Project Report for CPSC 583

Gonghan Xu

1 Introduction

1.1 Problem Statement

In the realm of Graph Neural Networks (GNNs), node classification remains one of the cornerstone tasks. Typically, GNN architectures focus on leveraging local graph structures, where nodes aggregate information from their neighboring nodes. Graph Attention Networks (GATs), for instance, employ an attention mechanism to weigh the importance of each neighbor during this aggregation. On the other hand, the Transformer architecture, originally designed for sequence data, have been useful for capturing complex relation within data through self-attention mechanisms. So the primary question driving this project is: Can the integration of GAT with a Transformer block lead to enhanced performance in node classification tasks, specifically in terms of accuracy?

1.2 Method Overview

Our approach addresses this question by integrating Graph Attention Networks (GATs) with Transformer blocks in different ways and comparing their performance with standalone GATs and graph Transformers. We want to know if a novel hybrid architecture can leverage the strengths of both GATs and Transformers to further improve the performance on node classification.

1.3 Experimental Overview and Summary of Results

We conducted experiments on two benchmark datasets, Cora and Citeseer, to evaluate our approach. Eight model variants were tested, which include standalone GAT and graph Transformer models, and models combining GAT and Transformer layers in different configurations.

The results of our experiments suggest that integrating GATs with post-aggregation transformer blocks does not help. However, mixing GAT layers and for-aggregation Graph Transformer layers can be helpful on certain node classificaiton tasks. The project implementation can be accessed on: `https://github.com/kiwi1236/cpsc583_colab3`.

2 Description of Related Works

Graph-based learning has been a focal point of research with many significant contributions over recent years. Herein, we outline some of the pivotal works that have shaped the field and provide a contextual landscape for our proposed methodology.

Yang et al.[1] introduced the Planetoid, a semi-supervised learning method for graph-based classification. The model's primary strength lies in its capability to learn from both labeled and unlabeled data and it provides the first classification benchmark on the Cora and Citeseer datasets. The Planetoid model achieved an accuracy of 75.7% on the Cora dataset and 64.7% on Citeseer.

Kipf and Welling[2] presented the Graph Convolutional Network (GCN), a seminal work that simplifies the training process of deep networks on graph data. The GCN leverages a localized first-order approximation of spectral graph convolutions to operate directly on graph structures. It showcased an impressive accuracy of 81.5% on Cora and 70.3% on Citeseer.

Veličković et al.[3] introduced the Graph Attention Network (GAT), which incorporates attention mechanisms into graph neural networks. By assigning different attention scores to different nodes, GAT offers a more adaptive aggregation method compared to its predecessors. The model achieved notable accuracies of 83.0% on Cora and 72.5% on Citeseer.

Yun and collaborators[4] proposed the Graph Transformer Networks (GTN) that integrate the self-attention mechanism of Transformers directly within the GNN's message-passing process. This deep integration captures both local and global node dependencies.

The previous works on GNNs, as mentioned above, mostly focus on refining the aggregation step with more and more complex aggregation mechanisms. While it is good to see performance enhancement, this also brings the negative aspects of complicating the model implementation and increasing the computational cost, particularly with graph Transformer layers. In contrast, it's noteworthy that our proposed methodology distinguishes itself by attempting to incorporate the Transformer block post-aggregation, rather than as a part of the aggregation itself. Incorporating the Transformer block directly with global attention (rather than involving manipulation with the adjacency matrix) provides simplicity of implementation and less computational cost, so it is rewarding to test how much performance improvement it will bring compared to the costly Transformer-based aggregation. For comparative purpose, we will also try out architectures that implement Transformer-based aggregation to tell the difference in performance.

3 Method

Our methodology focuses on exploring the integration of Graph Attention Networks (GATs) with different types of Transformer blocks (i.e. post-aggregation Transformer block vs. Transformer-based aggregation block) and test the effect of performance classification accuracy on the Cora and Citeseer datasets. Our approach aims to combine the local aggregation power of GATs with the capability of Transformers to capturing higher-order dependencies among node features.

Our baseline model is based on the Graph Attention Network (GAT) architecture. In our implementation, the GAT model closely follows the design presented by Veličković et al.[3], because we want to use their GAT implementation as a benchmark and see whether their reported GAT performance is recoverable.

