Comparing the Performance of Node Classification over Multiple GNN models

Ethan Reinhart

December 16, 2024

Abstract

Node classification is a fundamental task in graph machine learning, with applications spanning social network analysis, biological networks, and recommendation systems. In this study, we evaluate the performance of various graph neural network (GNN) architectures on node classification tasks. Specifically, we consider models such as Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), and Graph Isomorphism Networks (GIN), alongside enhanced variants incorporating normalization and dropout. The models are assessed across multiple datasets, with a focus on metrics such as training accuracy, validation accuracy, test accuracy, and training loss. Through this comprehensive evaluation, we aim to provide insights into the strengths and limitations of each architecture, highlighting their suitability for different graph structures and learning tasks.

1 Introduction

Graph-based data is ubiquitous, representing entities as nodes and their relationships as edges in diverse domains such as social networks, biological networks, and citation networks. The node classification task, which involves assigning labels to nodes based on their features and graph structure, is a critical problem in these applications. Traditional machine learning techniques often fall short of effectively leveraging graph topology, necessitating the use of specialized methods like Graph Neural Networks (GNNs).

GNNs have emerged as a powerful paradigm for learning on graphs, leveraging message-passing mechanisms to propagate information across nodes and edges. Among the prominent GNN architectures are the Graph Convolutional Network (GCN), which employs spectral graph theory for convolution operations; the Graph Attention Network (GAT), which introduces attention mechanisms for adaptive neighborhood aggregation; and the Graph Isomorphism Network (GIN), designed to maximize expres-

siveness and capture graph structure.

Despite the success of these models, their performance can vary significantly depending on the dataset and task requirements. We used four datasets as our benchmarks for model comparison. Two datasets used were source from the WebKB database, accessible through the $torch_geometric$ libraries. These datasets represent a structure of webpages as nodes, classified into categories based on keywords, and hyperlinks within a webpage as edges. These datasets have low homophily scores. The other two datasets used were from the Planetoid database, also accessible through the $torch_geometric$ libraries. In these datasets, nodes represent documents and edges represent citation links.

In this work, we systematically compare GNN models on node classification tasks with the homophily score of the training dataset. We evaluate each model on multiple datasets, analyzing their training accuracy, validation accuracy, test accuracy, and training loss. Our goal is to provide a comprehensive understanding of how these models perform in comparison with one another as well as how GNN-based models perform when working over datasets of varying homophily.

2 Methodology

We created a total of seven models to compare with one another, fine-tuning each model to optimize performance given the architecture. Our created models are as follows

1. GCN
   (a) Linear Layer
   (b) Label Propagation Layer
   (c) Activation Layer (ReLU)
   (d) Linear Layer
   (e) Label Propagation Layer
   (f) Softmax Function
2. GAT
   (a) GAT Convolutional Layer
   (b) Activation Layer (ELU)
   (c) GAT Convolutional Layer
   (d) Softmax Function
3. GIN
   (a) GIN Convolutional Layer
   (b) Activation Layer (ReLU)
   (c) GIN Convolutional Layer
   (d) Softmax Function
4. GCN + Batch Normalization + Dropout
   (a) Linear Layer
   (b) Label Propagation Layer
   (c) Batch Normalization Layer
   (d) Activation Layer (ELU)
   (e) Dropout Layer
   (f) GCN Convolutional Layer
   (g) Label Propagation Layer
   (h) Softmax Function

5. GCNConv + Batch Normalization + Dropout
   (a) GCN Convolutional Layer
   (b) Batch Normalization Layer
   (c) Activation Layer (ELU)
   (d) Dropout Layer
   (e) GCN Convolutional Layer
   (f) Softmax Function
6. GCNConv + Batch Normalization + Dropout + GCNConv Hidden Layer
   (a) GCN Convolutional Layer
   (b) Batch Normalization Layer
   (c) Activation Layer (ELU)
   (d) Dropout Layer
   (e) GCN Convolutional Layer
   (f) Batch Normalization Layer
   (g) Activation Layer (ELU)
   (h) Dropout Layer
   (i) GCN Convolutional Layer
   (j) Softmax Function
7. GCNConv + Dropout + Linear Hidden Layer
   (a) GCN Convolutional Layer
   (b) Batch Normalization Layer
   (c) Activation Layer (ELU)
   (d) Dropout Layer
   (e) Linear Layer
   (f) Batch Normalization Layer
   (g) Activation Layer (ELU)
   (h) Dropout Layer
   (i) GCN Convolutional Layer
   (j) Softmax Function

These models were chosen to provide a variety of different implementations of GCNs to compare to.

