Amirkabir University of Technology
(Tehran Polytechnic)

Department of Computer Engineering

Bachelor Thesis

Implementation and comparison of graph neural networks with learnable weight matrices and without learnable weight matrices on different tasks

By

Mohammadreza Rezaei

Supervisor

Dr. Mostafa Haghir Chehreghani

April 2025

Abstract

Graph Neural Networks (GNNs) are generally effective frameworks for learning from structured data. They use learnable weight matrices to aggregate node features and propagate information across the graph. However, some recent models have shown that removing these weight matrices can reduce computational complexity while maintaining performance. This study compares GNN architectures with and without learnable weight matrices, specifically GCN and LightGCN, across different tasks to analyze the impact of weights on feature aggregation, expressiveness, and computational efficiency. The models were evaluated on diverse tasks such as node classification and link prediction using standard performance metrics for analysis. The results help clarify the role of weight matrices in GNNs and assist in selecting appropriate architectures for various applications, including social networks and recommender systems.

Key words:

LightGCN, GCN, GNN

Introduction

Social networks, bioinformatics, transportation systems, and electronic commerce are examples of domains that contain structured data, which can be naturally represented as graphs consisting of nodes and edges. Learning effectively from such data requires models capable of capturing dependencies among connected entities. Graph Neural Networks (GNNs) were introduced as a powerful tool for this purpose since they can propagate messages between nodes and aggregate information from their neighbors through learnable parameters. In recent years, these networks have achieved remarkable success in various tasks such as node classification, link prediction, and molecular property prediction, as well as in higher-level applications like recommendation systems and reasoning over graph-structured information.

This thesis focuses on the implementation and comparison of GNN models with learnable weight matrices and without learnable weight matrices. Conventional GNN architectures, such as the Graph Convolutional Network (GCN), perform linear transformations of node features at each layer through weight matrices, followed by message passing and non-linear activation. However, recent studies have demonstrated that simplified architectures that remove weight matrices and non-linear functions can still perform competitively. A notable example is LightGCN, which preserves the essential message-passing mechanism while discarding unnecessary components such as feature transformation and activation functions.

This raises an important question: to what extent do learnable weight matrices contribute to GNN performance, and can their removal reduce computational complexity without degrading accuracy? To explore this, the present project compares standard GNNs that include trainable weights (for example, GCN) with their simplified counterparts (for example, LightGCN) across tasks such as node classification and link prediction.

The experiments are conducted on well-known benchmark datasets including Cora, Citeseer, PubMed, and Amazon Photo, with the goal of analyzing the trade-off between model expressiveness, computational efficiency, and overfitting risk. The findings from this research offer insights into how weight matrices influence GNN behavior and provide guidance for selecting appropriate architectures for various applications such as social network analysis, biological interaction modeling, and recommendation systems.

Related Work

1. **Graph Neural Network (GNN) – Scarselli et al. (2009)**
   Scarselli and his colleagues introduced the first general framework for Graph Neural Networks in 2009. Their model iteratively propagated messages between connected nodes until reaching a stable state, allowing the network to learn node representations based on neighborhood information. This recurrent structure enabled modeling of arbitrary graph dependencies and became the theoretical foundation for later GNN research.

2. **Graph Convolutional Network (GCN) – Kipf and Welling (2017)**
   Kipf and Welling proposed the Graph Convolutional Network as a simplified and computationally efficient variant of spectral graph convolutions. In GCN, each layer aggregates and normalizes features from neighboring nodes, applies a linear transformation, and then a non-linear activation function. This design achieved high accuracy on semi-supervised node classification tasks using citation networks such as Cora, Citeseer, and PubMed, and established GCN as a standard baseline for subsequent graph learning research.

3. **GraphSAGE – Hamilton et al. (2017)**
   Hamilton and colleagues introduced GraphSAGE, an inductive framework capable of learning node embeddings for previously unseen nodes. Instead of processing the entire graph, GraphSAGE samples a fixed number of neighbors for each node and applies different aggregation strategies such as mean, pooling, or LSTM-based aggregation. This allowed the model to handle large-scale and dynamic graphs efficiently, improving scalability and adaptability across different applications.

