v_i and v_j, $\boldsymbol{W}_r \in \mathbb{R}^{f' \times 2f}$ is a trainable matrix.

Adaptive Negative Sampling for GNN Training. Negative sampling has been proved to be an effective way to optimize the loss function $\mathcal{L}_g$ in Eq. 2. We apply importance sampling to choose negative example nodes, and estimate the node importance according to Proposition 2.

Proposition 2. *The importance of a node v_i to feature aggregation of $\mathcal{G}$ in l-th layer is proportional to $\sum_{j=1}^{N} \alpha_{j,i}^{(l)}$. (See proof in Appendix H)*

With these two constraints, we optimize the following loss function for node classification:

$$\mathcal{L} = \mathcal{L}_c + \lambda_g \mathcal{L}_g + \lambda_b \mathcal{L}_b, \tag{4}$$

[2]The constant factor depends only on number of layers and is the same for all nodes

[3]$\subseteq$ is due to the self-loop connection.

	Methods	Cora	Citeseer	Pubmed	PPI*
GNN	GCN [14]	$86.3(0.4)$	$75.6(0.3)$	$86.8(0.3)$	71.0†
	G-SAGE [15]	$86.9(0.4)$	$76.5(0.4)$	$85.7(0.4)$	76.8‡
	GAT [1]	$87.2(0.3)$	$77.3(0.3)$	$87.0(0.3)$	$97.3(0.02)$
	C-GAT	$\mathbf{88.4}(0.3)$	$\mathbf{79.9}(0.3)$	$\mathbf{87.6}(0.3)$	$\mathbf{98.8}(0.05)$
Ablation	w/o $\mathcal{L}_g$	$88.3(0.2)$	$78.7(0.2)$	$87.2(0.3)$	$98.1(0.04)$
	w/o $\mathcal{L}_b$	$88.2(0.3)$	$79.3(0.3)$	$87.2(0.2)$	$97.9(0.04)$
	w/o top k	$88.4(0.2)$	$78.5(0.2)$	$87.3(0.2)$	$97.5(0.03)$
	w/o NINS*	$88.2(0.3)$	$78.9(0.2)$	$87.4(0.3)$	$98.2(0.04)$

Table 1: Classification Accuracy Ablation and Comparison (G-SAGE: GraphSAGE; the number in "()" represents std of 10 runs of the algorithm; $*$: The accuracy of the attention based GNN models on **PPI: GaAN** [5] **98.7** $\pm$ **0.02** and **GeinePath** [4] **97.9**, respectively; †: The best accuracy of GCN on PPI reported in [4]; ‡: The best accuracy of GraphSAGE on PPI reported in [1]; $\star$ NINS: Node Importance based Negative Sampling.)

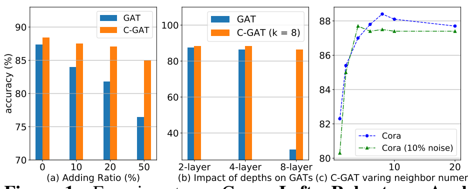

Figure 1: Experiments on Cora. **Left: Robustness Analysis**: Train on original graphs and perform testing on graphs by adding edges randomly; **Middle: Deeper GAT**: Classification accuracy comparison between GAT and C-GAT with different depth; **Right: Sensitive Analysis**: Impact of neighbor number k on classification accuracy of C-GAT.

where $\mathcal{L}_c$ represents the loss derived from the node classification error (e.g., cross entropy loss for multi-class node classification); $\lambda_g \geq 0$ and $\lambda_b \geq 0$ are two weight factors to make trade-offs among these losses, which are data dependent.

Constrained Feature Aggregation. For each node, the aggregate function only makes use of the features from neighbors with top k attention weights rather than all neighbors. From the constraint on attention computation in Eq. 3, attention weights of nodes from different classes should be small. Therefore, picking up nodes with top k attention weights would not only keep the smoothing effect on node features within the same class but also drop edges that connect different classes due to small attention weights.

