Optimizing Graph Neural Networks for Enhanced Community Detection in Social Networks

Parsa Kamalipour
Department of Computer Science
Concordia University
parsa.kamalipour@mail.concordia.ca

Abdulmoumen Al-Atrash
Department of Computer Science
Concordia University
a_alatr@live.concordia.ca

Aniket Roy
Department of Computer Science
Concordia University
aniket.roy@mail.concordia.ca

Abstract

Community detection, which is a main problem in social network analysis, looks to find clusters in graphs where nodes are more connected with each other than with the rest of the network. This is important in different fields like social network analysis and biology, and it helps in understanding complex systems and their organization. Moreover, the recent growth of social networks through the inherent interconnectedness of the internet has made properly analyzing these intricate structures increasingly important. Traditional methods like modularity optimization and spectral clustering have limitations in handling large networks and often miss detailed community structures. Optimization-based approaches, like Learning-Based Genetic Algorithms (LGA) and Quantum-Inspired Optimization, achieve impressive accuracy but struggle with generalizing to networks of varying complexity and domains. Similarly, modularity-based methods, while interpretable and computationally efficient, rely too heavily on modularity maximization and can fail to distinguish between communities within networks of varying sizes, limiting their effectiveness. Deep learning excels at capturing complex relationships but can be computationally intensive when applied to large-scale networks. One notable model, Graph Neural Networks (GNNs), has emerged as a powerful deep learning approach to solving the community detection problem. GNNs have demonstrated significant potential in overcoming the challenges faced by traditional methods, including issues with scalability, accuracy, and adaptability to networks of diverse sizes and complexities. This project aims to solve these limitations by using machine learning, specifically GNNs, due to their strong capabilities of identifying key patterns in complex data networks by effectively learning the high-dimensional feature representations of nodes and communities. With these tools, the project tries to make community detection more accurate and scalable. This plan aligns with course objectives, where we apply theoretical knowledge to practical problems in network analysis and machine learning. It addresses gaps in the existing methodologies for solving community detection by highlighting the merit of GNNs over traditional methods. Additionally, scalable enhancements such as graph sampling methods, optimized graph convolutions, and graph partitioning strategies are investigated to address the computational challenges associated with large-scale networks.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

1 Introduction

1.1 Introduction to Community Detection

Community detection is all about identifying clusters or groups within a network - which are called communities - where nodes are more densely connected to each other than to the rest of the network. This problem is very crucial in various domains, including social network analysis, biology, and information retrieval, as it helps uncover the underlying structure and functional organization of complex systems. For instance, community detections in biology aids in identifying interconnected gene networks, while in social networks, it reveals social group dynamics and interaction patterns. Traditional methods, such as modularity optimization and spectral clustering, have been widely used for this purpose. However, these approaches often face challenges in handling large-scale networks and capturing intricate community structures, particularly when networks exhibit complex relationships or irregular structural patterns. As social networks continue to grow and become more deeply integrated into daily life, it is becoming vital to analyze these complex structures efficiently and on a large scale. This development demands advanced solutions that maintain performance while achieving scalability to keep up with the diverse and dynamic nature of real-world networks.

1.2 Machine Learning Context

Recent advancements and novelties in machine learning, particularly deep learning, have introduced lots of new methodologies for community detection. Graph Neural Networks (GNNs) have emerged as very powerful tools capable of learning complex patterns in graph-structured data. For instance, the Contrastive Deep Nonnegative Matrix Factorization (CDNMF) model integrates contrastive learning with deep nonnegative matrix factorization to enhance and improve on community detection by capturing both network topology and node attributes at the same time [1]. Additionally, the integration of community detection algorithms with GNNs has been shown to improve link prediction tasks in scientific literature networks, demonstrating the synergistic potential of combining these approaches [2]. This highlights the latent potential of GNNs in addressing the limitations of modularity-based and optimization-based methods later discussed in this paper in offering both accuracy and scalability for complex networks.

1.3 Usage of Machine Learning in Community Detection

In this project, we aim to use machine learning techniques to address the challenges of community detection in large-scale networks, to be more precise in Social Networks. By using models such as GNNs and incorporating methods like optimization on previous ideas, we seek to develop an algorithm that can effectively identify communities with high accuracy and scalability. This approach aligns with the course objectives by applying machine learning methodologies to solve complex problems, thereby by doing this research we both learn a new Machine Learning methodology and also enhance our understanding of community structures in various real-world applications.

1.4 Challenges in Community Detection

Traditional approaches to community detection face significant limitations that complicate their effectiveness in modern, large-scale applications. One major challenge for community detection when seeking to maintain high performance and robust detection capabilities is scalability [3]. In the real-world applications, networks are generated at massive scale and necessitate efficient processing, often demanding methods that can process millions or even billions of nodes and edges efficiently. Another complication lies in the inherently noisy and irregular nature of data within networks [4]. Particularly, incorrect links, classified as false positives, and missing links, classified as false negatives, can distort community structures and lead to loss of accuracy for less nuanced methods. Finally, some networks tend to be heterogeneous such that nodes and edges can carry varying types of data [5]. This can pose issues to algorithms that rely heavily on structural relationships in the network to capture contextual information. Community detection is an inherently indeterminate problem—meaning that there is often no singular correct solution. This ambiguous nature demands robust methods that can generalize well to networks of diverse structures and sizes, which lays the foundation for the challenges explored in this paper.

