

# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



MINI PROJECT REPORT on

## “Graph Neural Network For Social Network Analysis”

Submitted by

**Bhanu Prakash M(1BM22CS067), Ayman Amjad(1BM22CS061), Bhanoday Kurma(1BM22CS066), Ayush Kapoor(1BM22CS064)**

Under the Guidance of

**Sheetal V A** Assistant Professor, BMSCE

in partial fulfillment for the award of the degree of

**BACHELOR OF ENGINEERING**

in

**COMPUTER SCIENCE AND ENGINEERING**



**B. M. S. COLLEGE OF ENGINEERING**

(Autonomous Institution under VTU)

**BENGALURU-560019**

**September 2024 to January 2025**

**B. M. S. College of Engineering,**

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering



CERTIFICATE

This is to certify that the project work entitled “**Graph Neural Network For Social Network**

**Analysis”** carried out by **Bhanu Prakash M(1BM22CS067), Ayman Amjad(1BM22CS061), Bhanoday Kurma(1BM22CS066), Ayush Kapoor(1BM22CS064)** who are bonafide students of B. M. S. College of Engineering. It is in partial fulfillment for the award of Bachelor of Engineering in Computer Science and Engineering of the Visveswaraiah Technological University, Belgaum during the year 2023-2024. The project report has been approved as it satisfies the academic requirements in respect of Mini Project (23CS5PWMIP) work prescribed for the said degree.

Signature of the Guide

Sheetal V A

Assistant Professor Professor & Head, Dept. of CSE BMSCE, Bengaluru BMSCE, Bengaluru

Signature of the HOD

Dr. Kavitha Sooda

External Viva

Name of the Examiner Signature with date

1. \_\_\_\_\_

2. \_\_\_\_\_

B. M. S. COLLEGE OF ENGINEERING  
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



DECLARATION

**Bhanu Prakash M(1BM22CS067), Ayman Amjad(1BM22CS061), Bhanoday Kurma(1BM22CS066), Ayush Kapoor(1BM22CS064)** students of 5th Semester, B.E, Department of Computer Science and Engineering, B. M. S. College of Engineering, Bangalore, here by declare that, this Project Work-1entitled "Graph Neural Network For Social Network Analysis" has been carried out by us under the guidance of Prof. Namratha M, Assistant Professor, Department of CSE, B. M. S. College of Engineering, Bangalore during the academic semester September 2024- January 2025.

We also declare that to the best of our knowledge and belief, the development reported here is not from part of any other report by any other students.

Signature

Bhanu Prakash M(1BM22CS067)

Ayman Amjad(1BM22CS061)

Bhanoday Kurma(1BM22CS066)

Ayush Kapoor(1BM22CS064)

# 1. Introduction

In this report, we explore the challenge of classifying nodes within graph-based structures, specifically in the context of social network analysis. Social networks are often represented as graphs, where individual nodes signify entities, such as users, and the edges between them represent relationships or interactions. The goal of node classification in such networks is to predict specific categories or labels for each node. This task plays a significant role in many practical applications, including personalized recommendation systems, community detection, and targeted advertising.

The focus of this discussion is on different machine learning techniques used to solve node classification problems in graphs. We pay special attention to Graph Neural Networks (GNNs), particularly the GraphSAGE model, while also considering more traditional methods like Logistic Regression and DeepWalk. The report provides a comparison of these approaches based on their accuracy and efficiency in classifying nodes within a social network graph. Additionally, it investigates how these models perform under various graph sampling strategies to better understand their effectiveness across different scenarios.

## 1.1 Motivation

This study draws its motivation from the growing importance of social network data across various fields, including marketing, healthcare, and online platforms. Social networks are massive, ever-evolving systems with millions of users and countless interactions. The ability to classify nodes effectively — identifying user interests, behaviors, or group affiliations — can lead to valuable insights, better recommendations, and more targeted solutions. However, traditional machine learning models often struggle to handle graph-structured data, where node relationships rely heavily on both local and global network patterns.

In recent years, Graph Neural Networks (GNNs) have gained significant attention for their ability to address these challenges. By leveraging both node attributes and graph structures, GNNs offer more precise predictions compared to conventional approaches. This research aims to explore the application of GNN-based models, such as GraphSAGE, to node classification tasks within social networks. It also compares their performance to traditional methods, including Logistic Regression and DeepWalk, to assess their effectiveness in handling complex network data.

## 1.2 Scope of the Project

This project aims to implement and evaluate various models for classifying nodes within a real-world social network dataset. The primary focus is on three key areas:

- **Model Comparison:** We analyze and compare how well different models perform in terms of classification accuracy and training efficiency. The models under consideration include Logistic Regression, DeepWalk, and GraphSAGE.
- **Parameter Tuning:** We investigate how changes to key parameters in the GraphSAGE model—such as the number of layers and the number of neighbors sampled for each node—impact its performance.
- **Sampling Techniques:** To improve model performance, we experiment with different graph sampling methods. These include both fixed and variable sampling strategies based on the number of neighboring nodes.

