GRAPH ATTENTION NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We present graph attention networks (GATs), novel neural network architectures that operate on graph-structured data, leveraging masked self-attentional layers to address the shortcomings of prior methods based on graph convolutions or their approximations. By stacking layers in which nodes are able to attend over their neighborhoods' features, we enable (implicitly) specifying different weights to different nodes in a neighborhood, without requiring any kind of costly matrix operation (such as inversion) or depending on knowing the graph structure upfront. In this way, we address several key challenges of spectral-based graph neural networks simultaneously, and make our model readily applicable to inductive as well as transductive problems. Our GAT models have achieved state-of-the-art results across three well established transductive and inductive graph benchmarks: the *Cora* and *Citeseer* citation network datasets, as well as a *protein-protein interaction* dataset (wherein test graphs are entirely unseen during training).

1 INTRODUCTION

Convolutional Neural Networks (CNNs) have been successfully applied to tackle problems such as image classification (He et al., 2016), semantic segmentation (Jégou et al., 2017) or machine translation (Gehring et al., 2016), where the underlying data representation has a grid-like structure. These architectures efficiently reuse their local filters, with learnable parameters, by applying them to all the input positions.

However, many interesting tasks involve data that can not be represented in a grid-like structure and that instead lies in an irregular domain. This is the case of 3D meshes, social networks, telecommunication networks, biological networks or brain connectomes. Such data can usually be represented in the form of graphs.

There have been several attempts in the literature to extend neural networks to deal with arbitrarily structured graphs. Early work used recursive neural networks to process data represented in graph domains as directed acyclic graphs (Frasconi et al., 1998; Sperduti & Starita, 1997). Graph Neural Networks (GNNs) were introduced in Gori et al. (2005) and Scarselli et al. (2009) as a generalization of recursive neural networks that can directly deal with a more general class of graphs, e.g. cyclic, directed and undirected graphs. GNNs consist of an iterative process, which propagates the node states until equilibrium; followed by a neural network, which produces an output for each node based on its state. This idea was adopted and improved by Li et al. (2016), which propose to use gated recurrent units (Cho et al., 2014) in the propagation step.

Nevertheless, there is an increasing interest in generalizing convolutions to the graph domain. Advances in this direction are often categorized as spectral approaches and non-spectral approaches.

On one hand, spectral approaches work with a spectral representation of the graphs and have been successfully applied in the context of node classification. In Bruna et al. (2014), the convolution operation is defined in the Fourier domain by computing the eigendecomposition of the graph Laplacian, resulting in potentially intense computations and non-spatially localized filters. These issues were addressed by subsequent works. Henaff et al. (2015) introduced a parameterization of the spectral filters with smooth coefficients in order to make them spatially localized. Later, Defferrard et al. (2016) proposed to approximate the filters by means of a Chebyshev expansion of the graph Laplacian, removing the need to compute the eigenvectors of the Laplacian and yielding spatially localized filters. Finally, Kipf & Welling (2017) simplified the previous method by restricting the filters to operate in a 1-step neighborhood around each node. However, in all of the aforementioned

spectral approaches, the learned filters depend on the Laplacian eigenbasis, which depends on the graph structure. Thus, a model trained on a specific structure can not be directly applied to a graph with a different structure.

On the other hand, we have non-spectral approaches (Duvenaud et al., 2015; Atwood & Towsley, 2016; Hamilton et al., 2017), which define convolutions directly on the graph, operating on groups of spatially close neighbors. One of the challenges of these approaches is to define an operator which works with different sized neighborhoods and maintains the weight sharing property of CNNs. In some cases, this requires learning a specific weight matrix for each node degree (Duvenaud et al., 2015), using the powers of a transition matrix to define the neighborhood while learning weights for each input channel and neighborhood degree (Atwood & Towsley, 2016), or extracting and normalizing neighborhoods containing a fixed number of nodes (Niepert et al., 2016). More recently, Hamilton et al. (2017) introduced GraphSAGE, a method for computing node representations in an *inductive* manner. This technique operates by sampling a fixed-size neighborhood of each node, and then performing a specific aggregator over it (such as the mean over all the sampled neighbors' feature vectors, or the result of feeding them through a recurrent neural network). This approach has yielded impressive performance across several large-scale inductive benchmarks.