One of the novel aspects of our methodology, as mentioned before, is the addition of a post-aggregation Transformer block to the GAT framework. The aggregated node features from the GAT layers are treated as sequences in the Transformer block, allowing the model to potentially capture higher-order dependencies among node features, which might not be possible with the GAT model alone. Another novel aspect is that we will conduct a careful comparative study over 8 variants of the GAT+Transformer architecture, by comparing which we will be able to tell which part of the architecture is contributing to the performance improvement or not.

3.1 Model Variants

To thoroughly evaluate the effectiveness of the post-aggregation Transformer, we developed and tested several model variants (see Table 4). For simplicity, we use GATL to denote a (single-hop) GAT layer, TRANS to denote a vanilla Transformer layer (no neighbor aggregation involved), TRANS-A to denote a graph transformer layer[5] that is used for aggregating neighboring nodes, and LIN to denote a plain linear layer for node classification: Each model variant represents a unique combination of GAT and Transformer elements, thereby allowing us to analyze the effects of different architectural integrations on node classification accuracy.

4 Experiments

4.1 Datasets

The Cora and Citeseer datasets are a well-known benchmark datasets (Table 2) in the graph learning community. They comprise scientific publications, where each node represents a document and edges signify citations between these documents. The node features are based on word vectors, indicating

Table 1: Summary of Models

Model	**Architecture**
Model-0	GATL + GATL
Model-1	GATL + GATL + GATL
Model-2	GATL + TRANS + GATL
Model-3	GATL + GATL + TRANS + LIN
Model-4	TRANS-A + TRANS-A
Model-5	GATL + TRANS-A + GATL
Model-6	GATL + GATL + TRANS-A
Model-7	TRANS-A + TRANS-A + TRANS-A

the presence (or absence) of specific words in the document. Documents are classified into distinct classes (7 classes for Cora and 6 classes for Citeseer) based on their content.

Table 2: Statistics for the Datasets

(a) Cora

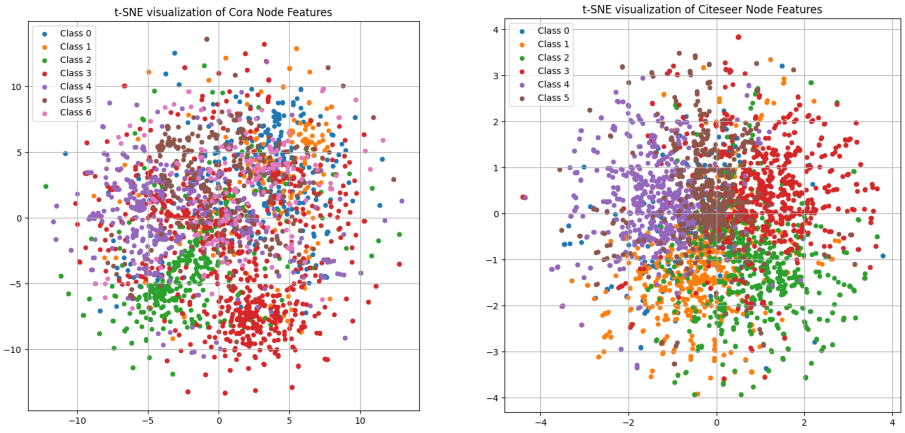
Statistic	Value
Number of Features	1433
Number of Classes	7
Number of Nodes	2708
Number of Edges	5278
Average Node Degree	3.90
Is Graph Directed	False
Number of Training Nodes	140
Number of Validation Nodes	500
Number of Test Nodes	1000

(b) Citeseer

Statistic	Value
Number of Features	3703
Number of Classes	6
Number of Nodes	3327
Number of Edges	4552
Average Node Degree	2.74
Is Graph Directed	False
Number of Training Nodes	120
Number of Validation Nodes	500
Number of Test Nodes	1000

The t-SNE visualization for the two datasets (Figure 1) display a diverse distribution of data points across different classes. There's noticeable clustering among certain classes, suggesting some similarity in feature space. However, significant overlap across the classes also highlights the intricacy of the datasets and the potential challenge of distinguishing between them based solely on node features.