4 Experimental Study

We investigate the proposed algorithm C-GAT in the following four aspects: (1) classification performance comparison; (2) robustness analysis of whether the C-GAT is effective in overcoming the overfitting problem, and improving generalization on unseen graph structure; (3) the depth of GAT models, to demonstrate whether C-GAT can prevent the over-smoothing problem suffered by GAT and (4) sensitivity analysis of the number of neighbors k used in feature aggregate functions. (See the data information and hyper-parameter setting in Appendix G)

(a) **Classification Performance Comparison:** From Table 1, we observe that C-GAT performs consistently better than all baseline models across all benchmarks. Specifically, we improve upon GAT with absolute accuracy gain of 1.2%, 2.6%, 0.6% and 1.5% on Cora, Citeseer, PubMed and PPI, respectively. Especially for the inductive learning problem PPI, we obtain the new state-of-the-art classification performance [4, 5]

(b) **Robustness Analysis:** we observe that randomly adding edges might connect different classes together. This aggravates the over-smoothing problem of GAT. However, from Fig. 1 (a), C-GAT still gets good performance even when the ratio of adding edges is up to 50%. Due to the class boundary constraint, C-GAT assigns small attention values on these boundary edges. Moreover, the proposed k selected neighbor based feature aggregation function further reduces such negative impacts (see more results on robustness in Appendix).

(c) **Deeper GAT:** Fig. 1 (b) compares C-GAT and GAT with different depths on "Cora"; In contrast to the degradation of GAT with deeper layers due to significant oversmoothing, C-GAT maintains good classification performance with bigger number of attention layers. This means that C-GAT is able to effectively overcome the problem of oversmoothing. This demonstrates the advantage of C-GAT especially in graph-level tasks where depth is critical [16].

(d) **Sensitive Analysis of Neighbor Number k:** we randomly add 10% edges to "Cora" to increase the chance of information propagation among different classes and then investigate how to set k on these noisy graphs. From Fig. 1 (c), we observe that the classification accuracy first increases to a peak value and then stabilizes or slightly decreases. This means that k plays a role of making trade-off between under-smoothing (not enough smoothing to tackle noise) and over-smoothing.

5 Conclusion

In this paper we provide analysis of the weakness of GAT models: overfitting of attention functions and over-smoothing of node representations. We propose a novel approach called constrained graph attention Network (C-GAT), to address these weaknesses of GAT by guiding the attention computation during GAT training using margin-based constraints. In addition, a layer-wise adaptive sampling approach is proposed for augmenting attention training with effective negative examples. Furthermore, to alleviate the over-smoothing problem we propose a new feature aggregate function which only selects the neighbors with top K attention weights rather than all the neighboring nodes. Extensive experiments on common benchmark datasets have verified the effectiveness of our approach, showing significant gains in accuracy on standard node classification benchmarks, especially in cases of deep models and noisy data, compared to the state-of-the-art GAT models.

References

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

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[4] Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, Le Song, and Yuan Qi. Geniepath: Graph neural networks with adaptive receptive paths. In *AAAI*, 2018.

[5] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. In *UAI*, 2018.

[6] Seongok Ryu, Jaechang Lim, and Woo Youn Kim. Deeply learning molecular structure-property relationships using graph attention neural network. *arXiv preprint arXiv:1805.10988*, 2018.

[7] John Boaz Lee, Ryan Rossi, and Xiangnan Kong. Graph classification using structural attention. In *KDD*. ACM, 2018.

[8] Kiran K Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li. Attention-based graph neural network for semi-supervised learning. *arXiv preprint arXiv:1803.03735*, 2018.

[9] Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, and Alexander A Alemi. Watch your step: Learning node embeddings via graph attention. In *NIPS*, 2018.

[10] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 2005.

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

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

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

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

[15] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

[16] Benedikt Bünz and Matthew Lamm. Graph neural networks and boolean satisfiability. *arXiv preprint arXiv:1702.03592*, 2017.

[17] Dana Randall. Rapidly mixing markov chains with applications in computer science and physics. *Computing in Science & Engineering*, 2006.

[18] László Lovász et al. Random walks on graphs: A survey. *Combinatorics, Paul erdos is eighty*, 1993.

