Exploring Performance of Graph Attention Network and Residual Graph Attention Network on Citation Networks: A Reproducibility Study

Jahnvi Patel
jpate201@illinois.edu

Group ID: 207
Paper ID: 7
Presentation link: https://youtu.be/fA7TFirfpSk
Code link: https://github.com/janv10/CS598_Project

1 Introduction

The ability to effectively process and learn from graph-structured data has become increasingly important in modern data science, as graphs are ubiquitous in many domains. Graph Neural Networks (GNNs) have been developed for this purpose and are particularly useful for data that cannot be represented in grid-like structures, such as images or audio signals (Zhang et al., 2022). Among the various types of GNNs, Graph Attention Networks (GATs) have gained popularity in graph-based machine learning, especially in citation networks, which are graph-structured data representing publications and citation relationships between them. "Graph Attention Networks" (Veličković et al., 2018) was the first paper to propose an efficient neural network architecture for graph data that could learn node representations by capturing important dependencies between nodes. The GAT architecture achieves this by selectively focusing on different parts of the graph using attention mechanisms, allowing the model to identify the nodes that are most crucial for predicting the target node's label or property. This approach has made the GAT architecture a well-regarded and widely-used model for graph-based deep learning, with state-of-the-art performance on several benchmark tasks.

In healthcare, graph-based machine learning methodologies like GATs have the potential to revolutionize analysis and decision-making on complex medical data, with particular applications in citation networks related to medical research. To improve GAT's performance, ResidualGAT, a GAT model variant that addresses the vanishing gradient issue by incorporating residual connections, has been introduced (He et al., 2015). Residual connections enable the model to learn residual functions that show the difference between the input and output of a layer, making it easier for the model to identify the underlying data features. Studies have demonstrated that Residual models have produced better results on various graph datasets, including citation networks (**?**). By comparing the performance of ResidualGAT and the standard GAT model, we can gain insights into the advantages of incorporating residual connections into the GAT architecture.

2 Scope of reproducibility

This reproducibility paper aims to assess the performance and robustness of the GAT model on three commonly used citation networks: *Cora*, *CiteSeer*, and *PubMed*, as reported in the original study by (Veličković et al., 2018). In addition, we will evaluate the ResidualGAT model's performance on these citation networks, a variant of the GAT model that incorporates residual connections to mitigate the vanishing gradient problem and improve model performance (Bresson and Laurent, 2017). Our hypothesis is that the ResidualGAT model will outperform the standard GAT model on these datasets.

This study will conduct a thorough analysis of the GAT model's performance across the citation networks, comparing it with the ResidualGAT model. We will adjust hyperparameters such as the number of layers, attention heads, and dropout rates to systematically evaluate their impact on GAT's performance and generalization across the citation networks. By doing so, we aim to gain a deeper understanding of how architectural modifications affect their performance and efficacy and optimize these hyperparameters to find the optimal balance between model complexity and generalization performance.

The insights gained from this evaluation will contribute to potential enhancements in GAT and provide further guidance for future research in graph-based machine learning methodologies. Our goal is to gain a better understanding of the GAT and ResidualGAT models and their applications in citation network classification tasks, exploring their robustness and generalization to different citation networks and hyperparameter configurations.

2.1 Addressed claims from the original paper

1. Preprocess the datasets (*Cora*, *CiteSeer*, and *PubMed*) by converting text features into a suitable representation and creating adjacency matrices and feature matrices for the graphs.

2. Implement the GAT model by referencing the original paper by (Veličković et al., 2018). This involves specifying the same number of graph convolutional layers, attention heads, and activation functions as in the paper. Add hyperparameters such as learning rate, weight decay, number of epochs, batch size, and early stopping criteria for training the GAT model.

3. Ablation study: Implement the ResidualGAT model on the datasets, following a similar procedure to the GAT model. Add residual connections in the first and second GAT layers to enable the model to learn residual functions that represent the difference between the input and output of each layer. Tweak hyperparameters, such as learning rate, weight decay, number of epochs, batch size, and early stopping criteria, to optimize the performance of the ResidualGAT model.

4. Train and evaluate the performance of the GAT and ResidualGAT models on the datasets. Use performance metrics such as accuracy, precision, recall, and F1-score to gather results.