2 Related Work

2.1 Overview

This section presents a literature review to examine pre-existing knowledge in the area of the NP-Hard problem of 'Community Detection' in social networks. We examine relevant existing work to ensure that our proposed approach is justified based on the problem context and that it differs from already proposed solutions to the NP-Hard problem. In this section, we identify and discuss three main categories of algorithms used to tackle the problem of community detection, namely optimization-based methods, modularity-based approaches, and deep learning techniques.

2.2 Optimization-Based Methods

A rather popular approach to solving challenging and inherently complex machine learning problems such as community detection is optimization. The Learning-Based Genetic Algorithm (LGA) [6] [Identifying Communities in Complex Networks Using Learning-Based Genetic Algorithm] is designed to tackle the NP-hard nature of community detection problems. This method carefully combines Learning Automata with genetic operations to get around problems that genetic algorithms often have, such as unwanted premature convergence and local optima. The LGA achieves a notable improvement in accuracy of 26.47% on real-world networks and a 48.32% improvement on synthetic networks. Akbar et al. [7] propose a quantum mechanics-inspired optimization approach that aims to detect underlying complex patterns, specifically in ecological communities. This paper simulates quantum principles for optimization detections for identifying patterns in biodiversity changes due to environmental factors such as land use and climate change. MARLCD [8] is a multi-agent reinforcement learning algorithm designed for community detection within complex networks. The algorithm employs "agents" to independently explore communities within the network, updating actions on successful detections. MARLCD outperforms state-of-the-art methods such as GA-Net and Meme-Net when tested on both real and synthetic datasets. However, these three approaches lack generalizability on networks of varying domains and complexity, as their underlying assumptions make them effective within their respective applications but restrict their flexibility and may be computationally excessive when applied to networks with different structural patterns and complexities.

2.3 Modularity-Based Methods

Modularity-based algorithms are simpler and easily interpretable techniques that are widely known for their community detection ability by maximizing modularity. Modularity is a novel metric based on the density of connected nodes within communities compared to their density with the rest of the network. Chen et al. [9] introduce a hierarchical clustering algorithm that is applied to optimize Max-Min Modularity. The authors argue that traditional statistical inference procedures make strong assumptions that instances are independent, and that relation-based methods cannot distinguish between features of the network domain, both of which lead to problematic conclusions about the data and lower performance. The Hybrid Genetic Tabu (HGT) [10] method aims to solve the community detection problem by maximizing modularity through a combination of Genetic Algorithms (GA) and Tabu Search (TS). This hybrid approach achieved higher detection accuracy compared to methods such as the Louvain and Label Propagation and demonstrated a stronger inclination toward medium-sized networks. Nevertheless, modularity-based approaches face several limitations; most prominent is their strong reliance on maximizing modularity as the primary objective, which is known to often be unable to distinguish differences in community sizes and connectivity, leading to suboptimal detection accuracy.

2.4 Deep Learning Methods

When it comes to utilizing powerful machine learning models to handle complex data processing, feature extraction, and pattern recognition, deep learning most often immediately comes to mind. Liu et al. [11] discuss various deep learning techniques for community detection, including deep neural networks, graph neural networks, and deep graph embeddings. The paper highlights the unique ability of deep learning models such as GNNs to overcome limitations of heavily relying on adjacency matrices and node attribute matrices by encoding higher-dimensional feature representations of nodes

and communities. These limitations, circumvented by appropriate deep learning techniques, are prominent in traditional community detection methods, which often lead to the loss of complex structural relationships. The deep transitive encoder [12] is a novel approach that transforms the network's adjacency matrix to capture indirect node relationships that traditional methods often miss. The autoencoder utilizes unsupervised transfer learning to effectively extract low-dimensional features from the network structure. This led to improved accuracy over traditional methods; however, this approach's reliance on the adjacency matrix transformation was found to be very computationally demanding, which poses a problem for large-scale real-world networks. Nooribakhsh et al. [13]. systematically provide a comprehensive overview of machine learning trends in tackling the community detection problem. The paper highlights the recent growth of deep learning applications on large-scale, complex networks due to their consistency in outperforming simpler methods, such as game-theoretic and clustering algorithms. Moreover, the authors emphasize its superiority over more traditional approaches, like density-based methods, which often lack the ability to grasp structural and feature-based characteristics of nodes in networks.

2.5 Conclusion

The literature review highlights the diverse and sophisticated attempts to solve the community detection problem in social networks, which is notoriously an NP-hard problem. To support our motivation, we highlight the advantages and limitations of several nuanced approaches in recent literature.