Attention mechanisms have become almost a *de facto* standard in many sequence-based tasks (Bahdanau et al., 2015; Gehring et al., 2016). One of the benefits of attention mechanisms is that they allow for dealing with variable sized inputs, focusing on the most relevant parts of the input to make decisions. When an attention mechanism is used to compute a representation of a single sequence, it is commonly referred to as *self-attention* or *intra-attention*. Together with Recurrent Neural Networks (RNNs) or convolutions, self-attention has proven to be useful for tasks such as machine reading (Cheng et al., 2016) and learning sentence representations (Lin et al., 2017). However, Vaswani et al. (2017) showed that not only *self-attention* can improve a method based on RNNs or convolutions, but also that it is sufficient for constructing a powerful model obtaining *state-of-the-art* performance on the machine translation task.

Inspired by this recent work, we introduce an attention-based architecture to perform node classification of graph-structured data. The idea is to compute the hidden representations of each node in the graph, by attending over its neighbors, following a *self-attention* strategy. The attention architecture has several interesting properties: (1) the operation is efficient, since it is parallelizable across node-neighbor pairs; (2) it can be applied to graph nodes having different degrees by specifying arbitrary weights to the neighbors; and (3) the model is directly applicable to *inductive* learning problems, including tasks where the model has to generalize to completely unseen graphs. Our approach of sharing a neural network computation across edges is reminiscent of the formulation of relational networks (Santoro et al., 2017), wherein relations between objects (regional features from an image extracted by a convolutional neural network) are aggregated across all object pairs, by employing a shared mechanism. We validate the proposed approach on three challenging benchmarks: *Cora* and *Citeseer* citation networks as well as an inductive *protein-protein interaction* dataset, achieving state-of-art results that highlight the potential of attention-based models when dealing with arbitrarily structured graphs.

2 GAT ARCHITECTURE

In this section, we will present the building block layer used to construct arbitrary graph attention networks (through stacking this layer), and directly outline its theoretical and practical benefits and limitations compared to prior work in the domain of neural graph processing.

2.1 GRAPH ATTENTIONAL LAYER

We will start by describing a single *graph attentional layer*, as the sole layer utilized throughout all of the GAT architectures used in our experiments. The particular attentional setup utilized by us closely follows the work of Bahdanau et al. (2015)—but the framework is agnostic to the particular choice of attention mechanism.

The input to our layer is a set of node features, $\mathbf{h} = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\}, \vec{h}_i \in \mathbb{R}^F$, where N is the number of nodes, and F is the number of features in each node. The layer produces a new set of node features (of potentially different cardinality F'), $\mathbf{h}' = \{\vec{h}'_1, \vec{h}'_2, \ldots, \vec{h}'_N\}, \vec{h}'_i \in \mathbb{R}^{F'}$, as its output.

In order to obtain sufficient expressive power to transform the input features into higher-level features, at least one learnable linear transformation is required. To that end, as an initial step, a shared linear transformation, parametrized by a *weight matrix*, $\mathbf{W} \in \mathbb{R}^{F' \times F}$, is applied to every node. We then perform *self-attention* on the nodes—a shared attentional mechanism $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$ computes *attention coefficients*

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j) \tag{1}$$

that indicate the *importance* of node j's features to node i. In its most general formulation, the model allows every node to attend on every other node, *dropping all structural information*. We inject the graph structure into the mechanism by performing *masked attention*—we only compute e_{ij} for nodes $j \in \mathcal{N}_i$, where $\mathcal{N}_i$ is some *neighborhood* of node i in the graph. In all our experiments, these will be exactly the first-order neighbors of i (including i). To make coefficients easily comparable across different nodes, we normalize them across all choices of j using the softmax function:

$$\alpha_{ij} = \mathrm{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}. \tag{2}$$

In our experiments, the attention mechanism a is a single-layer feedforward neural network, parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$ (illustrated by Figure 1 (left)). Fully expanded out, the coefficients computed by the attention mechanism may then be expressed as:

$$\alpha_{ij} = \frac{\exp\left(\vec{\mathbf{a}}^T [\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\vec{\mathbf{a}}^T [\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)} \tag{3}$$

where $\cdot^T$ represents transposition and $\|$ is the concatenation operation.