5. Visualize the results using plots and other visualization techniques for better interpretation.

3 Methodology

This reproducibility study aims to implement the GAT model from the paper (Veličković et al., 2018) and evaluate its performance on three citation network datasets (*Cora*, *CiteSeer*, and *PubMed*) against the ResidualGAT model to experiment for improved performance. We use the PyTorch deep learning framework for implementation and heavily rely on the original GAT paper for guidance in incorporating key components such as multi-head attention and feed-forward networks. The code provided in the original GAT paper (Veličković et al., 2018) is utilized for loading and preprocessing the datasets into suitable representations for model input. We create adjacency and feature matrices and split the datasets into training, validation, and testing sets. We then preprocess all three datasets on the GAT model.

We follow the specifications from the original paper by (Veličković et al., 2018) and use the same number of layers, attention heads, and activation functions in each layer for both the GAT and ResidualGAT models. Hyperparameters for the GAT model are also derived from the original paper to be as accurate to the original as possible. For ResidualGAT, we use the same hyperparameters as the GAT model, but they are further tuned for improved performance on the datasets. To mitigate overfitting during training, we apply early stopping and dropout regularization in both models. The GAT and ResidualGAT models are trained on the training set using the optimization algorithm Adam. Model performance is evaluated on the validation set using metrics such as accuracy, precision, recall, and F1-score. Finally, once both models are tested on the testing set, their performance metrics are reported.

A sensitivity analysis is conducted to evaluate the impact of hyperparameters and model configurations on ResidualGAT performance and is being compared to the GAT model. The data is visualized using plots to explore the structure and relationships within the citation network datasets and evaluate the training and validation processes of GAT and ResidualGAT.

In this reproducibility study, we have utilized our own computing resources, specifically using CPUs, to train and evaluate the models. For details on the computational resources used in this study, please refer to Section 3.5.

3.1 Model descriptions

The GAT model is a neural network designed to classify nodes on graphs by learning an embedding for each node in the input graph that captures its structural information and predicts its class label. The model includes two GATConv layers, each with a different number of heads. The first layer takes the node feature matrix and edge index matrix of the graph as input, and applies an attention mechanism to aggregate information from neighboring nodes, producing a new node feature matrix. The second layer takes the output of the first layer and produces the final node embeddings used for classification. By using multiple heads in each layer, the model can learn multiple attention mechanisms, each focusing on a different aspect of the graph's structure.

In this implementation, the number of trainable parameters for the GAT model is determined by multiplying 96 times the number of input features, 64 times the number of output classes, and 64 times the number of attention heads. For instance, if the dataset has 100 input features and 10 output classes and the GAT model has 4 attention heads, then the number of trainable parameters is (96 * 100) + (64 * 10) + (64 * 4) = 11,840.

The ResidualGAT model extends the GAT model by incorporating residual connections between the GATConv layers. The residual connections aim to improve the model's ability to capture long-range dependencies and avoid the vanishing gradient problem. The ResidualGAT model includes two pairs of GATConv layers, each consisting of a standard GATConv layer and a residual GATConv layer. The first pair of layers takes the node feature matrix and edge index matrix of the graph as input, and produces a new node feature matrix through an attention mechanism. The residual GATConv layer takes the original node feature matrix as input and produces a residual node feature matrix that is added to the output of the standard GATConv layer to form the final node embedding.

The second pair of GATConv layers takes the output of the first pair and produces the final node embeddings used for classification. The residual connections are applied again between the second pair of layers to further improve the model's performance. The number of trainable parameters for the ResidualGAT model is similar to the GAT model but with additional parameters from the layer normalization module. The trainable parameter count is (96 * num_features) + (64 * num_classes) + (64 * num_heads) + (16 * num_heads), where num_features, num_classes, and num_heads are the number of input features, output classes, and attention heads, respectively. For example, if the dataset has 100 input features and 10 output classes, and the ResidualGAT model has 4 attention heads, then the number of trainable parameters is (96 * 100) + (64 * 10) + (64 * 4) + (16 * 4) = 12,256.

3.2 Data descriptions

In this paper, we utilize three popular citation network datasets: *Cora*, *CiteSeer*, and *PubMed*, which contain graph-structured data representing scientific publications and their citation relationships. The datasets are available through the library *Planetoid*.

The *Cora* dataset is a citation network consisting of 2,708 scientific publications classified into one of seven classes. The citation network is constructed from the citations between the papers, and each paper is represented as a bag-of-words feature vector.

