Supervised Graph Attention Network for Semi-Supervised Node Classification

Dongkwan Kim
School of Computing
KAIST
dongkwan.kim@kaist.ac.kr

Alice Oh
School of Computing
KAIST
alice.oh@kaist.edu

Abstract

We propose supervised graph attention network (super-GAT), a novel neural network architecture for semi-supervised node classification in graphs. While learning (unsupervised) graph attention from the original graph attention network (GAT), we jointly train the attention values by supervised learning with information whether an edge exists between a pair of nodes. By giving supervision, we can assign not only implicit weights to nodes in the neighborhood but also explicit weights to nodes in any relation. Our model is based on GAT, and it only needs a few additional parameters and computation. We show how super-GAT performs on three transductive benchmark citation datasets: Cora, CiteSeer, and PubMed, and compared to baseline models including GAT, super-GAT achieves higher prediction accuracy for the first two datasets.

1 Introduction and Related Work

Deep neural architectures have shown excellent results in many fields [1, 2]. In particular, graph CNNs have shown overwhelming performance by generalizing the convolutional layers used in grid structures to graph structures in both spectral and non-spectral ways [3, 4, 5, 6, 7, 8, 9, 10].

One important model is the graph attention network (GAT) [11] which uses the attention mechanism shown to lead to a significant performance increase in various domains [12, 13]. As attention has been used in sequence-based tasks to help focus on relevant entities, graph attention captures which neighbors of a specific node are important in representing the node. In both sequence and graph learning, attention values are trained in an unsupervised manner.

Graph attention differs from conventional attention in that the model computes and learns the attention value only for the node pairs that are linked to each other. Veličković et al. [11] calls this masked attention. In addition to the explicit information that if node j is linked to node i, node j is important to node i than others, there is also implicit information that if node j is not linked to node i, j is not important to i. Using this implicit information, we make the attention value be predictive of the existence of an edge between two nodes. By doing so, our attention mechanism can encode structurally relevant elements.

We propose supervised graph attention networks (super-GAT) which takes advantage of the information about nodes that are not linked. Super-GAT gives supervision to graph attention with binary labels about the existence of an edge (0 or 1). Specifically, we use the attention value as input to predict the likelihood that an edge exists between a pair of nodes. This makes the graph's attention to learn the explicit importance of any node pair as well as the implicit importance of nodes in neighborhoods.

Our approach can be used to all graph-related tasks, but in this paper, we focus on the semi-supervised node classification task in the transductive setting. We conduct experiments on three real-world

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

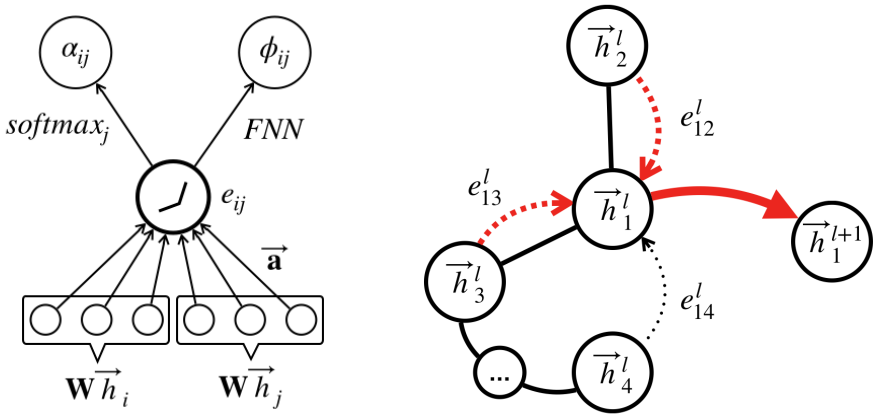

Figure 1: **Left:** The computation of normalized attention α_{ij} and the probability that an edge exists ϕ_{ij} in our model. We get $\vec{e}_{ij}$ and α_{ij} in the same way as GAT and calculate ϕ_{ij} by a small feed-forward neural network with e_{ij}. **Right:**. An illustration of (single-head) graph attention of node 1. Aggregation process is equal to GAT's, but attention between unlinked relations (e.g., node 1 and 4) are also computed for link prediction.

citation datasets (Cora, CiteSeer, and PubMed) and show that giving supervision to graph attention improves performance over GAT for Cora and CiteSeer.

