Exploration of Graph Attention Networks for Graph Neural Networks

Cheremy Pongajow[1 [2647002]]

Vrije University of Amsterdam, De Boelelaan 1081 HV, NL ai@vu.nl
https://vu.nl/en

Abstract. In this paper, we present the implementation of various graph neural networks (GNNs) and attention-based graph neural networks. GNNShave shown promising results in various graph-related tasks such as node classification, link prediction, and graph classification. However, traditional GNNs treat all nodes equally and do not consider the importance of different nodes in the graph. Attention mechanisms provide a solution to this problem by allowing GNNs to focus on relevant nodes during message passing. In this paper, we explore different attention mechanisms, including graph attention networks (GAT, GATv2), to enhance the performance of GNNs on graph-based tasks. We evaluate the proposed models on random split benchmark datasets and compare their performance with traditional GNNs. Our experimental results demonstrate that attention-based GNNs achieve similar in terms of accuracy. Furthermore, we provide an in-depth analysis of the learned attention weights to gain insights into the importance of different nodes in the graph.

Keywords: Graph Attention Networks · Graph Neural Networks · Attention Mechanism

1 Introduction

Graphs are found everywhere in our daily lives, such as in social networks, transportation networks, scientific publication networks, and molecular structures. Graphs offer a multitude of information due to their expressive power. Graph neural networks (GNNs) aim to learn a level of representation from the graph in a lower dimensional representation space to apply in downstream tasks such as node classification, link prediction and query and answering. GNNs show impressive performance on these downstream tasks and have now been vastly adopted in the graph domain. Research on graphs adopted more inspiration from deep learning by applying graph convolutional neural networks (GCNs). Convolutional neural networks have shown tremendous performance in the domain of image processing. Kipf et al. have found promising results in applying convolutional techniques to graphs, although graphs have a more sparse structure compared to the regular grid structure of images [**?**]. Although the GCNs perform well, they, have limitations regarding fixed neighborhood aggregation [**?**]. In

essence, this means that the aggregation function is applied uniformly across all nodes, ignoring the varying importance of the nodes. To address this limitation of assigning equal node importance Inspired by the self-attention mechanism, Veličković et al. (2018) developed the first graph attention network (GAT) to address some of the limitations of GCNs [**?**] [**?**]. The introduction of the self-attention mechanism in the model allows a learnable parameter to assign varying attention coefficients to the nodes which indicates the importance of each node. Recently more advances have been made by improving GAT with a subsequent adjustment called GATv2. GATv2 uses a dynamic form of attention over the static form that is used in the original GAT [**?**][**?**]. This limitation is addressed by modifying the order of operations.

In a recent study, the shortcomings of the evaluation of graph models come to the forefront [**?**]. This study highlighted that the empirical outcomes of graph benchmark datasets are highly dependent on the selected train/validation/test split. These findings suggest that the current literature may not be providing a fair and consistent basis for drawing comparisons between GNN architectures. This dependency raises concerns about current comparisons of GNN architectures in the current literature. The need for fairness and consistency becomes apparent to ensure insightful and unbiased assessment of GNN architectures. Therefore, this work aims to contribute to the evaluations and comparison of GAT and GATv2 by extending it with a fair benchmark comparison. Thereby, addressing the challenges in current benchmark evaluations and providing a meaningful understanding of the performance between GNNs on node classification.

2 Background

2.1 Graph Neural Networks

GNNs are popular architectures for downstream graph applications. The representation of these graphs can be represented as $G = (V, E)$, where V is the number of nodes and E represents the number of edges. The adjacency matrix represents the connections between the nodes where it is defined as $A \in R^{NxN}$ (N: number of nodes). Where each A_{ij} represent the undirected edge between node A_{ij} with 1 the presence of an edge and 0 the absence. A feature or node embedding is described as x_i for every node i and the feature matrix is defined as $X \in R^{NxF}$ (F: number of features). A simple GNN can be represented as:

$$f(H^{(l)}, A) = \sigma(AH^{(l)W^{(l)}}) \tag{1}$$

Here $W^{(l)}$ represents a learnable weight matrix of the l-th layer in the network. The σ is a non-linear activation function like a sigmoid or ReLU. A limitation of this model stems from non-normalized multiplication. By multiplying the adjacency matrix with the diagonal node degree matrix, this limitation gets addressed [**?**]. The normalization of the nodes needs to be symmetrical for the rows and columns, which gives $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$. This normalisation ensures that the

eigenvalues are bounded to a range between 0 and 1, which prevents the numerical instabilities and exploding gradients [**?**]. Combing the normalization with the previous equation results in the following:

$$f(H^{(l)}, A) = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{2}$$

Note that $\hat{A}$, is A added identity matrix I to include self-loops in the adjacency matrix. The addition of self-loops is commonly used as it allows for nodes to account for their features. Deriving the equation into vector form results in the following equation:

$$h(l+1)_v i = \sigma(\sum_{j=1}^{c_i} h(l)_v j * W(l)) \tag{3}$$

Here, the summation term shows the aggregation of the features of the neighbouring nodes of v_i. The coefficient of c_{ij} are originated form the symmetrical normalized adjacency matrix $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$. As the equation shows it enables smooth aggregation of node embedding and prevents the numerical instabilities and exploding gradients. Each node contributes equally to the update of the normalized feature embeddings. However, this equal contribution may not be a desirable trait in the model as it can't account for variable node importance. Here is where the self-attention mechanism comes in. The self-attention mechanism was introduced in natural language processing (NLP) [**?**]. The self-attention mechanism addresses the same problem in NLP as it adaptively assigns varying attention coefficients to words in a sentence based on their relevance. These attention coefficients are learnable parameters and allow the model to focus on relevant words. The original self-attention is denoted as follows:

2.2 Scaled Dot-Product Attention

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{4}$$

Here, the Q, K, V represents the query, key and values matrices. A softmax is used with a scaling of the dimensions of the key vectors determining the amount of attention each key should attribute. The dividing of the dimensions helps in stabilizing the gradients. This self-attention mechanism allows the model to selectively assign varying importance to words in a sentence or text. This method has been proven effective in NLP and has been adopted in graph machine-learning models. In essence, these methods introduce the same concept applied to graphs.

2.3 Graph Attention Networks

GATs are a popular GNN architecture [**?**]. GATs work by computing a weighted average of the representations, achieved through a learned scoring mechanism. This scoring function assigns scores to each edge in the graph by providing an

importance score of the nodes to one another. The initial equation computes the attention coefficients in GATs are denoted as follows:

$$e_{ij} = a(\mathbf{W}\tilde{\mathbf{h}}_i, \mathbf{W}\tilde{\mathbf{h}}_j) \tag{5}$$

Where e_{ij} are the attention coefficients which represent the importance of the features of the neighbour node j to the node i. Note that the computation is only done for $j \in N_i$, where N_i is the neighbourhood of the node [**?**]. Subsequently, a softmax function is introduced for obtaining the normalized attention coefficients:

$$\alpha_{ij} = \frac{\exp(LeakyReLU(\tilde{\mathbf{a}}^T[\mathbf{W}\tilde{\mathbf{h}}_i \,||\, \mathbf{W}\tilde{\mathbf{h}}_j]))}{\sum_{k\in\mathcal{N}_i} \exp(LeakyReLU(\tilde{\mathbf{a}}^T[\mathbf{W}\tilde{\mathbf{h}}_i \,||\, \mathbf{W}\tilde{\mathbf{h}}_k]))} \tag{6}$$

This shows a single-layer attention mechanism with the addition of a non-linearity of the LeakyReLU [**?**]. Here, ^{T} represents the transposition and $||$ a concatenation of matrices. The normalized attention coefficients are determined by the compatibility of node features, represented by the dot product with a learnable weight vector a. This allows for adaptive weighting of different neighbours based on their importance to the node [**?**].