The *CiteSeer* dataset is another citation network, consisting of 3,327 scientific publications classified into six classes. Like *Cora*, the citation network is constructed from the citations between the papers, and each paper is represented as a bag-of-words feature vector.

The *PubMed* dataset is a citation network consisting of 19,717 scientific publications from the medical field, classified into one of three classes. Like the other two datasets, the citation network is constructed from the citations between the papers, but each paper is represented using a binary bag-of-words feature vector, indicating the presence or absence of each word in the paper's abstract.

3.3 Hyperparameters

The hyperparameters used in the provided code for the GAT and ResidualGAT models include the number of attention heads in each GATConv layer, the dropout rate, the weight decay, and the learning rate. In the GAT model, there are 8 attention heads in the first GATConv layer and 1 attention head in the second layer. The dropout rate is set to 0.6 for both the node features and the attention coefficients. The learning rate is set to 0.005 with a weight decay of 0.0005. For the ResidualGAT models, the number of attention heads varies from 6 to 16 in the first layer and 1 to 8 in the second layer. The dropout rate for the node features is set to 0.8 and 0.7 for the attention coefficients, with the exception of the third model which has a 0.2 dropout rate for attention coefficients. The learning rate ranges from 0.001 to 0.005 with a weight decay of 6e-4 to 0.001.

To ensure the best performance of our GAT model on the training and validation data, we derived the hyperparameters from the original GAT paper (Veličković et al., 2018). In ResidualGAT, the hyperparameters were chosen based on their effectiveness in previous studies and experimentation via the original paper (Veličković et al., 2018) as well as trial and error. The number of attention heads in the GAT and ResidualGAT models affects the model's ability to learn different aspects of the graph structure, with more heads allowing for greater granularity in attention mechanisms. The dropout rate helps prevent overfitting by randomly dropping out nodes and attention coefficients during training. The learning rate and weight decay control the speed and regularization of the optimization process.

Additionally, the number of attention heads and dropout rate were varied to achieve the best validation accuracy, while the learning rate and weight decay were adjusted to optimize the training process. The patience parameter was set to stop training if the validation accuracy did not improve after a certain number of epochs. Overall, the hyperparameters were chosen to balance model performance and training stability on the given datasets.

3.4 Implementation

To implement the GAT model, we have developed our own version of the code based on the original paper (Veličković et al., 2018), while using the original code as a reference for both the implementation and hyperparameter tuning. Interested readers can find the original code on GitHub through the provided link here.

As for the ResidualGAT implementation, we have heavily relied on the GAT model implementation and mainly adjusted the hyperparameters to explore the potential benefits of Residual GAT in enhancing the performance of the GAT model.

3.5 Computational Requirements

While training on a GPU can substantially reduce the training time on the *Cora*, *CiteSeer*, and *PubMed* datasets, it is worth mentioning that a standard CPU equipped with at least 16 GB of memory has effectively trained the models with minimal layers and attention heads. Hence, a GPU is not necessary for these datasets and is not used. Moreover, to avoid computational overload, we trained the models for only 200 epochs, compared to the original paper (Veličković et al., 2018) where they trained the models for 10,000 epochs.

4 Results

Here are the results of GAT and ResidualGAT on *Cora* dataset,*CiteSeer*, and *PubMed* datasets.

4.1 *Cora* Dataset

On the *Cora* dataset (Figure 1), we observed a substantial improvement in performance for both the GAT and ResidualGAT models after training. Prior to training, the GAT model had an F1 score of 07.45%, which significantly increased to 99.29% for both the training and validation sets post-training. Additionally, the accuracy, precision, and recall scores also showed a remarkable improvement, indicating that the GAT model learned the task well.

Similarly, for the ResidualGAT model, the F1 score before training was 10.50%, which considerably increased to 99.29% for the training set and 78.76% for the validation set post-training. The accuracy, precision, and recall scores also showed a significant improvement for both the training and validation sets. However, the lower F1 score for the validation set suggests that the ResidualGAT model may be overfitting the training data.

Overall, both models exhibited impressive improvements in performance after the training, with the GAT model surpassing the ResidualGAT model on the validation set.