2 Model

In this section, we describe the supervised graph attention layer and the design of the loss function.

First, we explain the original graph attention layer and its notations from [11]. For a given graph $G = (V, E)$, $N = |V|$ is the number of nodes and F^l is the number of features at layer l. Graph attention layer takes a set of features $\mathbf{h}^l = \{\vec{h}^l_1, \vec{h}^l_2, \ldots, \vec{h}^l_N\}, \vec{h}^l_i \in \mathbb{R}^{F^l}$ as an input and produces another set of features $\mathbf{h}^{l+1} = \{\vec{h}^{l+1}_1, \vec{h}^{l+1}_2, \ldots, \vec{h}^{l+1}_N\}, \vec{h}^{l+1}_i \in \mathbb{R}^{F^{l+1}}$. To compute the output feature $\vec{h}^{l+1}_i$, we multiply the weight matrix $\mathbf{W}^{l+1} \in \mathbb{R}^{F^{l+1} \times F^l}$ to each input feature, linearly combine the features of node i and its first-order neighbors $j \in \mathcal{N}_i$ by normalized self-attention coefficients α^{l+1}_{ij}, and finally apply a non-linear activation function σ. Formally, we get $\vec{h}^{l+1}_i = \sigma\left(\sum_{j \in \mathcal{N}_i \cup \{i\}} \alpha^{l+1}_{ij} \mathbf{W}^{l+1} \vec{h}^l_j\right)$. We can use various attention mechanisms suggested by prior research [14, 12, 15, 13], but in our experiments, we retain the original attention mechanism of GAT:

$$e^{l+1}_{ij} = \text{LeakyReLU}\left((\vec{\mathbf{a}}^{l+1})^\top \left[\mathbf{W}^{l+1}\vec{h}^l_i \| \mathbf{W}^{l+1}\vec{h}^l_j\right]\right), \; \alpha^{l+1}_{ij} = \frac{\exp\left(e^{l+1}_{ij}\right)}{\sum_{k \in \mathcal{N}_i \cup \{i\}} \exp\left(e^{l+1}_{ik}\right)}, \tag{1}$$

where $\vec{\mathbf{a}}^{l+1} \in \mathbb{R}^{2F^{l+1}}$ is weight parameters and $\|$ is the vector concatenation operation.

Now we introduce our model using the notations above. The key idea of super-GAT is guiding attention values with the existence or absence of an edge between a pair of nodes. However, if the number of nodes is large, it is not efficient to use all possible samples in the complementary set of edges, $E^c = (V \times V) \setminus E$. So, we use negative sampling as in training word or graph embeddings [16, 17, 18], arbitrarily choosing a total of $|E|$ negative samples E^- from E^c. Then, we apply feed-forward neural network f_e to attention coefficients e_{ij} to predict the existence of edges in $E \cup E^-$. Putting a sigmoid layer $\phi(\cdot)$ to $f_e(e_{ij})$, we predict the existence of an edge between node pair (j, i),

$$\phi_{ij} = \phi\left(f_e(e_{ij})\right) = P\left((j,i) \in E\right). \tag{2}$$

Our optimization objective of layer l is a binary cross-entropy loss $\mathcal{L}^l_E$ defined by

$$\mathcal{L}^l_E = \sum_{(j,i) \in E \cup E^-} \mathbf{1}_{(j,i)=0} \cdot \log\left(1 - \phi\left(f_e(e^l_{ij})\right)\right) + \mathbf{1}_{(j,i)=1} \cdot \log \phi\left(f_e(e^l_{ij})\right), \tag{3}$$

Table 1: Statistics of the Datasets: Cora, CiteSeer and PubMed.