Once obtained, the normalized attention coefficients are used to compute a linear combination of the features corresponding to them, to serve as the final output features for every node (after potentially applying a nonlinearity, σ):

$$\vec{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}\vec{h}_j\right). \tag{4}$$

To stabilize the learning process of self-attention, we have found extending our mechanism to employ *multi-head attention* to be beneficial, similarly to Vaswani et al. (2017). Specifically, K independent attention mechanisms execute the transformation of Equation 4, and then their features are concatenated, resulting in the following output feature representation:

$$\vec{h}'_i = \mathop{\Big\|}_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right) \tag{5}$$

where $\|$ represents concatenation, α_{ij}^k are normalized attention coefficients computed by the k-th attention mechanism (a^k), and $\mathbf{W}^k$ is the corresponding input linear transformation's weight matrix. Note that, in this setting, the final returned output, $\mathbf{h}'$, will consist of KF' features (rather than F') for each node.

Specially, if we perform multi-head attention on the final (prediction) layer of the network, concatenation is no longer sensible—instead, we employ *averaging*, and delay applying the final nonlinearity (usually a softmax or logistic sigmoid for classification problems) until then:

$$\vec{h}'_i = \sigma\left(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right) \tag{6}$$

The aggregation process of a multi-head graph attentional layer is illustrated by Figure 1 (right).

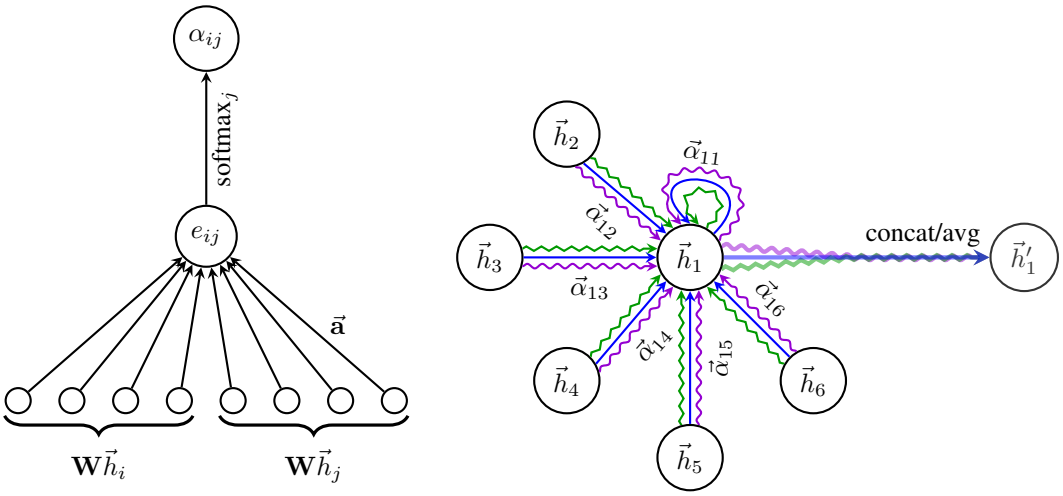

Figure 1: **Left:** The attention mechanism $a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$ employed by our model, parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$. **Right:** An illustration of multi-head attention (with $K = 3$ heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or average to obtain $\vec{h}'_1$.

2.2 Advantages and limitations

The graph attentional layer described in subsection 2.1 directly addresses several issues that were present in prior approaches to modelling graph-structured data with neural networks:

- Computationally, it is highly efficient: the operation of the self-attentional layer can be parallelized across all edges, and the computation of output features can be parallelized across all nodes. No eigendecompositions or similar costly matrix operations are required.
- As opposed to Graph Convolutional Networks (GCNs) (Kipf & Welling, 2017), our model allows for (implicitly) assigning *different importances* to nodes of a same neighborhood, enabling a leap in model capacity. Furthermore, analyzing the learned attentional weights may lead to benefits in model interpretability, as was the case in the machine translation domain (e.g. the qualitative analysis of Bahdanau et al. (2015)).
- The attention mechanism is applied in a shared manner to all edges in the graph, and therefore it does not depend on upfront access to the global graph structure or (features of) all of its nodes (a limitation of many prior techniques). This has several desirable implications:
  - The graph is not required to be undirected (we may simply leave out computing α_{ij} if edge $j \rightarrow i$ is not present).
  - It makes our technique directly applicable to *inductive* learning problems—including tasks where the model is evaluated on graphs that are *completely unseen* during training.
- The recently published inductive method of Hamilton et al. (2017) samples a *fixed-size neighborhood* of each node, in order to keep its computational footprint consistent; this does not allow it access to the entirety of the neighborhood while performing inference. Moreover, this technique achieved some of its strongest results when an LSTM (Hochreiter & Schmidhuber, 1997)-based neighborhood aggregator is used. This assumes the existence of a consistent sequential node ordering across neighborhoods, and the authors have rectified it by consistently feeding randomly-ordered sequences to the LSTM. Our technique does not suffer from either of these issues—it works with the entirety of the neighborhood, and does not assume any ordering within it.

Currently, the primary limitation of this model is of practical nature—even though the softmax activation used for computing the normalized activation coefficients (α_{ij}) is independently executed

Table 1: Summary of the datasets used in our experiments.

	Cora	**Citeseer**	**PPI**
Task	Transductive	Transductive	Inductive
# Nodes	2708 (1 graph)	3327 (1 graph)	56944 (24 graphs)
# Edges	5429	4732	818716
# Features/Node	1433	3703	50
# Classes	7	6	121 (multilabel)
# Training Nodes	140	120	44906 (20 graphs)
# Validation Nodes	500	500	6514 (2 graphs)
# Test Nodes	1000	1000	5524 (2 graphs)

across all nodes i, existing GPU-enabled tensor manipulation frameworks are only able to parallelize it efficiently for same-sized neighborhoods. This required us to use the method of Vaswani et al. (2017), wherein the attention is performed over all nodes, with masking applied by way of an additive bias of $-\infty$ to the masked entries' coefficients before applying the softmax. This leads to an $O(V^2)$ space complexity for storing intermediate results (where V is the number of nodes in the graph). Alternately, this requirement may be reduced to $O(D^2)$ (where D is the maximum node indegree in the graph) at the expense of performing a highly inefficient sequence of gather-partition-stitch operations.

This prevented us from evaluating our method on benchmarks with larger node counts, and is an important direction for future work. One potential avenue for addressing this is utilizing pointwise activation functions (such as the logistic sigmoid) to compute the α_{ij} values, which would allow for using efficient *sparse data representations* throughout the network. However, we have found that this approach yielded significant drops in predictive power across all our experiments.

3 Evaluation

We have performed comparative evaluation of GAT models against a wide variety of strong baselines and previous approaches, on three established graph-based benchmark tasks (transductive as well as inductive), achieving state-of-the-art performance across all of them. This section summarizes our experimental setup, results, and a brief qualitative analysis of a GAT model's extracted feature representations.

3.1 Datasets

Transductive learning We utilize two standard citation network benchmark datasets—Cora and Citeseer (Sen et al., 2008)—and closely follow the transductive experimental setup of Yang et al. (2016). In both of these datasets, nodes correspond to documents and edges to (undirected) citations. Node features correspond to elements of a bag-of-words representation of a document. Each node has a class label. We allow for only 20 nodes per class to be used for training—however, honoring the transductive setup, the training algorithm has access to all of the nodes' feature vectors. The predictive power of the trained models is evaluated on 1000 test nodes, and we use 500 additional nodes for validation purposes (the same ones as used by Kipf & Welling (2017)). The Cora dataset contains 2708 nodes, 5429 edges, 7 classes and 1433 features per node. The Citeseer dataset contains 3327 nodes, 4732 edges, 6 classes and 3703 features per node.

