Hanoi University of Science and Technology
School of Information and Communications Technology

Semester: 2024.1

Project I

Supervisor: Assoc. Prof. Le Duc Hau

Report

Graph convolutional networks for node classification on citation network

Name	Vu Trung Thanh
Student ID	20220066
Class	Data Science & AI 03 - K67

December 2024

Contents

1 Introduction

Graph Neural Networks (GNNs) are powerful tools for analyzing graph-structured data that can capturing complex relationships between entities by leveraging neighborhood information. Unlike traditional methods, GNNs leverage the graph structure, making them effective for solving problems in domains like social, biological, and citation networks.

Among GNNs, **Graph Convolutional Networks** (GCNs) stand out as the basic building blocks for various deep graph representation learning algorithms. GCNs extending convolution operations to graphs by aggregating and transforming features from neighboring nodes. This enables GCNs to learn meaningful node representations for tasks such as node classification and link prediction.

Citation networks, which represent scholarly publications as nodes and their citations as edges, provide a rich source of information for various tasks, including node classification. Node classification in this context involves assigning labels (e.g., research area, topic) to papers based on their citation patterns and content.

This project aims to evaluate the performance of various classic GCNs architectures for node classification on a citation network dataset, and focus on understanding how different GCN variants capture and utilize graph structure to predict node labels effectively.

2 Methods

2.1 GNNs

Let us define a graph G as $G = (V, E, X, Y)$, where:

- V denotes the set of nodes.
- $E \in \mathbb{R}^{|V|\times|V|}$ represents the set of edges.
- $X \in \mathbb{R}^{|V|\times d}$ is the node feature matrix, with $|V|$ representing the number of nodes and d the dimension of the node features.
- $Y \in \mathbb{R}^{|V|\times C}$ is the one-hot encoded label matrix, with C being the number of classes.

Let $A \in \mathbb{R}^{|V|\times|V|}$ denote the adjacency matrix of G. The task of interest is **node classification**, where each node $v \in V$ has a label $y_v \in Y$, and the goal is to learn a representation vector h_v of v such that its label can be predicted as $y_v = f(h_v)$.

Most architecture of GNNs follow a neighborhood aggregation strategy (message-passing GNNs) [4], where node representation h_v^l of v at layer l is computed as

$$h_v^l = \text{UPDATE}^l \left(h_v^{l-1}, \text{AGG}^l \left(\left\{ h_u^{l-1} | u \in \mathcal{N}(v) \right\} \right) \right),$$

where:

- $\mathcal{N}(v)$ represents the neighboring nodes adjacent to v.
- AGG^l is the message aggregation function at layer l.
- UPDATE^l is the update function at layer l.

Initially, the representation vector of each node v begins as the feature vector of that node: $h_v^0 = x_v \in X$. After l layers of the network, the output h_v^l is the result representation vector of v.

2.2 Model architectures

The architectures derived from message-passing GNNs mainly differ in their specific choices of the AGG^l and UPDATE^l functions. In this project, four classical GNN models are explored: **GCN**, **GraphSAGE**, **GAT**, and **GIN**.

Graph Convolutional Networks (GCN) [3] is the most well-known GNN architecture, which can be formulated as

$$h_v^l = \sigma \left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{1}{\sqrt{\hat{d}_u \hat{d}_v}} W^l h_u^{l-1} \right).$$

Here, $\hat{d}_v = 1 + \sum_{u \in \mathcal{N}(v)} 1$ is the degree of node v plus 1, and the term $\frac{1}{\sqrt{\hat{d}_u \hat{d}_v}}$ acts as a normalization factor; W^l is a learnable parameter matrix at layer l. σ is the nonlinear activation function, such as ReLU.

GraphSAGE [1] leverages node feature information to efficiently generate node embeddings, by learning a function that generates embeddings by sampling and aggregating features from a node's local neighborhood. In this project, the mean aggregator was used:

$$h_v^l = \sigma \left(W_1^l h_v^{l-1} + \left(W_2^l \cdot \text{mean}_{u \in \mathcal{N}(v)} h_u^{l-1} \right) \right).$$

W_1^l and W_2^l are learnable parameter matrices, and $\text{mean}_{u \in \mathcal{N}(v)} h_u^l$ calculate the average representation of neighbors of v.