4. **Message Passing Neural Network (MPNN) – Gilmer et al. (2017)**
   Gilmer et al. proposed the Message Passing Neural Network as a unifying framework that formalized GNN architectures through two main functions: message computation and node update. Their work showed that many existing models, such as GCN and GraphSAGE, could be described as special cases of MPNN. This general formulation clarified the theoretical basis of GNNs and facilitated their application in various domains, including quantum chemistry for molecular property prediction.

5. **Graph Attention Network (GAT) – Velickovic et al. (2018)**
   Velickovic and collaborators developed the Graph Attention Network, which introduced an attention mechanism into GNNs. Instead of treating all neighbors equally, GAT assigns a learnable attention coefficient to each edge, allowing the model to focus more on informative nodes during aggregation. This approach improved accuracy and interpretability, especially in semi-supervised learning tasks such as node classification on citation and protein–protein interaction datasets.

6. **Graph Isomorphism Network (GIN) – Xu et al. (2019)**
   Xu and his co-authors proposed the Graph Isomorphism Network, motivated by the Weisfeiler–Lehman graph isomorphism test. GIN employs a sum-based aggregation function combined with a multilayer perceptron, giving it strong expressive power for distinguishing non-isomorphic graph structures. It became one of the most powerful models for graph classification tasks and provided theoretical insights into the expressive limits of GNNs.

7. **Simplified Graph Convolution (SGC) – Wu et al. (2019)**
   Wu and colleagues presented Simplified Graph Convolution, which removed non-linear activation functions and combined multiple GCN layers into a single linear transformation. This simplification significantly reduced training time and memory consumption while preserving comparable performance. The model demonstrated that, in many cases, the key factor in GNN performance lies in the aggregation process rather than complex layer transformations.

8. **LightGCN – He et al. (2020)**
   He and co-authors introduced LightGCN, a minimal design tailored for recommendation systems. By removing both non-linear activation and feature transformation, LightGCN focuses solely on message propagation through graph structure. Despite its simplicity, it achieves competitive or superior results compared to more complex architectures, showing that efficient information flow is often sufficient for high-quality graph representation learning.

9. **Comprehensive Survey on GNNs – Wu et al. (2020)**
   In 2020, Wu and colleagues conducted a large-scale survey summarizing existing GNN architectures, optimization strategies, and application domains. Their review categorized models into spectral, spatial, and hybrid approaches, discussed their theoretical connections, and outlined open challenges in graph representation learning. This work provided a unified understanding of the field and served as a major reference for future developments.

10. **A Review on Graph Neural Networks – Zhou et al. (2020)**
    Zhou and co-authors published an extensive review covering the evolution, theoretical foundations, and applications of Graph Neural Networks. They analyzed message-passing mechanisms, training challenges such as over-smoothing and scalability, and major real-world use cases in social networks, chemistry, and natural language processing. Their work emphasized research directions for improving generalization, interpretability, and efficiency in GNN design.

Methodology

This chapter explains the detailed implementation process used in this research. Each model was trained on a specific dataset and followed a structured procedure consisting of preprocessing, model definition, training, and evaluation.

First, data preprocessing and loading were performed using the Cora dataset, which contains scientific publications as nodes and citation relationships as edges. Each publication's features were extracted from text attributes and stored as numerical vectors, while the citation links were used to form the adjacency matrix. Python libraries such as pathlib and scipy were used to read dataset files, generate efficient sparse graph representations, and normalize the adjacency matrix.

The next step involved defining the GCN model, composed of multiple key layers:

1. Input layer for node features,
2. Graph convolutional layer for local embedding extraction,
3. Dropout layer to prevent overfitting,
4. A second graph convolutional layer for refining embeddings,
5. Softmax layer for output classification probabilities.

The training process followed a standard pipeline:

1. Load dataset,
2. Normalize adjacency and feature matrices,
3. Initialize GCN model parameters,
4. Perform forward and backward propagation,
5. Evaluate the model by calculating loss and accuracy,
6. Visualize results for loss and accuracy trends.

The model was trained using the PyTorch framework with the Adam optimizer for dynamic parameter adjustment. Gradients were cleared in each iteration, losses were computed using the negative log-likelihood function, and accuracy was measured based on the predicted and true labels.

For systematic tuning and model configuration, the research integrated GraphGym, a specialized framework for automated exploration of architectures and hyperparameters in GNNs. GraphGym allows structured evaluation of different aggregation strategies, architectures, and learning setups, ensuring reproducible experiments by storing configurations in YAML and TXT files.