Inductive learning We make use of a protein-protein interaction (PPI) dataset that consists of graphs corresponding to different human tissues (Zitnik & Leskovec, 2017). The dataset contains 20 graphs for training, 2 for validation and 2 for testing. Critically, testing graphs remain *completely unobserved* during training. To construct the graphs, we used the preprocessed data provided by Hamilton et al. (2017). The average number of nodes per graph is 2372. Each node has 50 features that are composed of positional gene sets, motif gene sets and immunological signatures. There are 121 labels for each node set from gene ontology, collected from the Molecular Signatures Database (Subramanian et al., 2005), and a node can possess several labels simultaneously.

An overview of the interesting characteristics of the datasets is given in Table 1.

3.2 STATE-OF-THE-ART METHODS

Transductive learning For transductive learning tasks, we compare against the same strong baselines and state-of-the-art approaches as specified in Kipf & Welling (2017). This includes label propagation (LP) (Zhu et al., 2003), semi-supervised embedding (SemiEmb) (Weston et al., 2012), manifold regularization (ManiReg) (Belkin et al., 2006), skip-gram based graph embeddings (DeepWalk) (Perozzi et al., 2014), the iterative classification algorithm (ICA) (Lu & Getoor, 2003) and Planetoid (Yang et al., 2016). We also directly compare our model against GCNs (Kipf & Welling, 2017), as well as graph convolutional models utilising higher-order Chebyshev filters (Defferrard et al., 2016).

Inductive learning For the inductive learning task, we compare against the four different supervised GraphSAGE inductive methods presented in Hamilton et al. (2017). These provide a variety of approaches to aggregating features within a sampled neighborhood: GraphSAGE-GCN (which extends a graph convolution-style operation to the inductive setting), GraphSAGE-mean (taking the elementwise mean value of feature vectors), GraphSAGE-LSTM (aggregating by feeding the neighborhood features into an LSTM) and GraphSAGE-pool (taking the elementwise maximization operation of feature vectors transformed by a shared nonlinear multilayer perceptron). The other transductive approaches are either completely inappropriate in an inductive setting or assume that nodes are incrementally added to a single graph, making them unusable for the setup where test graphs are completely unseen during training (such as the PPI dataset).

Additionally, for both tasks we provide the performance of a per-node shared multilayer perceptron (MLP) classifier (that does not incorporate graph structure at all).

3.3 EXPERIMENTAL SETUP

Transductive learning For the transductive learning tasks, we apply a two-layer GAT model. Its architectural hyperparameters have been optimized on the Cora dataset and are then reused for Citeseer. The first layer consists of $K = 8$ attention heads computing $F' = 8$ features each (for a total of 64 features), followed by an exponential linear unit (ELU) (Clevert et al., 2016) nonlinearity. The second layer is used for classification: a single attention head that computes C features (where C is the number of classes), followed by a softmax activation. We did not find multi-head attention helpful for the output layer in these experiments, and attribute this as potentially caused by small training set sizes. For coping with the small training set sizes, regularization is liberally applied within the model. During training, we apply L_2 regularization with $\lambda = 0.0005$. Furthermore, dropout (Srivastava et al., 2014) with $p = 0.5$ is applied to both layers' inputs, as well as *to the normalized attention coefficients* (critically, this means that at each training iteration, each node is exposed to a stochastically sampled neighborhood).

Inductive learning For the inductive learning task, we apply a three-layer GAT model. Both of the first two layers consist of $K = 4$ attention heads computing $F' = 256$ features (for a total of 1024 features), followed by an ELU nonlinearity. The final layer is used for (multi-label) classification: $K = 4$ attention heads computing 121 features each, that are averaged and followed by a logistic sigmoid activation. The training sets for this task are sufficiently large and we found no need to apply L_2 regularization or dropout—we have, however, successfully employed skip connections (He et al., 2016) across the intermediate attentional layer. We utilize a batch size of 2 graphs during training.

Both models are initialized using Glorot initialization (Glorot & Bengio, 2010) and trained to minimize cross-entropy on the training nodes using the Adam SGD optimizer (Kingma & Ba, 2014) with an initial learning rate of 0.005. In both cases we use an early stopping strategy on the accuracy (transductive) or micro-F_1 (inductive) score on the validation nodes, with a patience of 100 epochs[1].