Graph Attention Network (GAT) [5] instead make use of the attention mechanism $a : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ that is used to compute *attention coefficients* $e_{ij} = a(Wh_i, Wh_j)$ that indicate the importance of

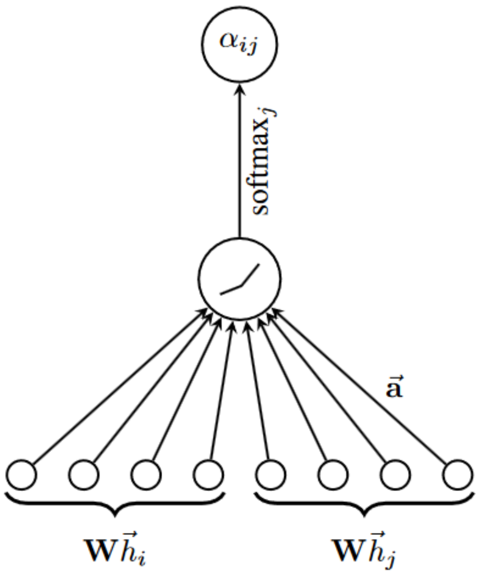


Figure 1: The attention mechanism, as described in [5].

node j's features to i. The node representation thus can be calculated as

$$h_v^l = \sigma\left(\sum_{u\in\mathcal{N}(v)\cup\{v\}} \alpha_{vu}^l W^l h_u^{l-1}\right), \text{where } \alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k\in\mathcal{N}(i)\cup\{i\}} \exp(e_{ik})}.$$

The specific implementation of attention mechanism a for this project is similar to the original paper's implementation, where Wh_i and Wh_j are first concatenated, then passed through single-layer feed-forward neural network and applied the LeakyReLU activation function.

Graph Isomorphism Network (GIN) [6] is provably the most expressive among the class of GNNs, and is as powerful as the Weisfeiler-Lehman graph isomorphism test for distinguish graph structures. Node representation can be iteratively calculated as

$$h_v^l = \text{MLP}^l\left((1+\epsilon^l)\cdot h_v^{l-1} + \sum_{u\in\mathcal{N}(v)} h_u^{l-1}\right),$$

where MLP^l is a multi-layer perceptron, and ϵ^l is a learnable scalar parameter. In this project, MLP^l is implemented as a two-layer MLP with ReLU activation.

3 Experimental setup

3.1 Dataset

The dataset used in the project is **ogbn-arxiv** provided by OGB [2], which is a directed graph representing the citation network between all Computer Science arXiv papers indexed by MAG. Each node is an arXiv paper, and each directed edge represents the citation of one paper with another paper. The feature vector for each node is a 128-dimension vector, represent the content of the title and abstract.

Graph type	# nodes	# edges	# features	# classes	Metric
Homophily	169,343	1,166,243	128	40	Accuracy

Table 1: Statistics of the ogbn-arxiv dataset.

The task provided is to classify the papers' subject into 40 areas, e.g., cs.AI, cs.OS, etc. The labels for the nodes are manually labeled by the authors and arXiv moderators.

3.2 Experimental settings

Due to the size of the dataset, the adjacency matrix is first preprocessed into a sparse format, which saves GPU memory and makes training the model feasible. The adjacency matrix is further processed to ensure symmetry by converting it into an undirected format. This step is crucial for the architectures used that assume undirected edges.

For training, multi-class cross-entropy loss calculated on class prediction and label was used. Accuracy was used to evaluate the performance of the models.

The data was split based on the original proposal from [2], which split the data based on the publication dates of the papers. Specifically, the models was trained on papers published until 2017, validate on papers published in 2018, and test on papers published from 2019 to now, mimicking application of the models for real-world scenario.

Batch normalization and dropout was utilized to stabilize the training process and reduce overfitting, which ultimately leads to better model performance. A set of hyperparameters was chosen after extensive testing. The models were trained with a hidden dimension size of 256 and a total of three layers. A dropout rate of 0.25 is applied during training. The models are optimized using the Adam optimizer with a learning rate of 0.01. Each model ran for 10 runs, and the results were averaged. All the model are run on local computer, with only GAT model run on cloud computing service due to the GPU memory limitation.

4 Results and Evaluation

Model	Training loss	Accuracy (%)			Time (s)
		Train	Validation	Test	
GCN	0.5505 ± 0.0018	83.21 ± 0.17	**72.81 ± 0.13**	**72.11 ± 0.06**	200.7 ± 0.5
GraphSAGE	**0.3005 ± 0.0019**	**96.05 ± 0.13**	72.33 ± 0.12	71.69 ± 0.30	**181 ± 0.7**
GAT	0.4999 ± 0.0031	84.52 ± 0.16	**72.81 ± 0.09**	71.92 ± 0.21	302.7 ± 0.5
GIN	0.7980 ± 0.0073	76.92 ± 0.21	72.43 ± 0.08	71.17 ± 0.13	190.1 ± 0.7

Table 2: Models performance (Mean ± Std). The best result are in bold.