Finally, a simple user interface was developed to visualize the results interactively. Users could input dataset names (Cora, Citeseer, PubMed, Amazon Photo), select the model layer type (GCN, GAT, LightGCN), and view performance metrics such as accuracy, F1-score, precision, recall, and AUC-ROC directly.

In summary, this methodology section describes a comprehensive pipeline — from dataset preprocessing and model architecture design to training, evaluation, and visualization — implemented to compare GNNs with and without learnable weight matrices under consistent experimental conditions.

Results and Discussion

This chapter presents the outcomes of implementing and comparing three Graph Neural Network architectures: Graph Convolutional Network (GCN), Light Graph Convolutional Network (LightGCN), and Graph Attention Network (GAT). These models were tested on four well-known benchmark datasets — Cora, Citeseer, PubMed, and Amazon Photo — to evaluate their performance in both node classification and link prediction tasks. The goal was to analyze the effect of learnable weight matrices and attention mechanisms on accuracy, efficiency, and generalization capability.

Dataset Description

The Cora dataset consists of 2,708 scientific publications as nodes connected by 5,429 citation links. Each paper is represented by a 1,433-dimensional feature vector derived from its textual content and classified into seven research areas. The Citeseer dataset contains 3,327 papers and 4,732 citation links within six categories, while PubMed includes 19,717 medical publications and 44,338 links categorized into three topics. The Amazon Photo dataset represents a product co-purchasing graph with 7,650 products, 119,043 edges, and 745-dimensional feature vectors divided into eight categories.

Evaluation Metrics

Several key metrics were used to assess the models: accuracy, precision, recall, F1-score, and AUC-ROC. These collectively measure classification correctness, sensitivity, balance between precision and recall, and model robustness in distinguishing positive and negative samples.

Node Classification Results

The experiments on node classification revealed that the three models behaved differently across datasets. On Cora, GCN achieved slightly higher accuracy, benefiting from its feature transformation capability. GAT also achieved strong performance by assigning dynamic importance to neighboring nodes during aggregation. LightGCN, though simpler, produced competitive accuracy while requiring less computational cost. Similar patterns were observed on Citeseer, where GCN and GAT slightly outperformed LightGCN, but the differences were small compared to their parameter and runtime overhead. On Amazon Photo, LightGCN achieved accuracy comparable to GCN and GAT while training faster, showing that removing learnable weights does not necessarily reduce performance in large-scale graphs.

Link Prediction Results

For link prediction on Citeseer and PubMed, the F1-score and AUC values demonstrated similar trends. On Citeseer, GCN achieved an F1-score of 0.7574, GAT achieved 0.7556, and LightGCN achieved 0.7462. On PubMed, LightGCN achieved the best F1-score of 0.7911, outperforming both GCN (0.7764) and GAT (0.7678). In terms of AUC, GCN reached the highest score of 0.9077 on PubMed, followed by GAT (0.8942) and LightGCN (0.8789).

These results confirm that LightGCN, despite its simplicity, performs competitively against more complex models that use learnable weights or attention mechanisms.

Analysis of Performance and Efficiency

Plots of training loss and accuracy indicated that GCN and GAT models fluctuated more due to their parameter updates and non-linear activations, whereas LightGCN exhibited smoother convergence and lower variance. From a computational perspective, LightGCN consumed less memory and trained faster, owing to its minimal architecture without transformation layers. GAT, while achieving interpretability through attention coefficients, required higher computation time and GPU memory.

Summary of Findings

The overall comparison demonstrates that GCN and GAT perform slightly better on smaller, feature-rich datasets where expressive transformations are beneficial, while LightGCN excels in scalability and efficiency, maintaining high accuracy without learnable weights. These findings suggest that removing weight matrices can significantly reduce complexity and overfitting without harming model effectiveness, especially in large or sparse graphs dominated by structural relationships.

Conclusion and Future Work

This study analyzed the strengths and weaknesses of three models—GCN, GAT, and LightGCN—in the tasks of node classification and link prediction, presenting a balanced and evidence-based view derived from experimental results and scientific sources.