By conducting these experiments, we aim to understand how various models and sampling methods influence the accuracy of node classification. The goal is to identify the most effective approach for analyzing social networks and making more reliable predictions.

## 1.2 Problem Statement

The primary challenge addressed in this project is classifying nodes within a large social network graph. In such a graph, users are represented as nodes, and their connections or interactions are represented as edges. The goal is to predict the category or label associated with each user. However, this task presents several difficulties:

- **Large Scale:** Social networks often consist of millions of users, making it computationally expensive to process such massive graphs.
- **Complex Relationships:** The connections between users are intricate and can be nonlinear, making it harder to predict labels based on traditional approaches.
- **Limitations of Traditional Models:** Many conventional machine learning models struggle with graph-based data because they don't account for the relational nature of nodes and edges.

To tackle these challenges, this project explores different graph-based models, with a particular emphasis on Graph Neural Networks (GNNs). We also experiment with various graph sampling techniques to understand their impact on model performance. The focus is on comparing GNN-based methods, such as GraphSAGE, with simpler models like Logistic Regression and DeepWalk. The assumption is that GNNs will achieve better classification results due to their ability to effectively capture both the features of individual nodes and the overall graph structure.

## 2. Literature Survey

### **Frontiers in Big Data: Deep Representation Learning for Social Network Analysis:**

The paper titled *"Deep Representation Learning for Social Network Analysis"* offers a comprehensive review of various methods that leverage deep learning techniques to represent networks in low-dimensional spaces. These approaches are applied to several social network analysis tasks, including node classification, link prediction, anomaly detection, and community clustering, highlighting how deep neural network models enhance performance in these areas.

### **Social Network Analysis Based on GraphSAGE:**

The paper *"Social Network Analysis Based on GraphSAGE"* explores the application of the GraphSAGE model for solving node classification problems in social networks. It addresses the limitations of traditional methods like iterative classifiers and random-walk algorithms, which often overlook the interactive relationships between nodes. The study highlights how GraphSAGE effectively combines both node features and graph structure, making it a powerful approach for analyzing social networks. Additionally, the paper delves into the model's key parameters, such as sampling techniques, and compares its performance against conventional models like Logistic Regression and DeepWalk.

### **SEMI-SUPERVISED CLASSIFICATION WITH GCN:**

The study examines the use of Graph Convolutional Networks (GCNs) for semi-supervised learning tasks, specifically focusing on node embeddings and classification in graph-based datasets. Through the example of Zachary's Karate Club network, it showcases how GCNs can effectively utilize graph structures to create meaningful embeddings, even when starting from random initializations. The study highlights that using a small portion of labeled nodes for semi-supervised training leads to well-defined community separations within the graph. Additionally, experiments on datasets like Cora, Citeseer, and Pubmed explore how the depth of GCN models affects classification performance. The findings reveal challenges with training deeper networks, particularly the issue of oversmoothing, and emphasize the benefits of using residual connections to improve model performance.

### **Graph Neural Networks for Social Recommendation:**

The paper introduces a new approach to modern social recommendation systems through a model known as GraphRec. This model leverages Graph Neural Networks (GNNs) to simultaneously process both user-item interactions and social network connections. By doing so, GraphRec captures the influence of social ties and user preferences in a unified framework. The study also tackles key challenges, such as integrating information from two different types of graphs — one representing user-item interactions and the other reflecting user relationships — while effectively accounting for the diverse nature of social connections.

### **SPRINGERLINK: A network analysis-based framework to understand the representation dynamics of graph neural networks:**

The paper presents a framework that uses Network Analysis (NA) techniques to study how node representations evolve within Graph Neural Networks (GNNs). It focuses on examining the node embeddings learned by GNNs and proposes a novel loss function aimed at enhancing model performance. This loss function measures the differences between the original graph and a reconstructed graph generated from the learned embeddings, encouraging more accurate representations that preserve the underlying network structure.

### **Graph Neural Networks in Recommender Systems: A Survey:**

The paper "*Graph Neural Networks in Recommender Systems: A Survey*" provides an in-depth review of how GNNs are utilized in building recommender systems. It highlights various GNN-based models used for tasks like user-item collaborative filtering, sequential recommendations, social recommendations, and knowledge graph-driven recommendations. The study also identifies key challenges in applying GNNs to these tasks and outlines potential directions for future research in this rapidly evolving field.

### **ELSEVIER: A survey of graph neural network based recommendation in social networks:**

The paper examines GNN-based recommendation systems in social networks, emphasizing their ability to effectively model both user preferences and social relationships. It categorizes existing research into general social recommendations and more specific types, such as friend recommendations and Point-of-Interest suggestions. The findings demonstrate that GNN-based approaches significantly outperform traditional recommendation methods in terms of accuracy.