[19] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[20] Fan Chung. Laplacians and the cheeger inequality for directed graphs. *Annals of Combinatorics*, 2005.

[21] Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. Adaptive sampling towards fast graph representation learning. In *NIPS*, 2018.

A Proof of Proposition 1

Before we give the proof, we first introduce the concepts of random walking normalized Laplacian and Laplacian smoothing as follows.

Random walking Normalized Laplacian Let $\boldsymbol{A}_{N\times N}$ be the attention weight matrix, $\boldsymbol{D} = \mathrm{diag}(d_1, d_2, \cdots, d_N)$ and $d_i = \sum_{j=1}^{N} \boldsymbol{A}_{i,j}$, then the graph Laplacian of $\mathcal{G}$ is defined as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$. And $\boldsymbol{L}_{rw} = \boldsymbol{D}^{-1}\boldsymbol{L}$ is the random walking normalized Laplacian of $\mathcal{G}$.

Laplacian Smoothing [11] on each row of the input feature matrix $\boldsymbol{X}$ is defined as:

$$\boldsymbol{y}_i = (1-\lambda)\boldsymbol{x}_i + \lambda \sum_{j}^{N} \frac{\alpha_{i,j}}{d_i}\boldsymbol{x}_j, \tag{5}$$

where $0 < \lambda \le 1$ is a parameter to controls the smoothness, *i.e.* the importance weight of the node's features with respect to the features of its neighbors. We can rewrite the Laplacian smoothing in Eq. 5 in matrix form:

$$\boldsymbol{Y} = (\boldsymbol{I} - \lambda \boldsymbol{D}^{-1}\boldsymbol{L})\boldsymbol{X} = (\boldsymbol{I} - \lambda \boldsymbol{L}_{rw})\boldsymbol{X} \tag{6}$$

Proof. As $\boldsymbol{A}_{N\times N}$ is the attention weight matrix, $d_i = \sum_{j}^{N} \alpha_{i,j} = 1$, then we can get that $\boldsymbol{D} = \boldsymbol{I}$. The random walk normalization of $\mathcal{G}$ is $\boldsymbol{L}_{rw} = \boldsymbol{D}^{-1}\boldsymbol{L} = \boldsymbol{I}^{-1}(\boldsymbol{I} - \boldsymbol{A}) = \boldsymbol{I} - \boldsymbol{A}$.
We can rewrite the graph attention operation $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}$ as $\boldsymbol{Y} = (\boldsymbol{I} - \boldsymbol{L}_{rw})\boldsymbol{X}$. According to the formulation of Laplacian smoothing in Eq. 5, we can conclude that graph attention is a special form of Laplacian smoothing with $\lambda = 1$. □

B Proof of Theorem 1

Proof. (1) We can view the random walk on graph G as a Markov chain with P. As G is undirected, connected and non-bipartite graph, the Markov chain is ergodic [17, 18]. And any finite ergodic Markov chain converges to a unique stationary distribution π [17]. (2) According to Perron-Frobenius Theorem [19, 20], such stationary distribution is just the Perron vector of P. And for the undirected graph, its Perron vector w.r.t. v_i is $d_{v_i}/\sum_{j}^{N} d_{v_j}$. □

C Proof of Theorem 2

Proof. (1) Let A_t be the attention matrix derived in t-th layer of GAT. According to Theorem 1, the random walk on the graph with A_t converges to a unique stationary distribution which depends on the degrees of the graph regardless of A_t. i.e., $\pi_t = \pi$ where π_t denotes the stationary distribution w.r.t. A_t and π is the unique stationary distribution. (2) Let f_i^k be the ith row of $\mathit{\Pi}_{t=1}^{k} A_k$, according to the converge analysis of random walk in [17], we have $||f_i^k - \pi_k|| \le \lambda_k ||f_i^{k-1} - \pi_k|| = \lambda_k ||f_i^{k-1} - \pi_{k_1}||$ as $\pi_k = \pi_{k-1}$, where λ_k is the mixing rate of random walk with A_k. By exploring the equation recursively, $||f_i^k - \pi|| \le \lambda_k ||f_i^{k-1} - \pi|| \le \cdots \le \mathit{\Pi}_{t=1}^{k}\lambda_t ||f_i^1 - \pi||$. Moreover, for strongly connected graph, the mixing rate $\lambda_t \in (0, 1)$ according to ref[1]. Then, $\lim_{k\to\infty} ||f_i^k - \pi|| = 0$. i.e., $\lim_{k\to\infty} f_i^k = \pi$. □