The training process graphs are included in the Appendix of this report. Among the models, GraphSAGE achieved the lowest training loss and highest training accuracy. However, this clearly indicates overfitting, as there is a significant gap between the training accuracy and the validation/test accuracy. The best-performing models are GCN and GAT, with GCN slightly outperforming GAT in test accuracy.

The GIN model, while not achieving the best results, demonstrates a smaller gap between training and validation/test accuracy, suggesting potential for improved performance with further tuning. Regarding resource usage (time and memory), GCN, GraphSAGE, and GIN are comparable, while GAT requires significantly more time and memory, making it less efficient in this context.

5 Conclusion

This project explored the application of Graph Neural Networks (GNNs) for node classification on the ogbn-arxiv citation network. Four classical GNN architectures — GCN, GraphSAGE, GAT, and GIN — were implemented and compared based on their training performance, generalization ability, and computational efficiency.

The findings demonstrate that GCN and GAT achieved the best test accuracy, with GCN slightly outperforming GAT in terms of generalization. GraphSAGE exhibited the highest training accuracy but suffered from overfitting, as evidenced by the significant gap between training and validation/test accuracy. GIN, while not the top performer, showed potential for further improvement due to its balanced performance across training and evaluation metrics.

From a resource perspective, GCN, GraphSAGE, and GIN offered similar training efficiency, whereas GAT required significantly more time and memory due to its attention mechanism. These insights highlight the trade-offs among the models and suggest that the choice of architecture should consider not only accuracy, but also resource constraints and scalability requirements.

This project has several limitations. First, it did not fully leverage all the dataset's information, as edges were treated as undirected and the temporal feature (publication year) was ignored. Second, the study did not explore more advanced architectures, such as graph transformers or self-supervised learning approaches, which have shown promise in recent research. Lastly, due to the complexity of the dataset, qualitative analysis of the results, such as visualizing node embeddings, was not conducted, limiting deeper insights into model behavior.

Future work could address these limitations by incorporating directed edges and temporal data into the model and experimenting with advanced architectures. Additionally, exploring qualitative methods for analyzing and visualizing node embeddings could provide a richer understanding of the model's performance and its ability to capture meaningful patterns in the data. These efforts could further enhance both accuracy and interpretability, making GNNs even more effective for real-world graph tasks.

6 Acknowledgement

I would like to express my sincere gratitude to Assoc. Prof. Le Duc Hau for providing me with such an interesting and challenging topic, as well as for their guidance and support throughout this project.

References

[1] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs, 2018.

[2] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. Open graph benchmark: Datasets for machine learning on graphs, 2021.

[3] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks, 2017.

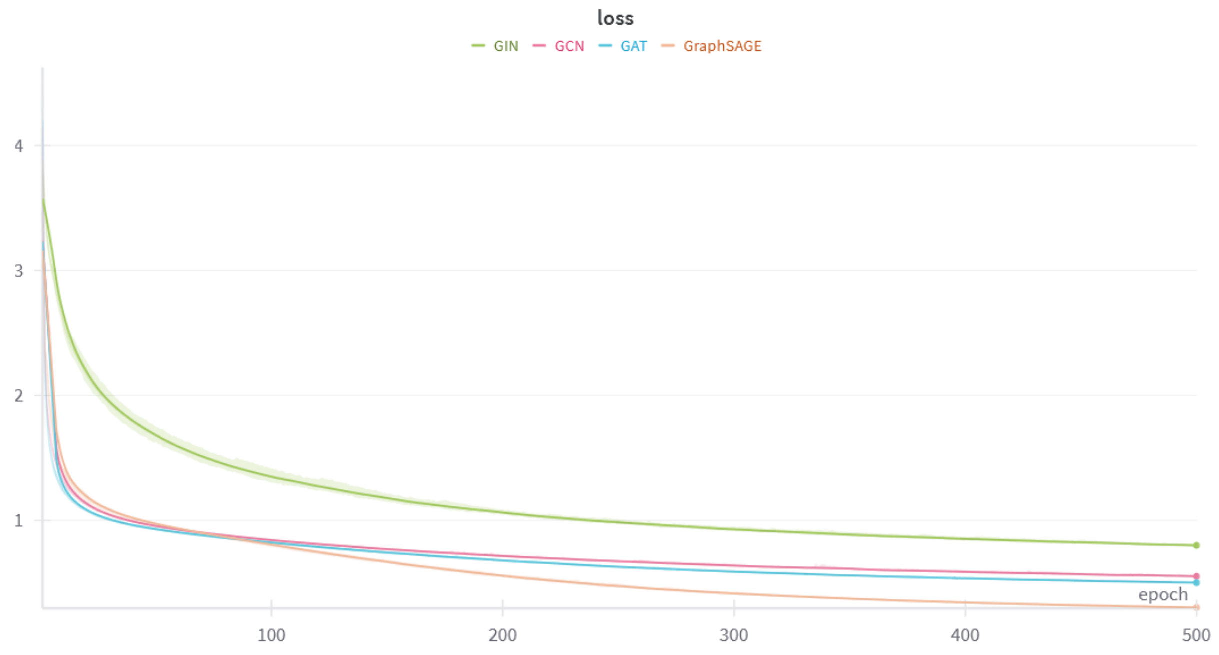
[4] Y. Luo, L. Shi, and X.-M. Wu. Classic gnns are strong baselines: Reassessing gnns for node classification, 2024.