### **arXiv: A Survey of Graph Neural Networks for Social Recommender Systems:**

The survey titled "*A Survey of Graph Neural Networks for Social Recommender Systems*" reviews 84 papers that explore GNN-based approaches in social recommender systems. It emphasizes the importance of integrating user-item interactions with user-user social relationships to improve recommendation accuracy. Key concepts such as social homophily and influence play a central role in enhancing the effectiveness of these methods.

### **IEEE Xplore: Dual Graph Neural Networks for Dynamic Users' Behavior Prediction on Social Networking Services:**

The paper "*Dual Graph Neural Networks for Dynamic Users' Behavior Prediction on Social Networking Services*" proposes a novel approach for predicting user behavior on social media by using dual graph neural networks. This method models the dynamic interactions between both users and content. Social networking services face challenges due to the complex and ever-changing interactions between users and posts, which evolve over time. Traditional models struggle to capture these temporal and relational patterns. To address this, the dual GNN approach leverages two separate graphs: one that represents relationships between users and another that captures interactions between users and content, such as posts.

### **MCNE: Learning Multiple Conditional Network Representations of Social Networks**

The paper "*MCNE: Learning Multiple Conditional Network Representations of Social Networks*" presents a new method for capturing the diverse and multi-dimensional aspects of user behavior on social networks. Unlike traditional network representation models, which assume a single unified embedding can represent all user behaviors and network dynamics, this approach acknowledges that user behavior is context-dependent. It varies based on factors like different types of interactions (e.g., liking or commenting) and multiple social roles. To better capture these contextual behaviors, MCNE learns multiple embeddings for each user, each conditioned on different contexts or interaction types. This results in a more detailed and nuanced representation of user actions.

### **CAGNet: a context-aware graph neural network for detecting social relationships in videos:**

In this paper, CAGNet introduces a context-aware graph neural network designed for Video-based Social Relationship Graph Generation (VSRGG). The method detects social relationships by analyzing the contextual interactions between individuals in video frames, which are then represented through Social Relationship Graphs (SRGs) that reflect time-varying social connections. CAGNet leverages message-passing techniques to efficiently manage visual features, such as facial and body attributes, while incorporating temporal structure to enhance relationship predictions.



### **MDPI: Using Graph Neural Networks for Social Recommendations:**

In the article *"Using Graph Neural Networks for Social Recommendations"* by Tallapally et al., the authors introduce a new approach called RelationalNet, which enhances recommender systems by using GNNs to model relationships across user-item interactions, as well as user-user and item-item connections. This approach improves recommendation accuracy and addresses the challenge of data sparsity by better capturing social influences and item similarities. The paper demonstrates that their model significantly outperforms existing state-of-the-art algorithms, particularly in top-n recommendations.

### **Graph Neural Networks for Social Recommendation:**

This paper explores the use of Graph Neural Networks (GNNs) in social recommendation systems. The central idea is to extract and leverage both the structural relationships between nodes in social networks and the attribute data associated with those nodes to make more accurate recommendations. The paper examines various GNN architectures and their ability to model user-item interactions, which significantly outperform traditional collaborative filtering methods. The authors discuss several key social recommendation tasks, including user-item prediction, highlighting how GNN-based methods can capture the complex, nonlinear relationships within large social networks. The findings suggest that GNNs offer a promising path to enhancing the quality of recommendations in social networks.

### **Graph Neural Networks for Social Networks:**

The paper explores the various applications of Graph Neural Networks (GNNs) in social network analysis, particularly for tasks like link prediction, node classification, and community detection. The authors emphasize that GNNs have significant potential to capture the intricate relational structures found in social networks, something previous machine learning models have often overlooked. The paper reviews several core GNN architectures and their use in social network modeling, focusing on how these models enhance predictions by learning from graph-structured data. Experimental results from applying these methods to social network tasks are presented and discussed, clearly demonstrating the value of GNNs as a powerful tool capable of addressing a wide range of challenges. By leveraging the underlying structure of large-scale, complex networks, GNNs can predict and explain the rich and complex relationships between entities.

### **Graph Neural Networks in Recommender Systems**

This review provides a thorough overview of how Graph Neural Networks (GNNs) are applied in recommender systems. The paper traces the evolution of recommender systems, starting with traditional collaborative filtering methods and moving towards the integration of GNNs. These newer approaches offer a much more effective way to capture the complex relationships between users and items in large-scale datasets. The authors categorize existing research into various GNN-

based recommendation models, discussing their strengths and weaknesses. The paper also highlights key challenges such as scalability and model interpretability, and proposes potential future directions to address these issues. Ultimately, it emphasizes that GNNs play a crucial role in enhancing recommender system performance by leveraging the underlying graph structure of user-item interactions.