	Cora	CiteSeer	PubMed
# Nodes	2708	3327	19717
# Edges	5429	4732	44338
# Features/Node	1433	3703	500
# Classes	7	6	3
# Training Nodes	140	120	60
# Validation Nodes	500	500	500
# Test Nodes	1000	1000	1000

where $\mathbf{1}_{\cdot}$ is an indicator function. To give the regularization effect from randomness, we use a subset of $E \cup E^{-}$ sampled by probability p_e at each training iteration. Also, we can simply employ K multi-head attentions to our method by using the mean of each head's loss value as an input of f_e.

$$f_e(e_{ij}^l) = \frac{1}{K}\sum_{k=1}^{K} f_e(e_{ij}^{l,(k)}), \quad \vec{h}_i^{l,(k)} = \sigma\left(\sum_{j\in\mathcal{N}_i\cup\{i\}} \alpha_{ij}^{l,(k)}\mathbf{W}^{l,(k)}\vec{h}_j^{l-1}\right). \tag{4}$$

Finally, we combine multi-class cross-entropy loss on node labels ($\mathcal{L}_V$), supervised graph attention losses for all L layers ($\mathcal{L}_E^l$), and L2 regularization loss, with hyperparameters λ_E, λ_2 respectively.

$$\mathcal{L} = \mathcal{L}_V + \lambda_E \cdot \sum_{l=1}^{L} \mathcal{L}_E^l + \lambda_2 \cdot \|\mathbf{W}\|_2. \tag{5}$$

Computational complexity of super-GAT and GAT The super-GAT has no difference in space and time complexity compared to GAT. Since e_{ij} is a scalar value, we only need two extra parameters (one for weight, one for bias) per layer for f_e. To compute $\mathcal{L}_E^l$ for a single attention head, we need additional operations in terms of $O(|E^{-}| \cdot F^{l+1} + |E|)$. Like GAT, space and time complexity is linear in the number of heads K, and computation of each head can be parallelized.

3 Experiments

Datasets We use three standard benchmark datasets for semi-supervised node classification tasks: Cora, CiteSeer, and PubMed [19, 20]. Each of these datasets consists of a single graph linked by document citations. For all datasets, we follow the transductive experimental setup and train/validation/test split of GAT [11]. We use 20 samples per class for training, 500 samples for validation, and 1000 samples for test. See Table 1 for details.

Experimental setup We follow most of the same experimental setup from GAT [11]. All parameters are initialized by Glorot initialization [21] and optimized by Adam [22] with a learning rate of 0.005 (Cora and CiteSeer) and 0.01 (PubMed). L2 regularization, dropout [23] to input features and attention coefficients, and early stopping on validation loss and accuracy are applied. For Cora and CiteSeer, we employ a two-layer super-GAT with eight attention heads for the first layer and one attention head for the second layer. We use exponential linear unit (ELU) [24] as a non-linear activation σ. For PubMed, we set the number of attention heads as eight for the second layer. Other settings are the same as architectures for Cora and CiteSeer. One setup that we cover but GAT does not is f_e. For this, we use a two-layer unit-width FNN, that is $f_e(x) = w_2 \cdot \sigma(w_1 \cdot x + b_1) + b_2$.

Baselines We compare our method against three baseline models. They are feed-forward neural network on node features (MLP), graph convolutional neural network (GCN) [6], and graph attention network (GAT) [11].

Hyperparameters There are at most four hyperparameters. We fix the dropout probability p_d to 0.6 and the edge sampling probability p_e to 0.9. Mixing coefficients of losses, λ_2 and λ_E, are tuned by Bayesian optimization for a set of random seeds (30 for Cora/CiteSeer and 20 for PubMed) with 100 random explorations and a total of 800 steps.

Implementation All models are implemented in PyTorch [25] and PyTorch Geometric [26].

Table 2: Summary of our model and baselines' accuracy for Cora, CiteSeer and PubMed.

Model	Cora	CiteSeer	PubMed
MLP	55.1%	46.5%	71.4%
GCN	81.5%	70.3%	**79.0%**
GAT	$83.0 \pm 0.7\%$	$72.5 \pm 0.7\%$	$\mathbf{79.0 \pm 0.3\%}$
Super-GAT (ours)	$\mathbf{83.4 \pm 0.6\%}$	$\mathbf{73.0 \pm 0.9\%}$	$78.5 \pm 0.4\%$

4 Results

We summarize the experimental results in Table 2. For all experiments, we measure the model performance with classification accuracy. We report the mean and standard deviation of accuracy after 100 runs and reuse the results introduced in prior work [6, 11].

Our model outperforms baselines for Cora and CiteSeer datasets. For PubMed, we get 78.5% which is slightly lower than GCN and GAT.