Figure 1: Visualization of the two datasets with t-SNE dimensionality reduction



(a) Cora

(b) Citeseer

4.2 Training Details

Our experimental setup involved training each model variant on the Cora and Citeseer datasets. We used PyTorch and PyTorch-Geometric for implementation and training. The hyper-parameters that we used followed the original GAT design[3] closely. Specifically, We used cross-entropy loss with L2 regularization with $\lambda = 0.001$, a dropout rate of 0.4 on the input and after all activations (ELUs following the GAT layers and ReLUs within the transformer layers). We used the Adam optimizer with a learning rate of 0.005. For training, we set a maximum number of 1000 epochs with early stopping on accuracy score with patience of 100 epochs. Within GAT and Transformer layers (either TRANS OR TRANS-A), we use 8 attention heads and an output embedding dimension of 64, except that if a layer is used as the output layer, then the output embedding dimension becomes the number of classes. For data preprocessing, we used the standard train-validation-test splits provided with the datasets. For each model, we conducted 10 runs and collected the mean accuracy and the standard deviation.

4.3 Results

The results of our experiments are summarized in the following tables, presenting the mean accuracy and standard deviation for each model on the Cora and Citeseer datasets.

Table 3: Experimental Results on Cora and Citeseer Datasets

Model	**Cora (Mean Accuracy ± Std)**	**Citeseer (Mean Accuracy ± Std)**
Model-0	0.7934 ± 0.0113	0.6828 ± 0.0056
Model-1	0.7957 ± 0.0097	0.6799 ± 0.0086
Model-2	0.7923 ± 0.0144	0.6774 ± 0.0129
Model-3	0.7969 ± 0.0097	0.6740 ± 0.0151
Model-4	0.7899 ± 0.0047	0.6835 ± 0.0082
Model-5	0.7984 ± 0.0132	0.6788 ± 0.0141
Model-6	**0.8091 ± 0.0096**	0.6871 ± 0.0107
Model-7	0.8001 ± 0.0133	**0.6875 ± 0.0079**

4.4 Comparison with Baselines

The performance of the original GAT model as reported by Veličković et al[3] achieved accuracies of 83.0% on Cora and 72.5% on Citeseer. In comparison, our implementation of GAT (Model-0), despite following the same experimental setup as the [3], showed lower performance (79.3% on Cora and 68.3% on Citeseer). This difference highlights the sensitivity of GNNs to implementation and training nuances. To evaluate the effectiveness of the post-aggregation Transformer block, we use the performance of our 2-layer GAT (Model-0) as the baseline. Then from Table 3 and Figure 2, we can see that the other models' performance lies closely around this baseline despite variations in the model architectures. The best-performing model on the Cora dataset is Model-6, which uses a GATL + GATL + TRANS-A architecture. The best-performing model on the Citeser dataset is Model-7, which is a three-layer graph transformer. Note that Model-6 is also running up close on the Citeseer dataset.

4.5 Analysis of Results

We will conduct comparative studies (see Figure 2) to understand the impact of each component in our model variants. By comparing performances between Model-0 (two-layer GAT) with Model-1 (three-layer GAT), we can see that having extra GAT layers on top of the original two-layer GAT does little help. Comparing Model-2 and Model-3 (both based on post-aggregation transformer blocks) together with Model-0, we can see that adding a post-aggregation transformer block does little change to the performance of the original GAT design either. Comparing Model-0 (two-layer GAT) and Model-4 (two-layer TRANS-A only), we can see that a pure two-layer graph transformer does not pose advantage to a pure two-layer GAT on the two data sets. Comparing both Model-05 (GATL + TRANS-A + GATL) and Model-6 (GATL + GAT + TRANS-A) together with Model-0 (two-layer GAT), we can see that mixing GAT and TRANS-A layers do help. And it is particularly helpful to use TRANS-A as the last layer for classification. Comparing Model-7 (three-layer TRANS-A) and

Model-1 (three-layer GAT), we can see that when there are more layers, using pure graph transformer may perform slightly but not decisively better than a pure GAT.