Optimization-based methods like the Learning-Based Genetic Algorithm (LGA) and Quantum-Inspired Optimization produce significant accuracy improvements over other state-of-the-art approaches but face complications in generalizing when presented with networks of varying domains and complexities. Traditional methods, such as modularity-based algorithms, provide simplicity and interpretability that come at the cost of a strict reliance on maximizing modularity. This restriction introduces limitations in identifying community structures and an inability to adapt to diverse community networks. Deep learning models, especially graph neural networks (GNNs), offer powerful solutions that serve to overcome these limitations by utilizing high-dimensional feature representations and avoiding the heavy reliance on adjacency matrices present in other models. Although techniques such as the deep transitive encoder and deep graph embeddings can achieve remarkable improvements over community detection accuracy, they often come with substantial computational demands to run.

This literature review serves as a justification of the proposed approach and highlights the potential of GNNs in solving the NP-hard problem of community detection while simultaneously minimizing limitations that accompany existing machine learning methods.

3 Proposed Approach

Scalability is a fundamental challenge in community detection, particularly in the context of large-scale graphs. Traditional methods often fail to meet the computational demands of modern networks, resulting in excessive memory requirements and prolonged training times. To address these limitations, this study proposes a comprehensive evaluation of Graph Neural Network (GNN) scalability through three distinct methodologies: full-batch training, neighbor sampling, and graph partitioning. These strategies are designed to optimize computational efficiency while preserving model accuracy.

The primary goal of this work is to analyze and compare various approaches to tackle scalability issues in large graphs during community detection. Training or evaluating on the entire graph at once can require significant system memory, especially for larger graphs. To address this, we test and compare multiple approaches to identify the ones that yield better accuracy and train time.

We implement a model using the GCNConv and GraphSAGEConv architectures, detailed below. The model structure can be referenced in the accompanying code implementation. Once implemented, the models are evaluated using three distinct training approaches, described later in this section.

3.1 GNN Architectures

This study employs two well-established GNN architectures, **GCNConv** and **GraphSAGEConv**, as the foundational models:

3.1.1 GCNConv

The `GCNConv` class implements the graph convolutional operator from the paper *"Semi-supervised Classification with Graph Convolutional Networks"* by Kipf and Welling (2016) [14]. This operator performs message passing by aggregating feature information from a node's neighbors, facilitating the learning of node representations that capture both local graph structure and node features.

Mathematical Formulation: The layer computes the output features $\mathbf{X}'$ as:

$$\mathbf{X}' = \hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\mathbf{X}\mathbf{\Theta}, \tag{1}$$

where:

- $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with added self-loops.
- $\hat{\mathbf{D}}$ is the diagonal degree matrix of $\hat{\mathbf{A}}$.
- $\mathbf{X}$ denotes the input feature matrix.
- $\mathbf{\Theta}$ represents the learnable weight matrix.

In a node-wise formulation, the output feature $\mathbf{x}'_i$ for node i is:

$$\mathbf{x}'_i = \mathbf{\Theta}^\top \sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{e_{j,i}}{\sqrt{\hat{d}_j \hat{d}_i}} \mathbf{x}_j, \tag{2}$$

where:

- $\mathcal{N}(i)$ denotes the set of neighbors of node i.
- $e_{j,i}$ is the edge weight from node j to node i (defaulting to 1.0 if not specified).
- $\hat{d}_i = 1 + \sum_{j \in \mathcal{N}(i)} e_{j,i}$ is the degree of node i in $\hat{\mathbf{A}}$.

3.1.2 GraphSAGEConv

The `GraphSAGEConv` class implements the GraphSAGE operator from the paper *"Inductive Representation Learning on Large Graphs"* by Hamilton et al. (2017) [15]. This operator enables inductive learning by aggregating feature information from a node's local neighborhood, allowing the model to generalize to unseen nodes or graphs.

Mathematical Formulation: The layer computes the output features $\mathbf{x}'_i$ for node i as:

$$\mathbf{x}'_i = \mathbf{W}_1 \mathbf{x}_i + \mathbf{W}_2 \cdot \mathrm{AGGREGATE}_{j \in \mathcal{N}(i)} \mathbf{x}_j, \tag{3}$$

where:

- $\mathbf{W}_1$ and $\mathbf{W}_2$ are learnable weight matrices.
- $\mathcal{N}(i)$ denotes the set of neighbors of node i.
- AGGREGATE is a function such as mean, LSTM, or max pooling that combines the features of neighboring nodes.

If the `project` parameter is set to `True`, each neighbor's feature $\mathbf{x}_j$ is first transformed via a non-linear function:

$$\mathbf{x}_j \leftarrow \sigma(\mathbf{W}_3 \mathbf{x}_j + \mathbf{b}), \tag{4}$$

where:

- $\mathbf{W}_3$ is a learnable weight matrix.
- $\mathbf{b}$ is a bias term.
- σ is an activation function (e.g., ReLU).