D Convergence Rate of Lazy Random Walk

Theorem in [20] answers how fast the lazy random walk based smoothing process converges to a stationary distribution.

Theorem 3. *Suppose that a strongly connected directed graph $\mathcal{G}$ on n nodes has Laplacian eigenvalues $0 = \lambda_0 \le \lambda_1 \cdots \lambda_{n-1}$. Then G has a lazy random walk with the rate of convergence of order $\frac{2}{\lambda_1}(-\log \min_v \pi(v))$. Namely, after at most $t \ge \frac{2}{\lambda_1}(-\log \min_v \pi(v) + 2c)$ steps, we have:*

$$\mathit{\Delta}(t) \triangleq \max_{1\le i\le n} \Big(\sum_{1\le j\le n} \frac{(P^{(t)}(v_i, v_j) - \pi(v_j))^2}{\pi(v_j)} \Big)^{\frac{1}{2}} \le e^{-c}.$$

Theorem 3 implies that it is difficult to prevent the over-smoothing of deep GAT by simply adding residual connections. This phenomenon has also been confirmed by experiments in [4].

E Multi-head Attention

Multi-head Attention is employed in GAT [1]. Specially, K independent attention heads are computed for feature aggregation at each layer, and the output of that layer is the concatenated outputs from all heads. To facilitate the analysis, we only focus on the attention aggregation and simplify Eq. 1 on each head as $\boldsymbol{X}^{(l,k)} = \boldsymbol{A}^{(l,k)}\boldsymbol{X}^{l-1}$, where $\boldsymbol{A}^{(l,k)}$ be the k-th ($1 \leq k \leq K_l$) head attention matrix in l-th layer of GAT, K_l is the head number of l-th layer. The output of l-th layer $\boldsymbol{X}^l = \|_{k=1}^{K_l}(\boldsymbol{A}^{(l,k)}\boldsymbol{X}^{l-1})$, where $\|$ denotes concatenation along the column (hidden) dimension. By expanding this equation for the previous layer, we can get that each independent component $\boldsymbol{A}^{(l,i)}\boldsymbol{X}^{l-1} = \boldsymbol{A}^{(l,i)} \|_{k=1}^{K_{l-1}} (\boldsymbol{A}^{(l-1,k)}\boldsymbol{X}^{l-2}) = \|_{k=1}^{K_{l-1}}(\boldsymbol{A}^{(l,i)}\boldsymbol{A}^{(l-1,k)}\boldsymbol{X}^{l-2})$. We can perform this expansion recursively for all layers. Therefore the output of l-th layer consists of multiple components, where each component can be viewed as a matrix-chain multiplication on l attention matrices from different heads and layers. According to Theorem 2, these matrix-chain multiplications will converge to the unique distribution π if $l \leftarrow \infty$. This means that multi-head attention GATs still suffer from over-smoothing problem if they go deep.

F Observation of Over-Smoothing on Data “Citeseer”

Fig. 2 shows the training loss, training error and the validation error of GAT models with different layers on benchmark dataset “Citeseer” (See detailed information of the data in Table 2). From this figure, we can observe that the deeper networks can still converge, but a performance degradation problem occurs: with the depth increasing, the accuracy degrades. In this paper, we demonstrate that such performance degradation is mainly due to over-smoothing effect of deeper GAT models.