Table 4: Summary of Models (Repeated)

Model	**Architecture**
Model-0	GATL + GATL
Model-1	GATL + GATL + GATL
Model-2	GATL + TRANS + GATL
Model-3	GATL + GATL + TRANS + LIN
Model-4	TRANS-A + TRANS-A
Model-5	GATL + TRANS-A + GATL
Model-6	GATL + GATL + TRANS-A
Model-7	TRANS-A + TRANS-A + TRANS-A

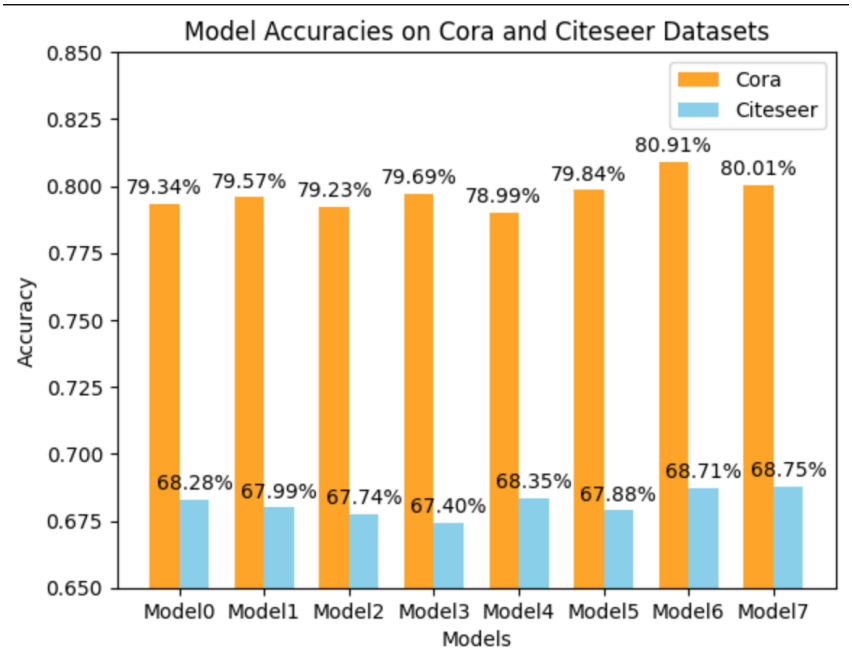


Figure 2: Results on test data across different models

So overall, we can see that adding a single post-aggregation transformer block does not really help. Replacing the post-aggregation transformer block with a for-aggregation graph transformer layer may help a little.

5 Conclusion

This project set out to explore the potential of integrating Graph Attention Networks (GATs) with post-aggregation Transformer blocks to enhance node classification accuracy on graph-based data. Our experimental investigation, conducted on the Cora and Citeseer datasets, has provided valuable insights into the synergistic effects of combining these two powerful architectures. The results indicate that the inclusion of post-aggregation Transformer blocks in GATs does not lead to improvements in node classification tasks.

The most significant findings emerged from our model variants, particularly Model-2 and Model-3, which integrated a single post-aggregation transformer layer into the GAT architecture. These models demonstrated little improvements in accuracy compared to the baseline GAT model and other more complex variants.

Besides, we also found that mixing GAT layers and for-aggregation Graph Transformer layers may outperform pure GATs and pure Graph transformers on certain tasks, as demonstrated most strongly by Model-6.

In conclusion, this project highlights the potential of hybrid GNN architectures in improving node classification accuracy. The nuanced performance variations observed in our models underscore the importance of architectural balance and adaptation to specific graph structures. As we continue to push the boundaries of what GNNs can achieve, integrating diverse approaches like GATs and Transformers may play a pivotal role in unlocking new capabilities and understanding in graph-based machine learning.

References

[1] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. "Revisiting semi-supervised learning with graph embeddings". In: *International conference on machine learning*. PMLR. 2016, pp. 40–48.

[2] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[3] Petar Velickovic et al. "Graph attention networks". In: *stat* 1050.20 (2017), pp. 10–48550.

[4] Seongjun Yun et al. "Graph transformer networks". In: *Advances in neural information processing systems* 32 (2019).

[5] Yunsheng Shi et al. "Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification". In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. Ed. by Zhi-Hua Zhou. Main Track. International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 1548–1554. DOI: 10.24963/ijcai.2021/214. URL: https://doi.org/10.24963/ijcai.2021/214.