3.2 Scalability Techniques

3.2.1 Full-Batch Training

Full-batch training involves processing the entire graph simultaneously during each training epoch. While this method captures the global structure and relationships within the graph, it is computationally expensive for large graphs. Memory constraints often limit its applicability, as demonstrated by the prohibitive resource requirements for the Reddit dataset.

3.2.2 Neighbor Sampling

Neighbor sampling addresses the limitations of full-batch training by selectively sampling a fixed number of neighbors for each node during training. This approach significantly reduces memory usage while maintaining high accuracy. The methodology is based on the GraphSAGE architecture, which aggregates information from sampled neighbors to compute node embeddings.

3.2.3 Graph Partitioning

Graph partitioning divides the graph into smaller, non-overlapping subgraphs, which are processed independently during training. This method, inspired by the approach outlined in Chiang et al. [16], facilitates parallel processing and reduces the memory footprint. By isolating subgraphs, this technique effectively balances computational efficiency and model performance.

3.3 Expected Contributions

The proposed approach systematically evaluates the trade-offs between computational efficiency and accuracy across three scalability techniques. By benchmarking these methods on varying graph sizes and complexities, this study aims to:

1. Identify optimal strategies for scalable GNN training.
2. Provide insights into the applicability of these methods for real-world large-scale networks.
3. Advance the development of robust community detection techniques that meet the demands of modern network analysis.

4 Results, Training, and Evaluation

In this section, we present the results of our experiments, including the training process, evaluation metrics, and analysis of the performance of the proposed approaches.

4.1 Datasets

We conducted experiments using three datasets of varying sizes and complexity to evaluate the scalability and performance of the proposed methods:

- **Synthetic Graphs:** Generated using the Stochastic Block Model (SBM), these graphs provide controlled environments to simulate community structures and test the model under predefined conditions.
- **CORA Dataset:** A citation network widely used in graph learning tasks, with moderate size and complexity.
- **Reddit Dataset:** A large-scale dataset with approximately 100x the nodes of the CORA dataset, highlighting scalability challenges.

4.2 Training Approaches

The training process was carried out using three distinct methodologies:

1. **Full-Batch Training:** The entire graph is processed simultaneously. This method faced memory constraints on large datasets like Reddit.

2. **Neighbor Sampling:** A subset of neighbors for each node is sampled during training, significantly reducing memory usage and enabling training on larger graphs.

3. **Graph Partitioning:** The graph is divided into smaller subgraphs processed independently, facilitating parallel computation and scalability.

4.3 Evaluation Metrics

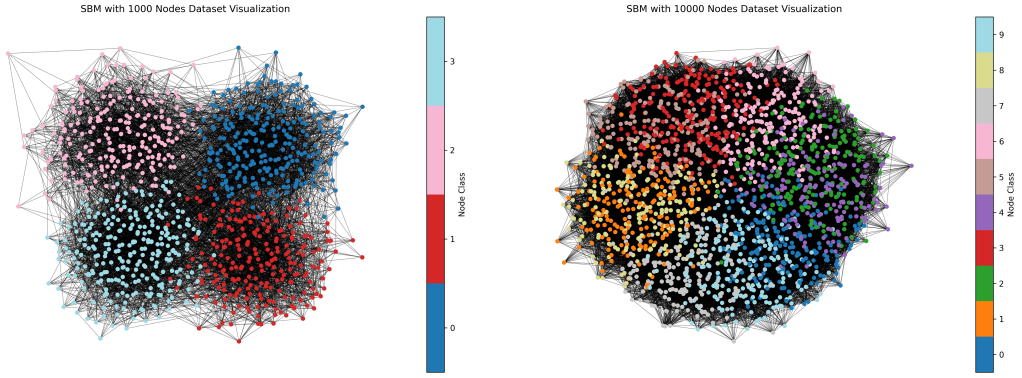
To assess the effectiveness of the models and approaches, we used the following metrics:

- **Accuracy:** The ability of the model to correctly classify nodes into communities.
- **Training Time:** The computational time required to complete the training process.
- **Memory Usage:** The memory consumption during training, indicating the scalability of each method.

4.4 Experimental Results

4.4.1 Synthetic Graphs

On synthetic graphs, all methods performed well in terms of accuracy. Neighbor Sampling and Graph Partitioning showed substantial reductions in memory usage and training time compared to Full-Batch Training.



(a) SBM dataset with 1000 nodes.

(b) SBM dataset with 10,000 nodes.