2.4 Multi-head Attention

The final step in the attention mechanism is the aggregation of multiple attention layers. By using K independent attention mechanism with concatenation results in the following equation:

$$\mathbf{h}'_i = \sigma\left(\Big\|_{k=1}^{K} \sum_{j\in\mathcal{N}_i} \alpha_{kij}\mathbf{W}_k\tilde{\mathbf{h}}_j\right) \tag{7}$$

This equation depicts the aggregation of the multiple attention heads resulting in the representation $h_j^{'}$. The multi-head attention stabilizes the learning process and benefits performance [**?**]. In the final layer of the multi-head attention concatenation of the heads isn't sensible [**?**] [**?**]. Therefore, an averaging is applied and the non-linearity is delayed in the order of operations.

$$\mathbf{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i} \alpha_{kij}\mathbf{W}_k\tilde{\mathbf{h}}_j\right) \tag{8}$$

The graph attention later in graphs introduced by Veličković et al. (2018) addresses several issues that were present in prior works [**?**]. It is computationally more efficient as it allows for parallel computing of the attention heads. In addition, it allows the model to adaptively assign importance to nodes. Furthermore, other than fixed-size neighbourhood methods it works on the entirety of the neighbourhood. However, a limitation of the GAT is that it is unable to leverage sparse matrix operations, which results in exponential memory complexity.

2.5 GAT vs GATv2

The GAT faces some problems as it utilises static attention over dynamic attention. The static form of attention restricts the model from capturing node importance across different queries.

$$GAT: \quad e(\mathbf{h}_i, \mathbf{h}_j) = LeakyReLU(\mathbf{a}^T \cdot [\mathbf{W}\mathbf{h}_i \,||\, \mathbf{W}\mathbf{h}_j]) \tag{9}$$

$$GATv2: \quad e(\mathbf{h}_i, \mathbf{h}_j) = \mathbf{a}^T \cdot LeakyReLU(\mathbf{W} \cdot [\mathbf{h}_i \,||\, \mathbf{h}_j]) \tag{10}$$

Here, the GATv2 equation modifies the order of operations to enable the layers to compute dynamic attention. By applying the learnable attention matrix differently after the non-linearity it allows to adapt to different query-key relations.

3 Evaluation

3.1 Datasets

The CORA dataset is a transductive benchmark dataset in the graph machine learning domain. The dataset contains citations from research in various domains, such as computer science, physics, and biology. Each node in the graph represents an individual paper with undirected edges representing the citation relationship between the paper. The CORA dataset contains a total of 2,708 nodes, which can be classified into one of the seven classes. The features of the nodes contain bag-of-words representations of the title and abstract. Besides the CORA dataset, the CiteSeer data is also utilized. The CiteSeer dataset is similar to the CORA dataset. It is composed of nodes representing individual papers as well. Similar to CORA, the CiteSeer dataset also has undirected edges indicating the relationship between the papers. However, the CiterSeer dataset has a bigger graph with six classes. The feature of the CiteSeer nodes is also a bag-of-words representation from the title and abstract.

We split the dataset into training, validation, and test sets, following the standard protocol used in the literature. This allows us to train our models on a subset of the data, tune hyperparameters using the validation set, and evaluate the final performance on the test set. The Cora dataset provides a challenging and realistic scenario for evaluating the effectiveness of attention mechanisms in graph neural networks.

3.2 Models

This work aims to explore the train/test/val split on the comparison of GAT mechanisms and other baselines. Therefore, we take the GAT with fixed architecture (which can be regarded as having the same aggregation and feature transformation ability) as the base model and compare the discrimination ability on graph with different attention mechanisms. We focus more on the difference

caused by attention distribution learned by GATs, thus choosing not to tune the architecture of GATs through careful optimization and parameter tuning. In this paper, we implement two GAT models The baseline models and GATs and their fixed architecture are as follows:

- GNN: A multi-layer feedforward network [**?**].
- GCN: A graph convolutional network that aggregates the feature of neighbouring nodes and concatenates it with the central node's feature. We set the layer of GCN to 2 [**?**].
- GAT: The originally proposed Graph Attention Network. We set the layer of GAT to 2 and multi-head number to 8 [**?**].
- GATv2: A dynamic graph attention variant that can learn dynamic attention by simply switching the order of internal operations in GAT. We set the layer of GATv2 to 2 and the multi-head number to 8 [**?**].