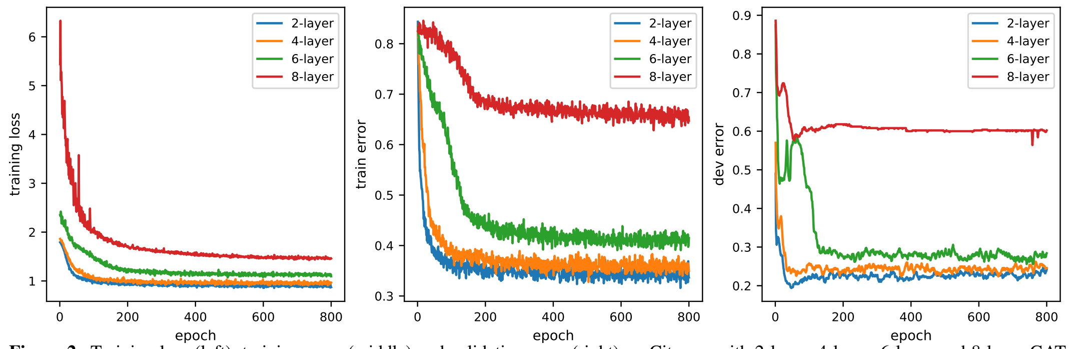

Figure 2: Training loss (left), training error (middle) and validation error (right) on Citeseer with 2-layer, 4-layer, 6-layer and 8-layer GAT models. The deeper network has higher training error, and thus validation error.

G Benchmark Information and Hyper-Parameter Settings

Data Set. Table 2 summarizes the statistical information of these datasets. Our experiments are conducted over standard data splits [21].

Table 2: Statistical Information on Benchmarks

Name	Nodes	Edges	Classes	Node features	Train/Dev/Test
Cora[a]	2708	5429	7	1433	1,208/500/1,000
Citeseer[a]	3327	4732	6	3703	1,827/500/1000
Pubmed[a]	19717	88651	3	500	18,217/500/1,000
PPI[b]	56944*	818716	121*	50	20/2/2$^\diamond$

a: transductive problem; b: inductive problem; $\star$: multi-label; $*$: total nodes in 24 graphs; $\diamond$: 20 graphs for train, 2 graphs for validation and 2 graphs for test.

Hyper-parameter Settings. For three transductive learning problems, we use two hidden layers with hidden dimension as 32 for Cora, 64 for Citeseer, and three hidden layers with hidden dimension 32 for Pubmed; we set the number of neighbors k used in feature aggregate function as 4 for Cora, Citerseer, and 8 for Pubmed. For the inductive learning problem PPI, we use three hidden layers with hidden dimension 128, and set k as 8. We make use of Adam as the optimizer and perform hyper-parameter search for all baselines and our method over the same validation set. The set of margin values (ζ_g, ζ_b) used in $(\mathcal{L}_g, \mathcal{L}_b)$ is $\{0.1, 0.2, 0.3, 0.5\}$ and the trade-off factor (λ_g, λ_b) of two losses is set as $\{1, 2, 5, 10\}$, learning rate is set as $\{0.001, 0.003, 0.005, 0.01\}$ and ℓ_2 regularization factor is set as $\{0.0001, 0.0005, 0.001\}$. We train all models using early stopping with a window size of 100.

H Proof of Proposition 2

Proof. Let's first review the feature aggregate function in GAT:.

$$\boldsymbol{h}_i^{(l+1)} = \delta(\sum_{j\in\mathcal{N}_i} \alpha_{i,j}^{(l)} \boldsymbol{W}^{(l)} \boldsymbol{h}_j^{(l)}) = \delta(\sum_{j=1}^{N} \hat{\alpha}_{i,j}^{(l)} \boldsymbol{W}^{(l)} \boldsymbol{h}_j^{(l)}), \tag{7}$$

where $\hat{\alpha_{i,j}} = \alpha_{i,j}$ if $j \in \mathcal{N}_i$, otherwise $\hat{\alpha_{i,j}} = 0$. We can view $\hat{\alpha_{i,j}}$ as the importance of v_j of v_i given the graph with features $\boldsymbol{H}^{(l)} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_N]^T$. We can rewrite it as a form of conditional probability $\hat{\alpha}_{i,j} = p(v_j|v_i, \mathcal{G}, \boldsymbol{H}^{(l)})$. If we define $q(v_i|v_1, v_2, \cdots, v_N)$ (denoted as $q(v_i)$ for simplification) as the probability of sampling v_i given all the nodes of the current layer, then we get $\hat{\alpha}_{i,j}^{(l)} = \frac{p(v_j|v_i)}{q(v_i)}$. Then, according to Bayes's formula, we can get $q(v_i|v_1, v_2, \cdots, v_N) \propto \sum_{j=1}^{N} \hat{\alpha}_{j,i}^{(l)} = \sum_{j=1}^{N} \alpha_{j,i}^{(l)}$. □