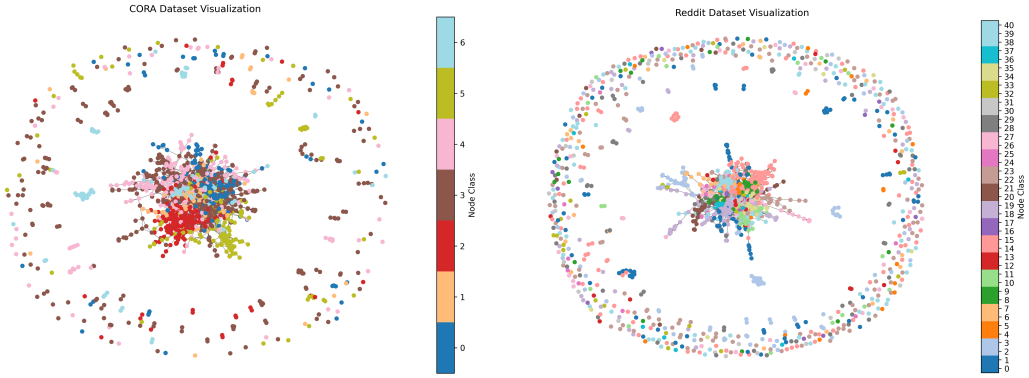
Figure 1: Visualization of SBM datasets with 1000 and 10,000 nodes. Node classes are color-coded to highlight community structures.

4.4.2 CORA Dataset

On the CORA dataset, Neighbor Sampling achieved comparable accuracy to Full-Batch Training but with reduced memory and time requirements. Graph Partitioning demonstrated similar efficiency gains while maintaining high accuracy.

4.4.3 Reddit Dataset

On the large-scale Reddit dataset, Full-Batch Training was infeasible due to excessive memory requirements. Both Neighbor Sampling and Graph Partitioning enabled training and achieved competitive accuracy. Among the two, Neighbor Sampling had slightly better training times, while Graph Partitioning excelled in memory optimization.



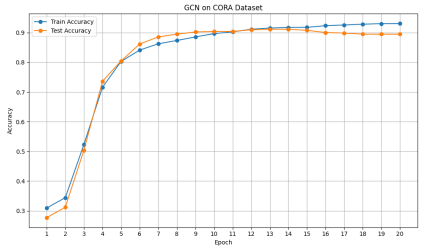
(a) Visualization of the CORA dataset.

(b) Visualization of the Reddit dataset.

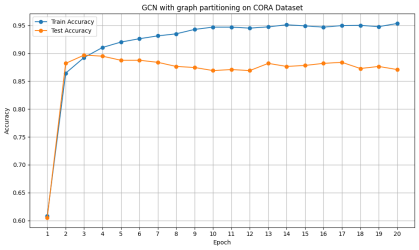
Figure 2: Visualizations of the CORA and Reddit datasets. Node classes are color-coded to represent different communities or research topics.

4.4.4 Performance on CORA Dataset

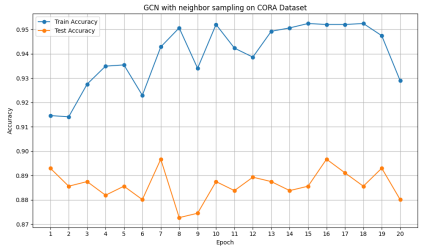
This subsection compares the performance of different models and training methods on the CORA dataset. The accuracy trends for both training and testing are shown for the various configurations.



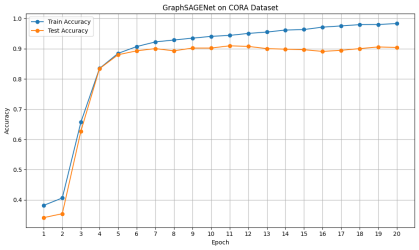
(a) GCN on CORA Dataset.



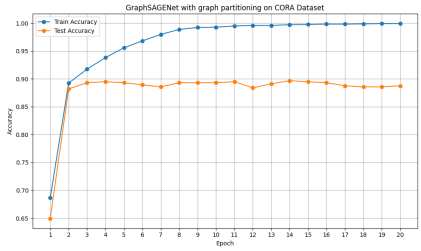
(b) GCN with Graph Partitioning on CORA.



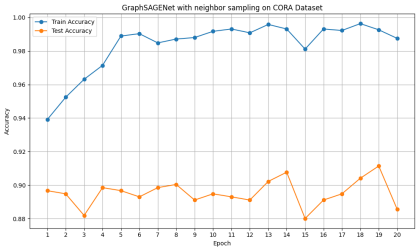
(c) GCN with Neighbor Sampling on CORA.



(d) GraphSAGE on CORA Dataset.



(e) GraphSAGE with Graph Partitioning on CORA.

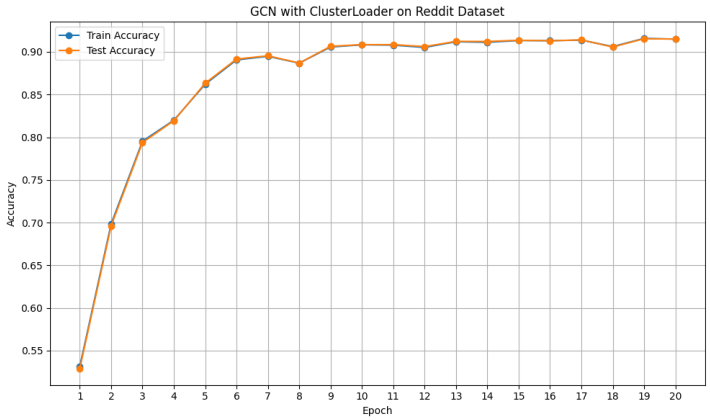


(f) GraphSAGE with Neighbor Sampling on CORA.