4.2 *CiteSeer* Dataset

The GAT network was evaluated on the *CiteSeer* dataset (Figure 2), with an initial F1 score of 14.21%. After training for 150 epochs with early stopping, the F1 scores for both the training and validation sets improved, reaching 99.17% and 70.60%, respectively. The training accuracy was 92.50% and the validation accuracy was 71.60%. However, the precision, recall, and F1 score were all higher for the training set, suggesting that the model may have overfit the training data.

For the ResidualGAT network, the F1 score before training was 16.13%. The network was trained for 200 epochs, and the F1 score increased over time, reaching 65.78% at the end of training. The training accuracy was 99.17%, while the validation accuracy was 68.80%. The precision, recall, and F1 score were all higher for the training set than for the validation set, indicating some overfitting.

Comparing the two models, we can see that the GAT network achieved higher validation accuracy and F1 score than the ResidualGAT network, indi-

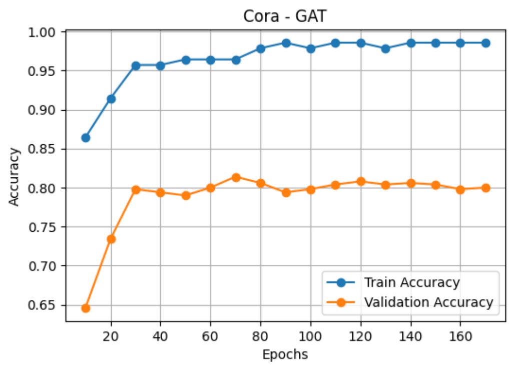

Figure 1: GAT and ResidualGAT on the *Cora* dataset

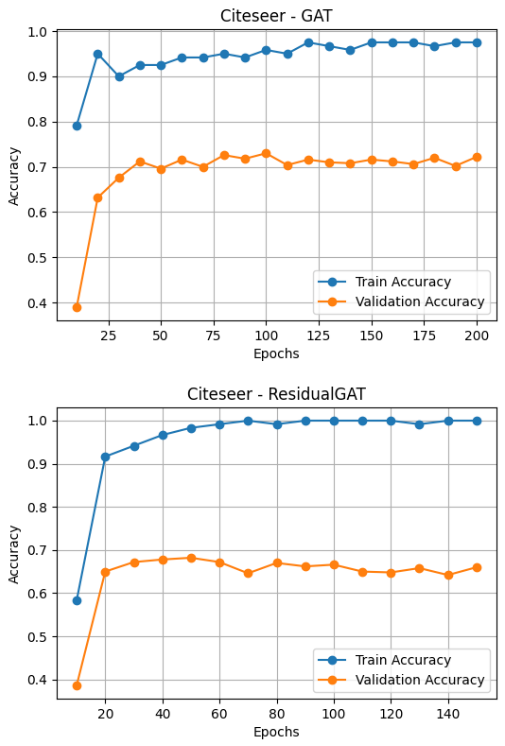

Figure 2: GAT and ResidualGAT on the *CiteSeer* dataset

cating that it was more successful in generalizing to unseen data. However, both networks exhibited some overfitting, with higher performance on the training set than on the validation set. Overall, further tuning and regularization may be necessary to improve the performance of both networks.

4.3 *PubMed* Dataset

For the GAT model on the *PubMed* dataset (Figure 3), the initial F1 score before training was 0.2502. However, the model showed significant improvement after training, with a final F1 score of 98.33% on the training set and 80.39% on the validation set. Notably, the F1 scores for the training and validation sets were very close, indicating that the model is not overfitting. Additionally, the model exhibited strong performance across all metrics, with accuracy, precision, and recall values all above 79%.

For the ResidualGAT model, the F1 score before training was 0.3249, and again, it improved significantly after training, reaching a final F1 score of 1.0 on the training set and 79.41% on the validation set. The training F1 score is perfect, which suggests that the model might be overfitting the training data. However, the validation F1 score is close to the training score, indicating that the model is still performing well on unseen data. The other metrics (accuracy, precision, and recall) are also good, with all values above 79%.

Comparing the two models, the ResidualGAT model performs slightly better on the training set, with a perfect F1 score, while the GAT model has a lower but still excellent F1 score of 98.33%. However, on the validation set, the GAT model performs slightly better with an F1 score of 80.39% compared to 79.41% for the ResidualGAT model. Therefore, the GAT model might be a better choice for this dataset.