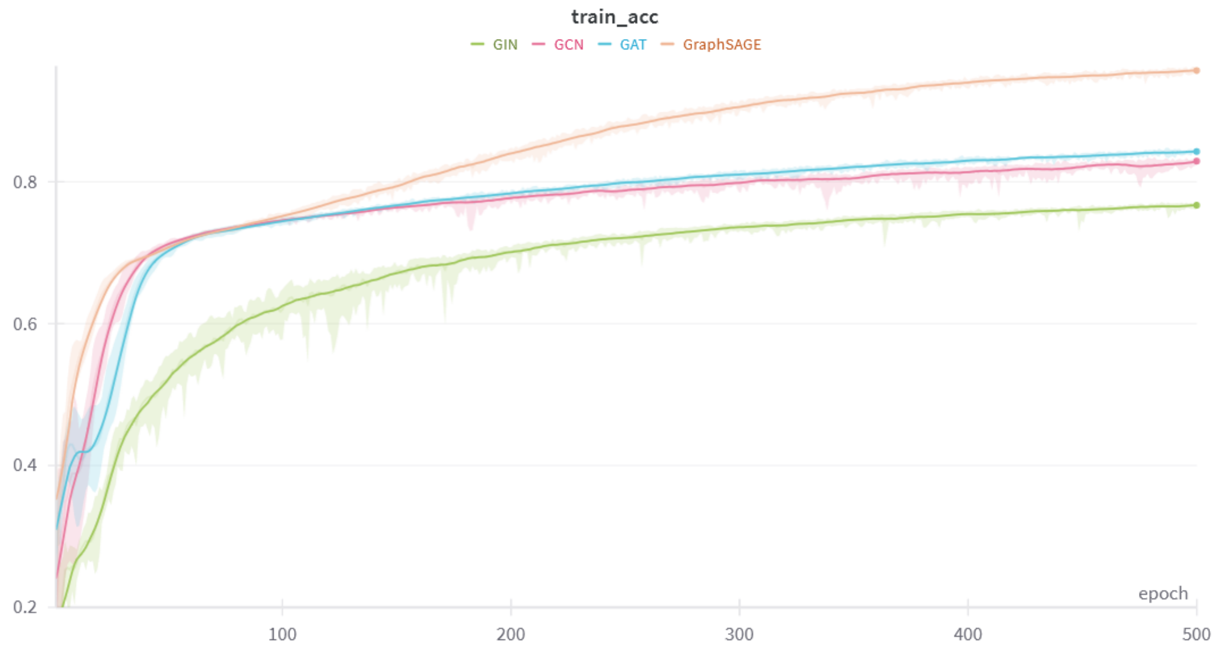
[5] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks, 2017.