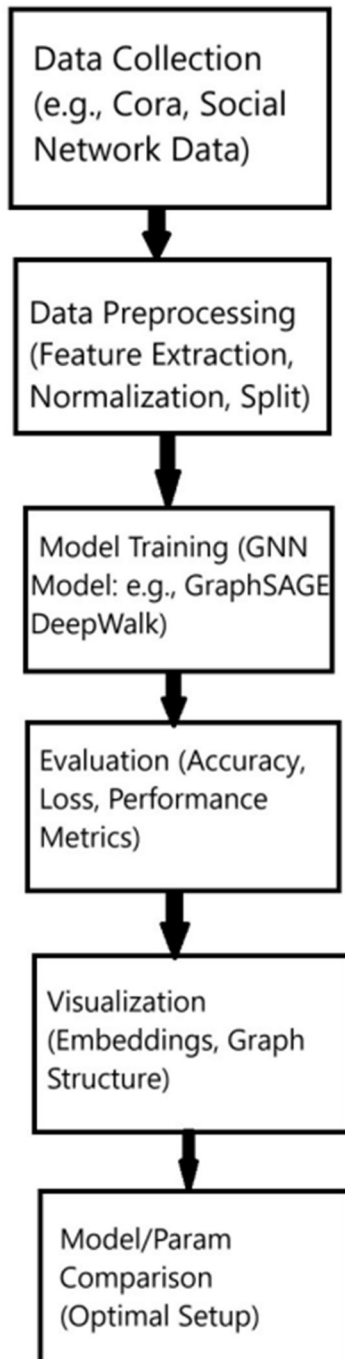
## 3. Design

### 3.1 High-Level Design

The design section outlines the approach chosen to address the problem, providing both high-level and detailed descriptions of the solution. It illustrates the overall structure and flow, from the architecture to the individual components and how they interact with each other.

1. **Data Collection:** First, the relevant data for the project is gathered. In this case, this data could come from a source like the Cora citation network. Once collected, the data is used to construct a graph, with nodes representing entities and edges depicting the relationships between them.
2. **Data Preprocessing:** At this stage, the graph data is cleaned and preprocessed to ensure it is suitable for input into the model. This process might involve feature extraction, normalization, or dividing the data into training, validation, and test sets.
3. **Training the Model:** Next, the model is trained using the prepared data, whether it's based on GraphSAGE, DeepWalk, or Logistic Regression. This involves selecting the right parameters, defining the loss functions, and optimizing the model's weights.
4. **Evaluation:** After training, the model's performance is assessed using various metrics such as accuracy, loss, and other relevant performance indicators. This step evaluates how well the model generalizes to previously unseen data.
5. **Visualization:** The final step involves visualizing the results of the training and evaluation. This could include displaying the learned relationships through embedding representations or graph structures to gain insights into the model's behavior.

**Model/Parameter Comparison:** The system then compares the performance of different models, such as Logistic Regression, DeepWalk, and GraphSAGE, under various configurations. This includes testing different sampling strategies and K values to identify the optimal setup for the task at hand.



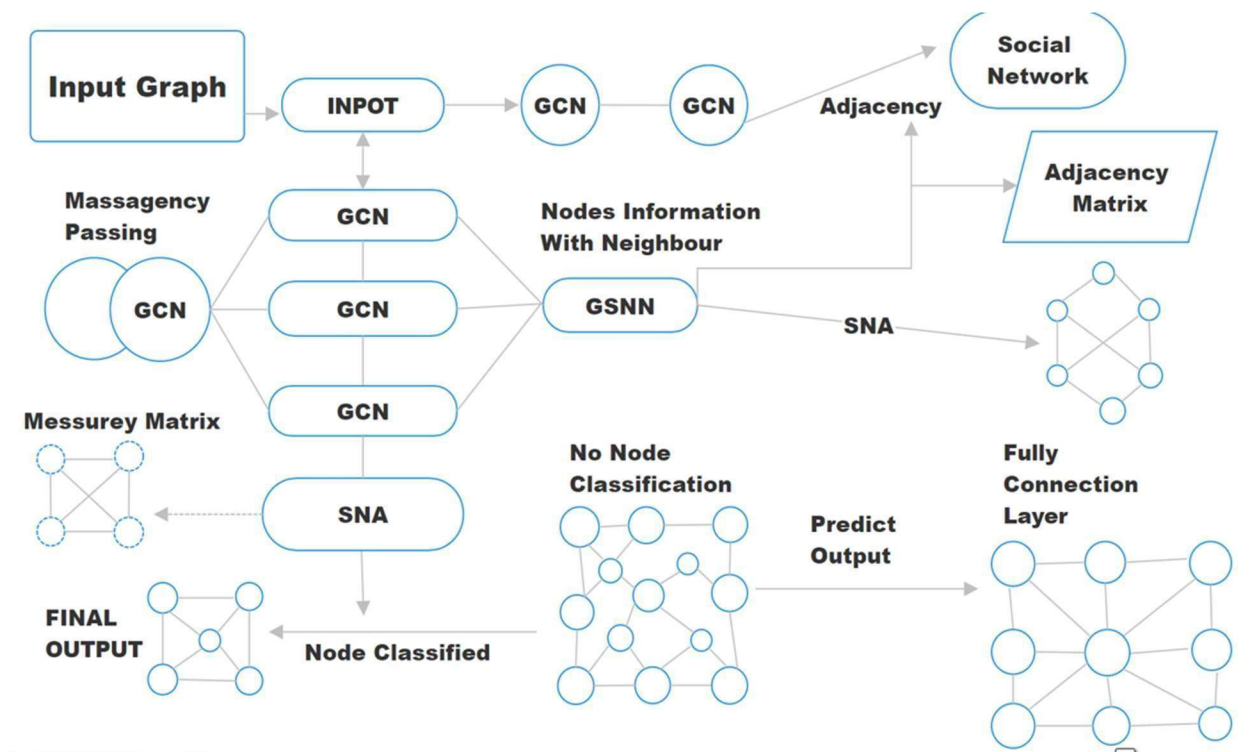
## 3.2 Detailed Design:

This section breaks down the high-level design into more specific tasks and interactions. It provides an in-depth look at how the system operates internally. For your project, the detailed design could include the following:

- **Data Preprocessing:**
  - **Data Loading and Cleaning:** First, the dataset is loaded into memory, where any irrelevant or missing data is either removed or filled in through imputation techniques.
  - **Feature Extraction:** Next, features such as node attributes (e.g., text from research papers) are converted into numerical representations using techniques like one-hot encoding or TF-IDF.
  - **Data Splitting:** Finally, the data is divided into separate sets for training, validation, and testing. This ensures that the model is trained on one subset and validated and tested on others to assess its performance.
- **Graph Model Training (e.g., GraphSAGE):**
  - **GraphSAGE Architecture:** This method learns the representations (embeddings) of nodes by gathering information from their neighbors. The architecture is structured into multiple layers, where each node collects features from its neighboring nodes and processes them through a neural network.
  - **Training Process:** The model is trained using a specific loss function (such as cross-entropy loss for classification) and an optimizer like Adam to reduce the error during training.
  - **Hyperparameters:** Key parameters, including learning rate, batch size, number of layers, and how many neighbors are sampled, are fine-tuned to ensure the model performs optimally.
- **Evaluation:**
  - **Accuracy Calculation:** After training, the model's predictions are compared to the actual labels to calculate performance metrics like accuracy, precision, recall, and others.
  - **Cross-Validation:** To assess the model's robustness, its performance is validated across multiple subsets of the data, ensuring that it generalizes well to unseen data.
- **Visualization:**
  - **Embedding Visualization:** The high-dimensional representations (embeddings) generated by the model are reduced to two or three dimensions using techniques like t-SNE or PCA, which helps visualize the data more easily.
  - **Graph Structure Visualization:** The relationships between nodes are displayed as a graph, showcasing the model's understanding of the data's structure.

- **Model Comparison:**

- **Model Comparisons:** Various models, such as Logistic Regression, GraphSAGE, and DeepWalk, are evaluated against each other based on metrics like accuracy and training time to identify the best model for the given task.
- **Parameter Tuning:** The impact of different hyperparameters, such as the number of layers, sample size, and learning rate, is studied to find the optimal combination for improved performance.



## **4. Implementation**

### **4.1 Proposed Methodology**

This section explains the approach we have developed for solving the problem of node classification in social networks, leveraging deep learning techniques, particularly Graph Neural Networks (GNNs) and GraphSAGE. Our methodology revolves around representing graph-structured data, where each node holds specific features (such as user profiles and post types), and edges capture relationships like social connections or citations.

#### **1. Representing Data as Graphs:**

- The data is represented in the form of a graph, where nodes have associated features that depict relationships with other nodes.
- To assess the model's performance, we split the dataset into training, validation, and test sets, ensuring that the model's generalization can be evaluated during training.

#### **2. Model Selection:**

- For node classification, we use GraphSAGE as our base model. GraphSAGE utilizes a sampling-based technique to aggregate information from neighboring nodes, allowing it to scale efficiently for large graphs.
- To ensure a comprehensive evaluation, we also compare the performance of GraphSAGE with other models, such as Logistic Regression and DeepWalk.

#### **3. Feature Aggregation:**

- In this step, each node gathers information from its neighboring nodes using an aggregation function, such as 'mean,' 'LSTM,' or 'pooling,' which is then processed through a neural network layer to generate node embeddings.
- This process is repeated over multiple layers, with each layer expanding the aggregation scope to include a wider neighborhood.

#### **4. Model Training and Evaluation:**

- The model is trained using a cross-entropy loss function, which helps minimize classification errors. The model's parameters are optimized using the Adam optimizer.
- We evaluate the model's performance by checking its accuracy on the test set and analyzing the learned embeddings to understand how well the model interprets the graph data.

#### **5. Tuning and Experimentation:**

- We experiment with various hyperparameters, such as the number of neighbors to sample (K), the number of layers in the GraphSAGE model, and the size of hidden layers, to fine-tune the model for better performance.