4.4 Additional results not present in the original paper

We went beyond the original paper by training ResidualGAT on the datasets and compared the results with the GAT model. The results support the claims made in the original paper that GAT is a suitable model for citation network datasets. Though the claim that ResidualGAT is superior is not satisfactorily supported. Further optimization and experimentation may be required to improve performance.

5 Discussion

In theory, ResidualGAT should have the potential to perform better than the standard GAT in certain

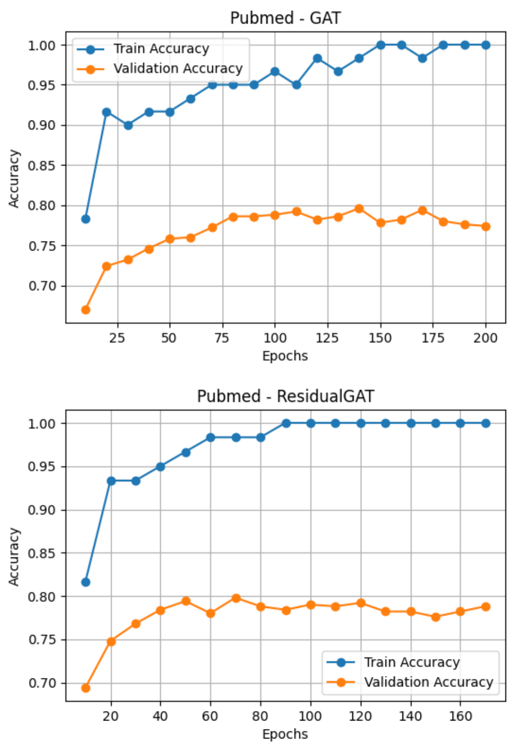

Figure 3: GAT and ResidualGAT on the *PubMed* dataset

scenarios, particularly when dealing with deeper architectures. The residual connections in ResidualGAT are designed to help with these problems by allowing gradients to flow more easily through the network.

The experimental results demonstrate that both GAT and ResidualGAT models are effective for citation network datasets. The GAT model achieved a significant improvement in performance on all datasets, outperforming ResidualGAT on the validation set. The ResidualGAT model showed a slight improvement on the training set for the *PubMed* dataset, but both models still exhibited some overfitting, even with early stopping. Although ResidualGAT was expected to perform better in scenarios with deeper architectures, the results did not fully support this claim. However, further investigation is necessary, such as performing ablation studies on different model components.

The reproducibility study successfully reproduced the results of the original paper (Veličković et al., 2018) with minor differences. The difference in hyperparameter tuning methods could explain the slight variations in performance, such as the limitation of epochs due to computational availability.

In conclusion, the experimental results provide further evidence of the effectiveness of GAT and ResidualGAT models on citation network datasets. However, further tuning and regularization are necessary to improve the models' generalization and reduce overfitting.

5.1 5.1 What was easy

Overall, reproducing the experiments in the original paper (Veličković et al., 2018) was relatively straightforward. The authors provided detailed descriptions of their methods, and the code was well-documented and easy to follow. Additionally, the datasets used in the experiments were publicly available, and the authors provided instructions on how to download and preprocess them.

5.2 5.2 What was difficult

The most difficult part of this study was comprehending the hyperparameter tuning process in the ResidualGAT models. Achieving this demanded a thorough comprehension of graph convolutional networks and considerable experimentation with hyperparameters. The original paper (Veličković et al., 2018) was not very clear on the hyperparameter tuning process, and we had to rely on trial and error to adjust hyperparameters, which was time-consuming.

5.3 Recommendations for reproducibility

To enhance the reproducibility of experiments utilizing GAT and ResidualGAT models on citation network datasets, we propose conducting hyperparameter testing on each dataset using a range of hyperparameters. Additionally, we recommend providing detailed information on the methods employed to evaluate hyperparameter tuning in future studies. This information is necessary for others to replicate the results and comprehend the impact of various hyperparameters on the model's performance.

6 Communication with the original authors

As part of our reproducibility study, we accessed the code and datasets used in the original paper online, without reaching out to the authors for additional information. We found these resources instrumental in replicating the original results and believe they were sufficient for a fair and accurate assessment of the reproducibility study.

References

Xavier Bresson and Thomas Laurent. 2017. Residual gated graph convnets. *CoRR*, abs/1711.07553.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks.

Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2022. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):249–270.