Figure 3: Performance comparison on the CORA dataset for GCN and GraphSAGE models under different training strategies: Full-Batch, Graph Partitioning, and Neighbor Sampling.

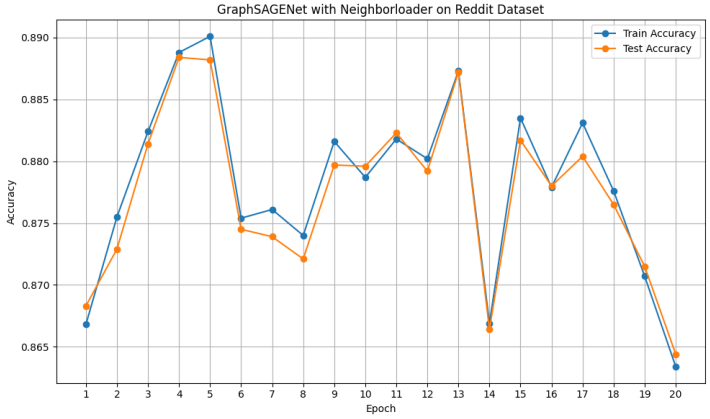
4.4.5 Performance on Reddit Dataset

This subsection compares the performance of different models and training methods on the Reddit dataset. The accuracy trends for both training and testing are shown for the various configurations.



(a) GCN with ClusterLoader on Reddit Dataset.

(b) GCN with Neighborloader on Reddit Dataset.

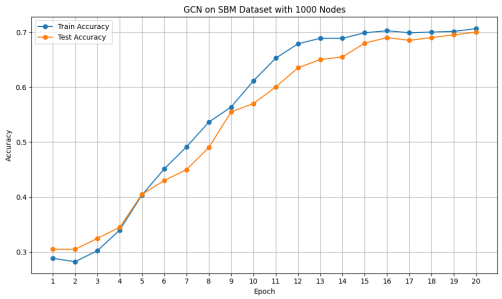


(c) GraphSAGE with Neighborloader on Reddit Dataset.

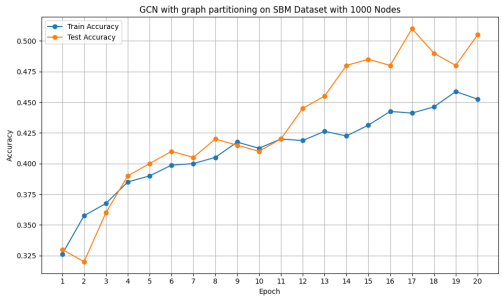
Figure 4: Performance comparison on the Reddit dataset for GCN and GraphSAGE models under different training strategies: ClusterLoader and Neighborloader.

4.4.6 Performance on SBM Dataset (1000 Nodes)

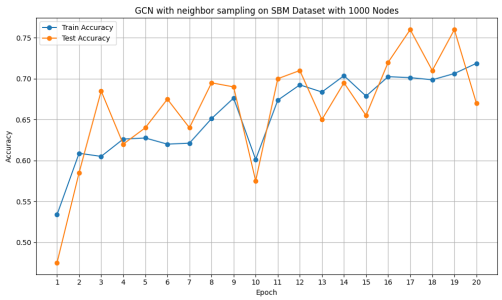
This subsection compares the performance of different models and training methods on the SBM dataset with 1000 nodes. The accuracy trends for both training and testing are shown for the various configurations. [Figure 5f]



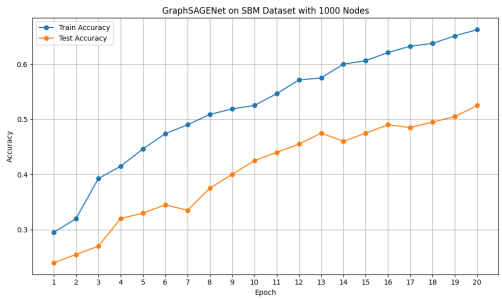
(a) GCN on SBM Dataset.



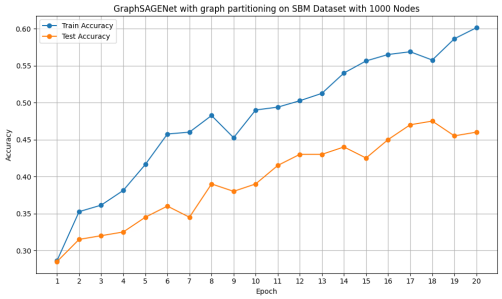
(b) GCN with Graph Partitioning on SBM Dataset.