We also analyze the un-normalized attention value distributions of positive and negative edges. For Cora and PubMed datasets, we draw box-and-whisker plots with un-normalized attention values from the first layer e^1_{ij} of our model and GAT in figure 3. We attach the same plots for CiteSeer in the appendix. In these plots, we use the mean values of all attention heads and directions, in other words, $e^1_{\{i,j\}} = \frac{1}{2K}\sum_{k=1}^{K}(e^{1,(k)}_{ij} + e^{1,(k)}_{ji})$.

We make three observations. First, super-GAT shows larger differences in mean attention values between the positive and negative samples, compared to GAT. This is an expected result of our architecture design. Second, the attention values of super-GAT are more dispersed compared to those of GAT. We interpret this to mean that giving supervision to graph attention lets the model concentrate on a smaller number of nodes among the neighbors. Finally, for PubMed, there are more negative samples that are outliers than Cora and CiteSeer, regardless of the models. These outliers contradict our assumption that the attention value is related to the existence of edges. This might explain why our approach does not improve the performance on the PubMed dataset.

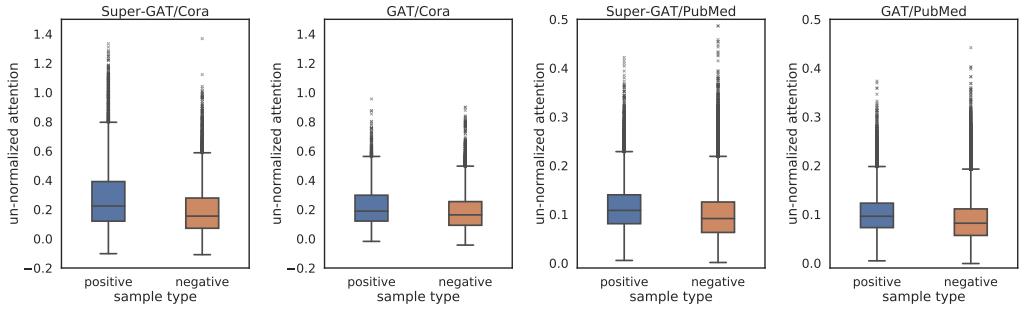

Figure 2: Box plots of un-normalized attention values from the first layer of super-GAT and GAT trained with Cora and PubMed datasets.

5 Conclusion and Future Work

We have introduced supervised graph attention networks, leveraging labels about edge existence to give supervision to graph attention. Super-GAT achieved better performance than GAT in the transductive problem through efficient training procedures without major architectural changes.

There are several ways to improve this research. Future work will cover 1) extending to node classification in the inductive setting, 2) giving supervision to self-attention graph pooling [27] for graph-level classification, and 3) examining the effectiveness of negative edge sampling guided by graph statistics (e.g., node distance).

Acknowledgments This research was supported by the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921).

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs, 2014.

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

[5] 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.

[6] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[7] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

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

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

[10] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124, 2017.

[11] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[12] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*, 2015.

[13] 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.

[14] John Boaz Lee, Ryan A Rossi, Sungchul Kim, Nesreen K Ahmed, and Eunyee Koh. Attention models in graphs: A survey. *arXiv preprint arXiv:1807.07984*, 2018.

[15] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015.

[16] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[17] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[18] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[19] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[20] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*, 2016.

[21] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[24] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations (ICLR)*, 2016.

[25] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

[26] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[27] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *Proceedings of the 36th International Conference on Machine Learning*, 09–15 Jun 2019.

A Box Plot about CiteSeer

As we described in the section 4, we attach un-normalized attention value distribution of positive and negative samples of CiteSeer in the appendix. See section 4 for the analysis.

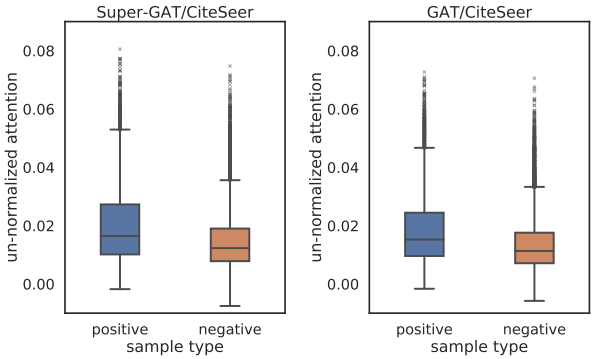

Figure 3: Box plots of un-normalized attention values from the first layer of super-GAT and GAT trained with CiteSeer datasets.