GCN demonstrated strong and stable performance in both node classification and link prediction tasks. It effectively utilized both node features and graph structure, providing a solid computational baseline with relatively low complexity. In most datasets, it achieved nearly the highest classification accuracy, differing by less than one percent from the top result, and exhibited exceptional AUC and recall values on the PubMed dataset.

GAT showed its main advantage in its ability to learn the relative importance of neighboring nodes through attention coefficients. In node classification experiments on the Cora and Amazon Photo datasets, GAT achieved the highest accuracy, slightly outperforming GCN. This indicates its effectiveness in identifying influential neighbors in graphs with high homophily. However, in heterophilous graphs, this advantage was smaller. In link prediction, GAT did not achieve the best F1 or AUC scores but still delivered competitive results. Its main drawbacks were its higher computational complexity and lower training stability due to attention weight calculations for each edge, especially in large graphs.

LightGCN, unlike GCN and GAT, does not use weight matrices in its layers. This model specializes in efficiently capturing structural information in graphs while relying less on node feature vectors. Its simpler design led to faster training and reduced overfitting, since it contained fewer parameters per layer. Although its feature learning capacity was limited compared to GCN and GAT, it performed well in link prediction and recommendation-like tasks where structural relations are more important than feature information. In several cases, LightGCN produced recall values as high as or higher than the other two models, confirming its ability to maintain strong predictive power with minimal complexity.

In summary, the results show that each model has distinct strengths: GCN serves as a reliable general-purpose baseline, GAT offers interpretability and adaptive neighbor weighting, and LightGCN provides computational efficiency and robust performance for large-scale or structure-dominant graphs.

For future work, it is suggested to use LightGCN-like models in large-scale graphs, especially in recommendation systems and social networks, where graph structure plays the most critical role. It is also recommended to explore hybrid approaches combining the advantages of attention mechanisms with lightweight architectures to achieve both efficiency and high accuracy.

References

[1] Gilmer, Justin; Schoenholz, Samuel S.; Riley, Patrick F.; Vinyals, Oriol; and Dahl, George E. *Neural message passing for quantum chemistry*., 2017.
https://proceedings.mlr.press/v70/gilmer17a.html

[2] Hamilton, William L.; Ying, Zhitao; and Leskovec, Jure. *Inductive representation learning on large graphs*., 2017.
https://arxiv.org/abs/1706.02216

[3] He, Xiangnan; Deng, Kuan; Wang, Xiang; Li, Yan; Zhang, Yongdong; and Wang, Meng. *LightGCN: Simplifying and powering graph convolution network for recommendation*., 2020.
https://arxiv.org/abs/2002.02126

[4] Kipf, Thomas N.; and Welling, Max. *Semi-supervised classification with graph convolutional networks*., 2017.
https://openreview.net/forum?id=SJU4ayYgl

[5] Scarselli, Franco; Gori, Marco; Tsoi, Ah Chung; Hagenbuchner, Markus; and Monfardini, Gabriele. *The graph neural network model*., 2009.
https://arxiv.org/abs/2009.11859

[6] Veličković, Petar; Cucurull, Guillem; Casanova, Arantxa; Romero, Adriana; Liò, Pietro; and Bengio, Yoshua. *Graph attention networks*., 2018.
https://arxiv.org/abs/1710.10903

[7] Wu, Felix; Souza, Amauri; Zhang, Tianyi; Fifty, Christopher; Yu, Tao; and Weinberger, Kilian Q. *Simplifying graph convolutional networks*., 2019.
https://proceedings.mlr.press/v97/wu19e.html

[8] Wu, Zonghan; Pan, Shirui; Chen, Fengwen; Long, Guodong; Zhang, Chengqi; and Yu, Philip S. *A comprehensive survey on graph neural networks*., 2020.
https://arxiv.org/abs/1901.00596

[9] Xu, Keyulu; Hu, Weihua; Leskovec, Jure; and Jegelka, Stefanie. *How powerful are graph neural networks?*, 2019.
https://openreview.net/forum?id=ryGs6iA5Km

[10] Zhou, Jie; Cui, Ganqu; Zhang, Zhengyan; Yang, Cheng; Liu, Zhiyuan; Wang, Lifeng; Li, Changcheng; and Sun, Maosong. *Graph neural networks: A review of methods and applications*., 2020.
https://arxiv.org/abs/1812.08434