[1]The code to reproduce the experiments can be found here: XXXXXXXXXX

Table 2: Summary of results in terms of classification accuracies, for the Cora and Citeseer datasets.

Transductive		
Method	**Cora**	**Citeseer**
MLP	55.1%	46.5%
ManiReg (Belkin et al., 2006)	59.5%	60.1%
SemiEmb (Weston et al., 2012)	59.0%	59.6%
LP (Zhu et al., 2003)	68.0%	45.3%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%
ICA (Lu & Getoor, 2003)	75.1%	69.1%
Planetoid (Yang et al., 2016)	75.7%	64.7%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%
GCN (Kipf & Welling, 2017)	81.5%	70.3%
GAT (ours)	**83.3%**	**74.0%**
improvement w.r.t GCN	1.8%	3.7%

Table 3: Summary of results in terms of micro-averaged F_1 scores, for the PPI dataset.

Inductive	
Method	**PPI**
Random	0.396
MLP	0.422
GraphSAGE-GCN (Hamilton et al., 2017)	0.500
GraphSAGE-mean (Hamilton et al., 2017)	0.598
GraphSAGE-LSTM (Hamilton et al., 2017)	0.612
GraphSAGE-pool (Hamilton et al., 2017)	0.600
GAT (ours)	**0.936**
improvement w.r.t GraphSAGE	32.4%

3.4 RESULTS

The results of our comparative evaluation experiments are summarized in Tables 2 and 3.

For the transductive tasks, we report the classification accuracy on the test nodes of our method, and reuse the metrics already reported in Kipf & Welling (2017) for state-of-the-art techniques. Specifically, for the Chebyshev filter-based approach (Defferrard et al., 2016), we provide the maximum reported performance for filters of orders $K = 2$ and $K = 3$.

For the inductive task, we report the micro-averaged F_1 score on the nodes of the two unseen test graphs, and reuse the metrics already reported in Hamilton et al. (2017) for the other techniques. Specifically, as our setup is supervised, we compare against the supervised GraphSAGE approaches.

Our results successfully demonstrate state-of-the-art performance across all three datasets—in concordance with our expectations, as per the discussion in Section 2.2. More specifically, we are able to improve upon GCNs by a margin of 1.8% and 3.7% on Cora and Citeseer, respectively, suggesting that assigning different weights to nodes of a same neighborhood may be beneficial. It is worth noting the improvement achieved on PPI dataset (32.4% w.r.t. GraphSAGE, without significant hyperparameter tuning), highlighting the potential of the method to be applied in inductive settings.

The effectiveness of the learned feature representations may also be investigated qualitatively—and for this purpose we provide a visualization of the t-SNE (Maaten & Hinton, 2008)-transformed feature representations extracted by the first layer of a GAT model pre-trained on the Cora dataset (Figure 2). The representation exhibits discernible clustering in the projected 2D space. Note that these clusters correspond to the seven dataset's labels, verifying the model's discriminative power across the seven topic classes of Cora. Additionally, we visualize the relative strengths of the nor-

malized attention coefficients (averaged across all eight attention heads). Properly interpreting these coefficients (as performed by e.g. Bahdanau et al. (2015)) will require further domain knowledge about the dataset under study, and is left for future work.

4 CONCLUSIONS

We have presented graph attention networks (GATs), novel convolution-style neural networks that operate on graph-structured data, leveraging masked self-attentional layers. The graph attentional layer utilized throughout these networks is computationally efficient (does not require costly matrix operations, and is parallelizable across all nodes in the graph), allows for (implicitly) assigning different importances to different nodes within a neighborhood while dealing with different sized neighborhoods, and does not depend on knowing the entire graph structure upfront—thus addressing many of the theoretical issues with previous spectral-based approaches. Our models leveraging attention have successfully achieved state-of-the-art performance across three well established node classification benchmarks, both transductive and inductive (especially, with completely unseen graphs used for testing).

There are several potential improvements and extensions to graph attention networks that could be addressed as future work, such as overcoming the practical problems described in subsection 2.2 to be able to handle larger benchmarks. A particularly interesting research direction would be taking advantage of the attention mechanism to perform a thorough analysis on the model interpretability. Moreover, extending the method to perform graph classification instead of node classification would also be relevant from the application perspective. Finally, extending the model to incorporate edge features (possibly indicating relationship among nodes) would allow us to tackle a larger variety of problems.