The specific datasets used were the Cornell and Texas datasets from WebKB and the Cora and CiteSeer dataset from Planetoid. The homophily scores from these datasets are as follows:

1. Texas: 0.10769
2. Cornell: 0.13087
3. Cora: 0.80997
4. CiteSeer: 0.73550

We see the Texas and Cornell datasets have low homophily while the Cora and CiteSeer datasets have high homophily. This serves as a good metric to compare datasets with varying homophily.

3 Results

We will be using the training loss, training accuracy, validation accuracy, and testing accuracy as our benchmarks for model analysis. As a baseline, here we show that our accuracy increases for most models over epochs. This shows that the model is continuing to learn and that 200 epochs is a reasonable stopping point for the models.

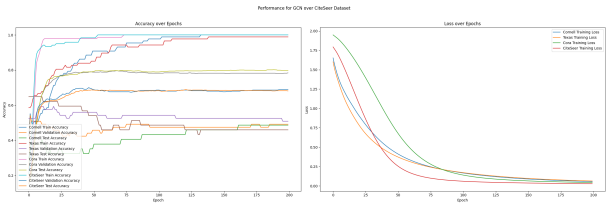

Figure 1: GCN Performance on CiteSeer

The rest of the models exhibited similar performance. We see quickly that the GCN (and all other graph-based models), perform significantly better on datasets with high homophily. This holds across all models we use. Depictions of performances across all other models can be found on GitHub.
We show the average metrics for each model. The average metrics for each model, meaning that the final metrics across all datasets for each model were averaged. We see that

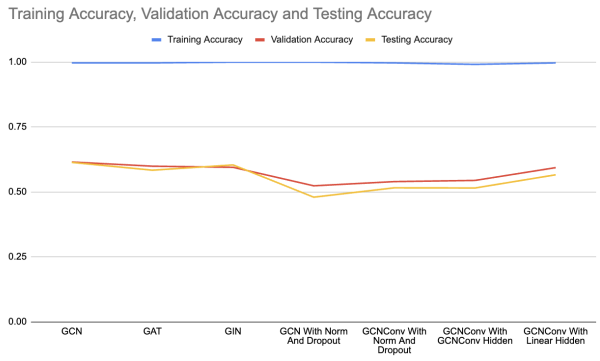

Figure 2: Average Final Metrics vs Models

the GCN marginally outperforms all other models. We additionally see that common techniques such as Batch Normalization and Dropout do not help with these models and datasets. We hypothesize this is because the datasets were relatively sparse and there dropout inhibits learning for the trade-off of better generalization. This would be beneficial in very large datasets but due to the limited size, we see this reduces model accuracy slightly. We hypothesize that Batch Normalization inhibits performance as it speeds up gradient convergences. This could lead to gradient convergence to a local minimum.
We see that the baseline GCN performs better than all other models. We see the GAT performs similarly to the GCN. We hypothesize this is due the attention mechanism allowing attention weights to neighbors as well as the message-passing mechanism. We hypothesize that the relatively small dataset leads to greater attention on neighboring nodes resulting in slightly worse performance when compared with using a very large, very sparse dataset. We see that the GIN model performs similarly as well. We believe this is due to the message-passing mechanism and feature aggregation of neighbors. These both function very similarly to the label propagation algorithm as we therefore should expect better performance.
For the rest of the models, tuning hyperparameters and adding more layers proved relatively futile. This is likely because the original GCN was well-tuned and had an optimal architecture for working with the dataset. Even more architecture modifications and hyperparameter tuning could likely lead to marginal benefits, but we were satisfied with the provided results.

4 Conclusion

We note that simpler architectures prove more beneficial when training over relatively small, sparse datasets. We see this in the performance comparison with the simple architectures of GIN, GAT, and GCN the show the best performance. In the future, we would like to experiment with other models such as LightGCN and APPNP. We would like to also go further into architecture modifications for simple models such as the GCN model.
We see that these graph-based models perform best on datasets with high homophily. In our future work, we will use graph-based models for datasets with high homophily and other models for datasets with low homophily.

Acknowledgements

Professor Yu Wang