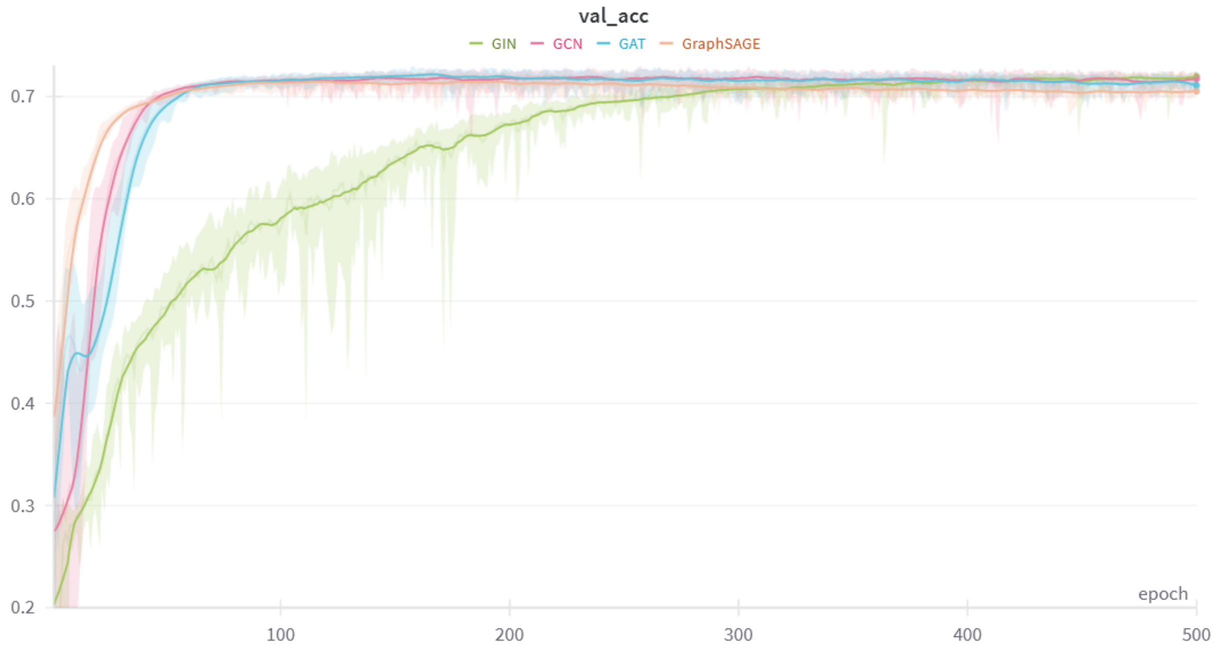
[6] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks?, 2019.

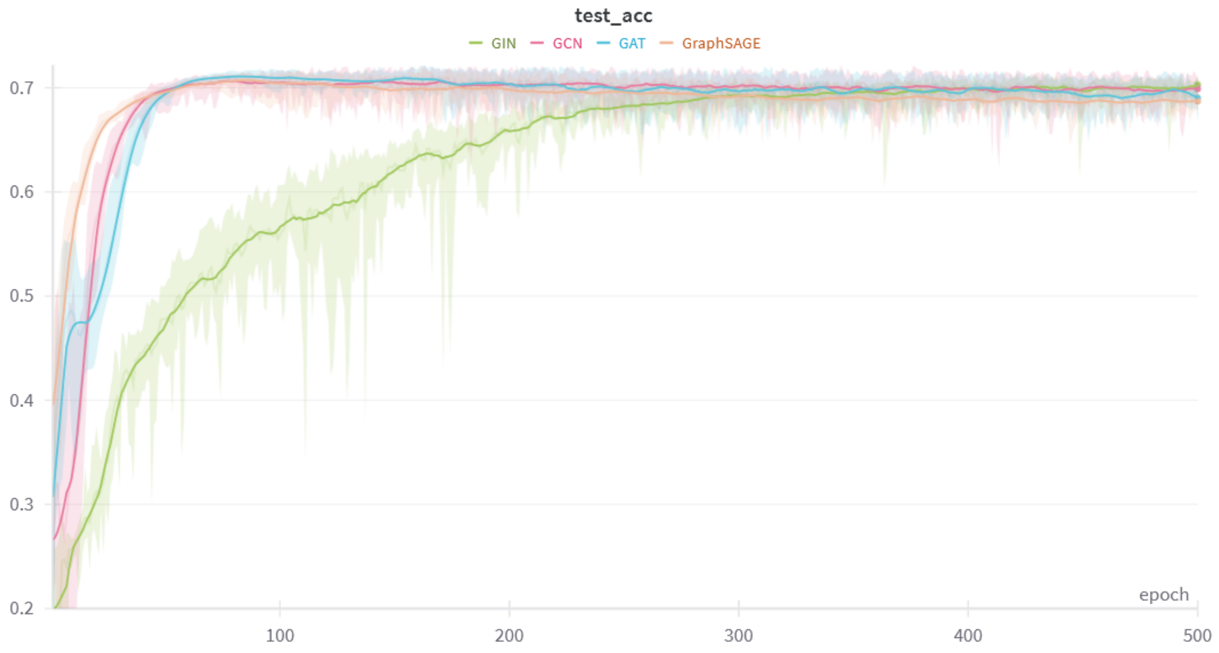
Appendix

Here are the graphs for the training process of the models.






val_acc
GIN
GCN
GAT
GraphSAGE
0.7
0.6
0.5
0.4
0.3
0.2
epoch
100
200
300
400
500


test_acc
GIN
GCN
GAT
GraphSAGE
0.7
0.6
0.5
0.4
0.3
0.2
epoch
100
200
300
400
500