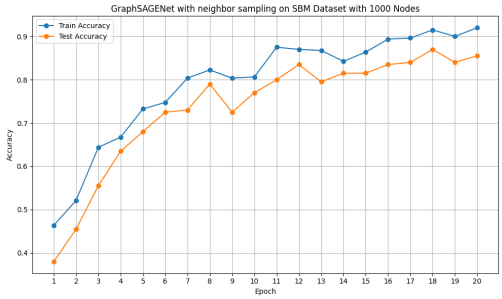
(c) GCN with Neighbor Sampling on SBM Dataset.



(d) GraphSAGE on SBM Dataset.



(e) GraphSAGE with Graph Partitioning on SBM Dataset.

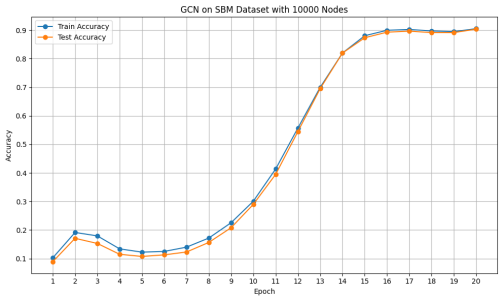


(f) GraphSAGE with Neighbor Sampling on SBM Dataset.

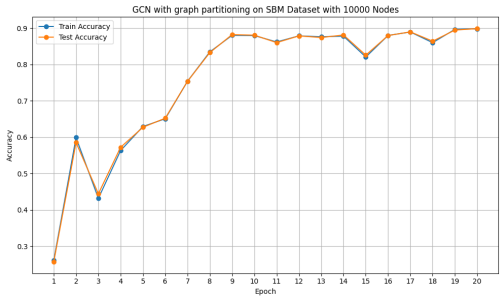
Figure 5: Performance comparison on the SBM dataset with 1000 nodes for GCN and GraphSAGE models under different training strategies: Full-Batch, Graph Partitioning, and Neighbor Sampling.

4.4.7 Performance on SBM Dataset (10,000 Nodes)

This subsection compares the performance of different models and training methods on the SBM dataset with 10,000 nodes. The accuracy trends for both training and testing are shown for the various configurations. [Figure 6f]

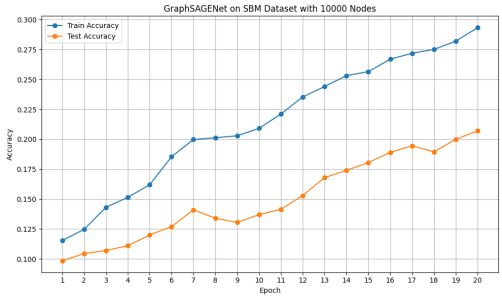


(a) GCN on SBM Dataset.

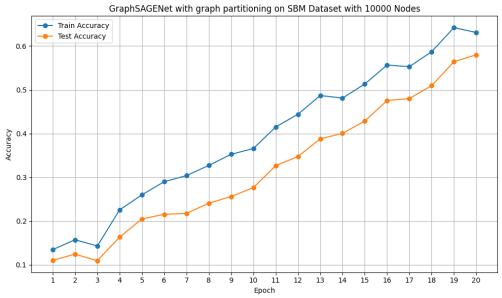


(b) GCN with Graph Partitioning on SBM Dataset.

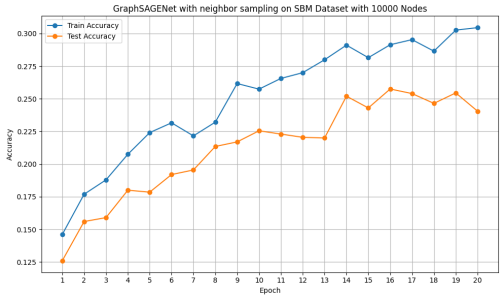
(c) GCN with Neighbor Sampling on SBM Dataset.



(d) GraphSAGE on SBM Dataset.



(e) GraphSAGE with Graph Partitioning on SBM Dataset.



(f) GraphSAGE with Neighbor Sampling on SBM Dataset.

Figure 6: Performance comparison on the SBM dataset with 10,000 nodes for GCN and GraphSAGE models under different training strategies: Full-Batch, Graph Partitioning, and Neighbor Sampling.

4.4.8 Performance Summary Tables

Table 1: Performance on SBM Dataset with 1,000 Nodes.

Model	Method	Test Accuracy	Train Time (s)
GCNet	Full Batch	70%	0.19
	Neighbor Sampling	67%	3.32
	Graph Partitioning	50.5%	0.58
GraphSAGENet	Full Batch	52.5%	0.13
	Neighbor Sampling	85.5%	3.17
	Graph Partitioning	46%	0.65

SBM Dataset with 1,000 Nodes

Table 2: Performance on SBM Dataset with 10,000 Nodes.

Model	Method	Test Accuracy	Train Time (s)
GCNet	Full Batch	90%	12.66
	Neighbor Sampling	17.85%	44.62
	Graph Partitioning	89.85%	9.51
GraphSAGENet	Full Batch	20.7%	10.23
	Neighbor Sampling	24.05%	42.07
	Graph Partitioning	58%	8.40

SBM Dataset with 10,000 Nodes

Table 3: Performance on CORA Dataset.