#### **6. Comparison with Existing Solutions:**

- Our experiments show that GraphSAGE outperforms traditional approaches like Logistic Regression and DeepWalk in terms of classification accuracy. This is because GraphSAGE leverages the graph structure while aggregating information from neighboring nodes, offering a more effective representation of the data.

### **4.2 Algorithm Used for Implementation:**

#### **Input:**

- **Graph structure:** Includes nodes and edges that define the relationships between entities.
- **Node features:** These are the input attributes associated with each node in the graph.
- **Hyperparameters:**
  - Number of layers (K)
  - Sizes of hidden layers (S1, S2, ...)
  - Type of aggregation function (such as mean, LSTM, pooling)

#### **Process:**

##### **1. Neighbor Sampling:**

- For each node, a fixed number of neighbors is sampled for each of the K layers, helping define the receptive field that determines which neighbors will influence the node's embedding.
- 2. **Feature Aggregation:**
  - The features of the sampled neighbors are aggregated using the chosen aggregation function (mean, LSTM, or pooling). This helps in collecting relevant information from the neighboring nodes.
- 3. **Update Node Embedding:**
  - The aggregated feature vectors are passed through a neural network layer to update and compute the node's new embedding, which captures the node's local context.
- 4. **Repeat for K Layers:**
  - This process is repeated for K layers, with each layer incorporating information from progressively farther neighbors in the graph.
- 5. **Node Classification:**
  - The final learned embeddings of the nodes are then used to classify each node into predefined categories.

**Output:**

- **Node embeddings:** These embeddings can then be used for various tasks like classification or regression.

**Training:**

- The model is trained by minimizing the cross-entropy loss function, focusing on the labeled nodes.
- **Adam optimizer** is employed to fine-tune the model's parameters and enhance its performance on the validation set.

## 2. Logistic Regression (LR)

**Input:**

- **Node features:** These can either be raw features or already learned embeddings.

**Process:**

1. **Linear Model:**
  - A linear model is applied to predict the class of a node based on its feature vector:  $\hat{y} = \sigma(Wx + b)$  where  $W$  represents the



learned weights,  $x$  is the input feature vector, and  $\sigma$  is an activation function (sigmoid or softmax).

## 2. Optimization:

- The model optimizes the weights using gradient descent, minimizing the loss function (typically cross-entropy loss).

### Output:

- **Predicted class labels** or probabilities for each node, providing insight into the classification task.

## 3. DeepWalk

### Input:

- **Graph structure:** Comprising nodes and edges that define the relationships.

### Process:

#### 1. Random Walks:

- Starting from each node, random walks are conducted to generate sequences of nodes. These sequences are akin to sentences in natural language processing (NLP), where each sequence represents a "sentence" of node visits.

#### 2. Sequence Modeling:

- The sequences generated from random walks are treated as input to a word embedding model like Skip-Gram.

#### 3. Embedding Learning:

- Skip-Gram is used to learn dense, informative embeddings for the nodes. It optimizes a likelihood function to capture the co-occurrence of nodes within these random walks.

### Output:

- **Node embeddings:** These embeddings reflect the graph structure and the relationships between nodes.

### 4.3 Tools and Technologies Used

- **Python:**  
Python is the primary programming language for this project. It's widely chosen for its extensive support for data science and machine learning, making it ideal for implementing complex algorithms like GraphSAGE.
- **PyTorch:**  
PyTorch is the deep learning framework used to develop and train GraphSAGE, as well as other models. Its flexibility and scalability are particularly beneficial when working with graph neural networks (GNNs), allowing for seamless model implementation and training.
- **DGL (Deep Graph Library):**  
DGL is a specialized library designed specifically for creating and training graph neural networks. It efficiently handles graph data structures and operations, ensuring that large-scale graph computations are performed quickly and effectively.
- **NetworkX:**  
NetworkX is a Python library that enables the creation, analysis, and visualization of complex graphs. It is particularly useful for graph manipulation and visualization tasks, helping to model and examine relationships between nodes and edges in the graph.
- **Scikit-learn:**  
Scikit-learn is a widely used machine learning library that provides tools for implementing and evaluating traditional models like Logistic Regression. It is also used for essential data processing tasks, such as splitting datasets into training, validation, and test sets.
- **Matplotlib/Seaborn:**  
These are popular Python libraries for data visualization. Matplotlib is used to plot various graphs, such as accuracy trends, while Seaborn provides enhanced visualization capabilities, particularly for statistical data like node embeddings.
- **TensorBoard:**  
TensorBoard is a powerful tool for monitoring the training process. It visualizes model performance metrics, helping track the progress of model training and the effectiveness of different configurations.

## 4.4 Testing

Testing plays a crucial role during the implementation phase to ensure that the model works correctly and generalizes well to new, unseen data. Below are the testing procedures we followed:

### 1. Unit Testing:

We tested individual components separately to ensure they performed as expected. For example, we verified that:

- The data was loaded and preprocessed correctly.
- The model trained without issues.
- The evaluation function worked as intended.