3.3 Experiment

In the setup of the experiments, we keep the original architecture implemented in the papers. Also, the number of attention heads is fixed and not a hyperparameter. All previously listed models are evaluated on the CORA and CiteSeer datasets. For a fair evaluation, the training procedure for all the models is equal, including the same Adam optimizer with Xavier initialization. This helps in minimizing any potential bias that could occur from this. However, instead of the split provided by the Planetoid framework, an evaluation is employed over 10 train/validation/test splits [**?**]. This k-fold cross-validation, procedures a more reliable insight into the performance of the current models. This addresses the overfitting and generalizability of the models highlights in shortcomings of the evaluation of graph machine learning models [**?**]. Lastly, the metric used for evaluation was the mean test accuracy.

3.4 Results

Table **??** shows the mean accuracy and standard deviation of all models for the CORA and CiteSeer datasets averaged over 20 splits and 20 random initializations for each split. The sizes of the split were 20 per class; validation set: 500 nodes; and test set: 1000 same as in the Planetoid framework. The hyperparameters have been optimized on the Cora dataset. The first layer consists of K = 8 attention heads computing F = 8 features each, followed by an exponential linear unit (ELU) for non-linearity. The second layer is used for classification: a single attention head that computes seven features with a softmax activation. During training, we apply L2 regularization with λ 0.0005 also the dropout is set at 0.6 for both layers.

Planetoid Split	Cora	CiteSeer
GNN	0.569	0.599
GCN	0.802	0.682
GAT	0.68	0.639
GATv2	0.757	0.612

Table 1. Performance comparison of different graph neural network models on Cora and CiteSeer datasets. Results are presented with the mean accuracy.

The results of the experiments with the regular planetoid split on the CORA and CiteSeer showed comparable results as [**?**]. Among the implemented GNN approaches the GCNs had the best performance on the planetoid. This is contrary to other results shown in the literature [**?**][**?**]. Table **??** mean test set accuracy and standard deviation in percent averaged over 10 random train/validation/test splits with 10 random weight initializations for all models and all datasets.

Random Split	CORA	CiteSeer
GNN	0.544 ± 0.03	0.527 ± 0.03
GCN	0.79 ± 0.01	0.658 ± 0.02
GAT	0.68 ± 1.3	0.64 ± 0.02
GATv2	0.75.7 ± 1.3	0.635 ± 0.03

Table 2. Performance comparison of different graph neural network models on Cora and CiteSeer datasets. Results are mean test set accuracy and ± standard deviation over 20 random train/validation/test.

A noteworthy finding is that the GCN is able to achieve the best performance across all models. While this result seems surprising, similar to other findings that have been reported in the literature [**?**]. The unexpected performance of GCN across both CORA and CiteSeer datasets raises intriguing questions and challenges conventional wisdom. The prevailing literature, as referenced in [**?**] and [**?**], suggested alternative s (that GATs and its variants outperform traditional GCNs on various datasets. These discrepancies of outcomes highlight the importance and need for reproducibility in the graph machine learning. However, it must be noted that the outcomes of the experiment are only limited to few datasets. But the results highlight the sensitivity of GNNs to random factors, such as weight initializations and dataset splits. This emphasizes the need for multiple runs and careful consideration of experimental setups to draw robust conclusions.

4 Conclusion

This work showed an evaluation of various GNN architectures on the node classification task. This work enabled a fair and reproducible comparison of different

GNN models. The results highlight the fragility of experimental setups that consider only a single train/validation/test split of the data. The results also surprisingly show that a GCN model can outperform the more sophisticated GAT and GATv2 architectures when results are averaged over multiple data splits. This work shows that improved evaluations are necessary for transductive learning tests as the split highly influences the results and robustness of the evaluations.