I Experimental Results of Robust Analysis and Deeper GAT

To evaluate the robustness of C-GAT, in particular, whether the induced attention function is robust to the graph structure, we conduct experiments by perturbing edges in "Cora" data. Fig. 3 presents the experimental results. From this figure, we can observe that:

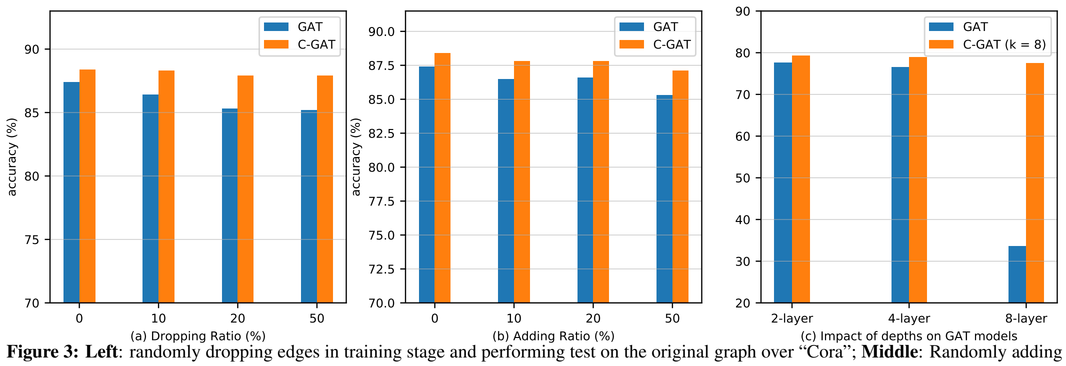

Figure 3: Left: randomly dropping edges in training stage and performing test on the original graph over "Cora"; **Middle**: Randomly adding edges in training stage and performing test on the original graph "Cora". For adding edges, we first randomly select a set of nodes according to a given sampling ratio, and then random add one edge on these nodes. **Right**: Classification performance comparison between GAT and C-GAT with different depth on "Citeseer"

(a) By randomly dropping some edges in training stage (see Fig. 3 (a)), C-GAT get a relative stable performance when increasing the ratio of dropped edge. In contrast, the performance of GAT shows a descending trend. This is because of that, for a missing edge in testing stage, the attention value w.r.t. this edge in C-GAT is still convincible as the two constraints. That is, if the missing edge connected two nodes share same labels, according to the constraints, the attention weight will be higher and results in a better smoothing operator. In contrast, if the missing edge connected two nodes with different labels, because of proposed constraints and proposed feature aggregation function, the impact of such edge can be eliminated as well. In contrast, for GAT without these constraints, there is still information propagation no matter the missing edge lies in classification boundary or not, and even assign large attention values for the classification boundary edges, and lead to over-smoothing.
(b) By random adding some edges in training stages (see Fig. 3 (b)),the performance of C-GAT still keeps relative stable but GAT's performance decreases when increasing the ratio of adding edges. This is because of that, the randomly adding edges might connect different classes together. This will result in more information propagation among different classes and easily lead to the over-smoothing. This hurts the quality of the training data. The constraints in C-GAT can be viewed as a data cleaner which can improve the quality of the training data. In contrast, GAT has no such ability and leads to the induced model perform worse in testing stage.
(c) Compares C-GAT and GAT with different depths on "Citeseer". Our proposed C-GATs maintain good classification performances with increasing attention layers. Again these results prove over-smoothing is not an issue for C-GAT.