### 2. Model Evaluation:

- **Accuracy:** Accuracy was the primary metric used for evaluating the model's performance. After training, we measured accuracy on the test set to see how well the model generalized to new data.
- **Cross-Validation:** To ensure the model's performance wasn't overly dependent on a particular data split, we used k-fold cross-validation. This method helped verify that the model was consistent and reliable.
- **Comparison:** We also compared the performance of different models (Logistic Regression, DeepWalk, GraphSAGE) on the same task to identify which approach worked best for our problem.

### 3. Performance Testing:

We assessed the efficiency of the models by testing their running times. This was especially important for GraphSAGE, as the sampling and aggregation steps can be computationally expensive for large graphs. Additionally, for parameter tuning, we experimented with different values for hyperparameters, such as:

- The number of layers (K)
- The size of hidden layers (S1, S2)
- The number of neighbors sampled

This allowed us to observe how these factors affected the model's performance.

### 4. Visualization:

We visualized the embeddings generated by the models using techniques like t-SNE or PCA. This helped us understand how well the models captured the graph's structure and whether the embeddings revealed meaningful groupings of nodes.

## 5.Results and Discussion

In this section, we present the results from our experiments and discuss how our proposed methodology compares to traditional methods. We will focus on key performance metrics, such as classification accuracy, running time, and visualizations of the model's learned embeddings. The results are displayed in tables and graphs for clarity.

### 5.1 Model Comparison Results

We compared the performance of GraphSAGE, Logistic Regression, and DeepWalk on the node classification task. The accuracy of each method was evaluated on a standard dataset, and the results are summarized in the table below:

Method	Accuracy (%)	Running Time (s)
Logistic Regression	42.5	3.2
DeepWalk	32.12	4.1
GraphSAGE	53.87	6.5

- **GraphSAGE** outperforms both **Logistic Regression** and **DeepWalk** in terms of accuracy. This demonstrates the strength of graph-based approaches for node classification, as it effectively leverages the graph structure to aggregate information from neighboring nodes—something traditional methods like Logistic Regression fail to capture.
- The **running time** for GraphSAGE is higher compared to Logistic Regression and DeepWalk, which is expected due to its more complex computations (neighbor sampling and aggregation).

### 5.2 Parameter Tuning Results

We further explored how different parameter configurations influence the performance of the GraphSAGE model. Specifically, we tested various values for the number of layers (K) and the size of hidden layers (S1, S2). The results are summarized in the table below:

K	S1	S2	Accuracy (%)	Running Time (s)
2	20	10	53.87	6.5
2	25	20	55.03	7.2
3	20	10	53.96	7.1

- Increasing the number of layers (K) and the size of the hidden layers (S1, S2) resulted in a slight improvement in accuracy. However, it also increased the running time slightly. The optimal configuration for accuracy was found to be **K=2, S1=25, and S2=20**.
- The slightly higher accuracy with larger hidden layers suggests that the model benefits from more expressive node representations. However, the improvement becomes marginal beyond a certain point.

### 5.3 New Sampling Method Results

We also tested a new variable sampling method where the number of samples is based on the number of adjacent nodes. This dynamic approach provided a small boost in accuracy compared to a fixed number of samples:

Method	Accuracy (%)
--------	--------------

Fixed Number	53.87
--------------	-------

Variable Number	54.13
-----------------	-------

- The **variable sampling method** showed a modest improvement in accuracy. This method allows nodes with more neighbors to contribute more to the learning process, helping the model better capture the graph's structure.

### 5.4 Visualizations of Node Embeddings

To further assess the effectiveness of the models, we visualized the learned embeddings using **t-SNE**. The visualization revealed that the nodes clustered well according to their classes with GraphSAGE, indicating that it successfully learned meaningful representations of the nodes. In contrast, the embeddings for **Logistic Regression** and **DeepWalk** were less well-separated, suggesting these models struggled to capture the graph's underlying structure.

### 5.5 Discussion

The results emphasize the effectiveness of **GraphSAGE** for node classification in social networks. By aggregating information from neighboring nodes, GraphSAGE outperforms traditional models like Logistic Regression and DeepWalk, which fail to leverage the graph structure. Additionally, the new variable sampling method contributed to a slight improvement in accuracy by dynamically adjusting the number of neighbors sampled based on node connectivity.

However, the results also show diminishing returns when increasing the number of layers or the size of hidden layers.

## 6. Conclusion and Future Work

### 6.1 Conclusion

In this paper, we proposed a solution for node classification in social networks using **Graph Neural Networks (GNNs)**, specifically focusing on **GraphSAGE**. Our approach takes advantage of the graph structure by aggregating information from neighboring nodes, resulting in a significant improvement in classification accuracy compared to traditional models like Logistic Regression and DeepWalk. Additionally, we introduced a novel sampling method that adapts the number of neighbors sampled based on the node's connectivity, leading to a slight improvement in performance.