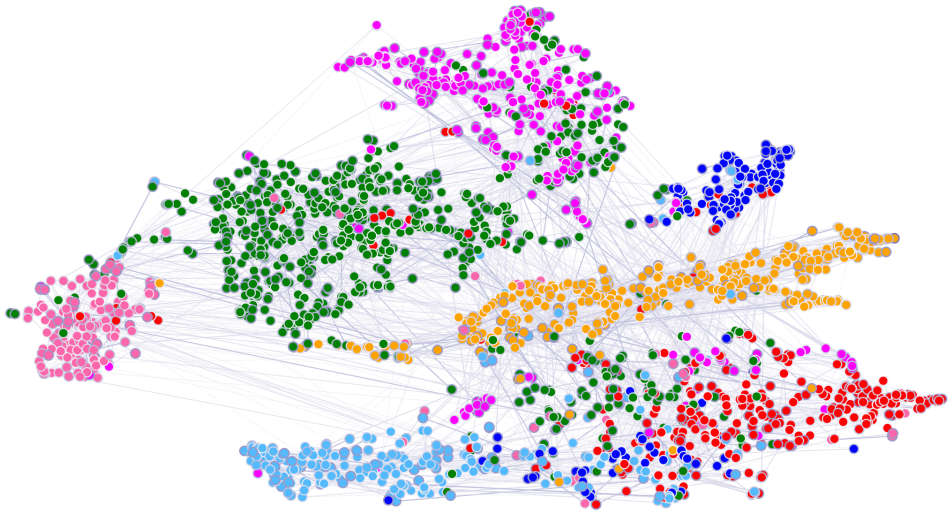

Figure 2: A t-SNE plot of the computed feature representations of a pre-trained GAT model's first hidden layer on the Cora dataset. Node colors denote classes. Edge thickness indicates aggregated normalized attention coefficients between nodes i and j, across all eight attention heads $(\sum_{k=1}^{K} \alpha_{ij}^{k} + \alpha_{ji}^{k})$.

REFERENCES

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

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

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7 (Nov):2399–2434, 2006.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *International Conference on Learning Representations (ICLR)*, 2014.

Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

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

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016.

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*, pp. 2224–2232, 2015.

Paolo Frasconi, Marco Gori, and Alessandro Sperduti. A general framework for adaptive processing of data structures. *IEEE transactions on Neural Networks*, 9(5):768–786, 1998.

Jonas Gehring, Michael Auli, David Grangier, and Yann N. Dauphin. A convolutional encoder model for neural machine translation. *CoRR*, abs/1611.02344, 2016. URL `http://arxiv.org/abs/1611.02344`.

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*, pp. 249–256, 2010.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *IEEE International Joint Conference on Neural Networks*, pp. 729734, 2005.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Neural Information Processing Systems (NIPS)*, 2017.

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

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

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Simon Jégou, Michal Drozdzal, David Vázquez, Adriana Romero, and Yoshua Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Workshop on Computer Vision in Vehicle Technology CVPRW*, 2017.

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

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

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *International Conference on Learning Representations (ICLR)*, 2016.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

Qing Lu and Lise Getoor. Link-based classification. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 496–503, 2003.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pp. 2014–2023, 2016.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710. ACM, 2014.

Adam Santoro, David Raposo, David GT Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *arXiv preprint arXiv:1706.01427*, 2017.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

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

A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *Trans. Neur. Netw.*, 8(3):714–735, May 1997. ISSN 1045-9227. doi: 10.1109/72.572108.

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

Aravind Subramanian, Pablo Tamayo, Vamsi K Mootha, Sayan Mukherjee, Benjamin L Ebert, Michael A Gillette, Amanda Paulovich, Scott L Pomeroy, Todd R Golub, Eric S Lander, et al. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences*, 102(43):15545–15550, 2005.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pp. 639–655. Springer, 2012.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning*, pp. 40–48, 2016.

Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pp. 912–919, 2003.

Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017.