Model	Method	Test Accuracy	Train Time (s)
GCNet	Full Batch	89.48%	0.19
	Neighbor Sampling	88.01%	4.86
	Graph Partitioning	87.08%	0.86
GraphSAGENet	Full Batch	90.41%	0.88
	Neighbor Sampling	88.56%	8.93
	Graph Partitioning	88.75%	1.58

CORA Dataset

Table 4: Performance on Reddit Dataset.

Model	Method	Test Accuracy	Train Time (s)
GCNet	Full Batch	N/A (Out of Memory)	N/A
	Neighbor Sampling	88.6%	1892.33
	Graph Partitioning	88.12%	1688.57
GraphSAGENet	Full Batch	N/A (Out of Memory)	N/A
	Neighbor Sampling	86.44%	13108.7
	Graph Partitioning	N/A (Out of Memory)	N/A

Reddit Dataset

4.5 Discussion

The results demonstrate the effectiveness of scalable GNN training methods. Neighbor Sampling strikes a balance between accuracy and computational efficiency, while Graph Partitioning offers the best memory optimization. These findings validate the proposed approaches for handling large-scale graphs and emphasize the importance of choosing the right method based on the dataset and computational constraints.

5 Conclusion

In conclusion, this project looks to overcome the limitations present in traditional community detection approaches by using advanced machine learning methods, specifically Graph Neural Networks (GNNs). Traditional methods, such as modularity-based and optimization methods, often struggle with generalizability and the ability to adapt to networks of varying sizes and structures. By focusing on both accuracy and scalability, this approach is aimed at making community detection more useful for large, complex networks.

This project not only applies theoretical ideas from the course but also shows how they can be used for real-world challenges in network analysis. Through this work, there's a chance to understand network structures better and how advanced machine learning models help improve

community detection. This paper showcases the potential of advanced deep learning models, such as GNNs, to optimize the solutions to the NP-Hard problem of 'Community Detection' in social networks.

References

[1] Yuecheng Li, Jialong Chen, Chuan Chen, Lei Yang, and Zibin Zheng. Contrastive deep nonnegative matrix factorization for community detection, 2024.

[2] Chunjiang Liu, Yikun Han, Haiyun Xu, Shihan Yang, Kaidi Wang, and Yongye Su. A community detection and graph neural network based link prediction approach for scientific literature, 2024.

[3] Xinyu Que, Fabio Checconi, Fabrizio Petrini, and John A. Gunnels. Scalable community detection with the louvain algorithm. In *2015 IEEE International Parallel and Distributed Processing Symposium*, pages 28–37, 2015.

[4] Yiye Ruan, David Fuhry, and Srinivasan Parthasarathy. Efficient community detection in large networks using content and links. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, page 1089–1098, New York, NY, USA, 2013. Association for Computing Machinery.

[5] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and Philip S. Yu. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):17–37, 2017.

[6] Gholam Reza Abdi, Amir Hosein Refahi Sheikhani, Sohrab Kordrostami, Bagher Zarei, and Mohsen Falah Rad. Identifying communities in complex networks using learning-based genetic algorithm. *Ain Shams Engineering Journal*, page 103031, 2024.

[7] S. Akbar and S. K. Saritha. Quantum inspired community detection for analysis of biodiversity change driven by land-use conversion and climate change. *Scientific Reports*, 11(1):14332, 2021.

[8] Mir Mohammad Alipour and Mohsen and Abdolhosseinzadeh. A multiagent reinforcement learning algorithm to solve the community detection problem. *Signal and Data Processing*, 19(1), 2022.

[9] Jiyang Chen, Osmar R. Zaïane, and Randy Goebel. *Detecting Communities in Social Networks using Max-Min Modularity*, pages 978–989.

[10] Bouchema Sara Cheikh Salmi and Zaoui Sara. An enhanced evolutionary approach for solving the community detection problem. *Journal of Information and Telecommunication*, 6(1):83–100, 2022.

[11] Fanzhen Liu, Shan Xue, Jia Wu, Chuan Zhou, Wenbin Hu, Cecile Paris, Surya Nepal, Jian Yang, and Philip S. Yu. Deep learning for community detection: progress, challenges and opportunities. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20, 2021.

[12] Ying Xie, Xinmei Wang, Dan Jiang, and Rongbin Xu. High-performance community detection in social networks using a deep transitive autoencoder. *Information Sciences*, 493:75–90, 2019.

[13] Mahsa Nooribakhsh, Marta Fernández-Diego, Fernando González-Ladrón-De-Guevara, and Mahdi Mollamotalebi. Community detection in social networks using machine learning: a systematic mapping study. *Knowledge and Information Systems*, 66(12):7205–7259, December 2024.

[14] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[15] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018.

[16] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD '19. ACM, July 2019.