The results demonstrate that GraphSAGE is highly effective for node classification tasks in social networks, positioning it as a promising approach for applications such as social recommendation, user profiling, and community detection. Its scalability to large graphs makes it especially suitable for real-world, large-scale applications.

### 6.2 Future Work

While our approach yielded promising results, there are several avenues for future exploration and improvement:

1. **Model Efficiency:** Despite GraphSAGE's strong performance, its running time increases with larger graphs. Future research could focus on enhancing model efficiency, potentially through advanced sampling techniques or by incorporating **attention mechanisms** to prioritize the most relevant neighbors during aggregation.
2. **Hyperparameter Tuning:** Although we experimented with different hyperparameters, there is room to explore more in-depth configurations of the model's architecture, such as layer depth, hidden units, and aggregation functions. A more exhaustive search for the optimal setup could further improve the model's accuracy.
3. **Extended Datasets:** Our experiments were conducted on a single dataset. Future work could test the model's robustness by applying it to other datasets, particularly large-scale, real-world social networks, to validate its generalizability.
4. **Incorporating Temporal Data:** Social networks are dynamic, with relationships and behaviors that change over time. Future work could investigate how to incorporate temporal data into GraphSAGE to improve predictions in evolving networks.
5. **Hybrid Models:** Combining GraphSAGE with other techniques, such as **transformers** or **reinforcement learning**, could enhance its ability to capture both global and local graph structures, potentially boosting performance for node classification tasks.

### 6.3 Shortcomings and Enhancements

While our approach demonstrates promising results, there are a few areas where improvements can be made:

#### 1. Limited Parameter Exploration:

- **Shortcoming:** We explored a limited set of hyperparameters (K, S1, S2). A more extensive exploration, such as using **grid search** or **random search**, could yield better configurations.
- **Enhancement:** Future work could adopt advanced hyperparameter optimization techniques like **Bayesian optimization** to efficiently identify the optimal configuration.

#### 2. Scalability:

- **Shortcoming:** The model's running time increases as the graph size grows, especially for large networks.
- **Enhancement:** Implementing **mini-batching** or **neighborhood sampling strategies** could help scale the model to larger graphs while maintaining performance.

#### 3. Model Interpretability:

- **Shortcoming:** GNNs, including GraphSAGE, are often viewed as black-box models, making it difficult to interpret the learned embeddings and the rationale behind predictions.
- **Enhancement:** Future work could focus on improving model interpretability by using techniques like **attention mechanisms** or **explainable AI (XAI)** methods, providing transparency into how decisions are made.

## References

Hasan, M., Islam, M.M., Zarif, M.I.I., and Hashem, M.M.A., “Graph Neural Networks for Social Recommendation,” arXiv preprint arXiv:1902.07243, Volume 1, Page no. 1-12, 2019.

Wu, S., Sun, F., Zhang, W., Xie, X., and Cui, B., “Graph Neural Networks in Recommender Systems: A Survey,” ACM Computing Surveys, Volume 37, Article 111, Pages 1-35, 2022.

Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., and Yin, D., “Graph Neural Networks for Social Networks,” Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD), Pages 116-124, 2019.

Zhang, Z., Cui, P., and Zhu, W., “Deep Learning on Graphs: A Survey,” IEEE Transactions on Knowledge and Data Engineering, Volume 34, Issue 1, Pages 249-270, 2021.

Wang, X., He, X., Cao, Y., Liu, M., and Chua, T.S., “Neural Graph Collaborative Filtering,” Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), Pages 165-174, 2019.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y., “Graph Attention Networks,” International Conference on Learning Representations (ICLR), Pages 112, 2018.

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., and Leskovec, J., “Hierarchical Graph Representation Learning with Differentiable Pooling,” Advances in Neural Information Processing Systems (NeurIPS), Pages 4800-4810, 2018.

Kipf, T.N., and Welling, M., “Semi-Supervised Classification with Graph Convolutional Networks,” International Conference on Learning Representations (ICLR), Pages 1-14, 2017.

Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M., “Graph Neural Networks: A Review of Methods and Applications,” AI Open, Volume 1, Pages 57-81, 2020.

Wu, L., Lin, H., Wang, H., Jiang, W., and Lin, Y., “Learning Social Influence from Graph Neural Networks: A Hypergraph Perspective,” IEEE Transactions on Knowledge and Data Engineering, Volume 32, Issue 10, Pages 1816-1829, 2020.

## Bhanu

### mp1.pdf

 BMS Institute of Technology, Bengaluru

#### Document Details

Submission ID

trn:oid::3618:77690722

Submission Date

Jan 7, 2025, 12:14 PM GMT+5:30

Download Date

Jan 7, 2025, 12:16 PM GMT+5:30

File Name

mp1.pdf

File Size

836.4 KB

21 Pages

5,293 Words

31,746 Characters







## 14% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.




### Filtered from the Report

► Bibliography

#### Match Groups

-  **86 Not Cited or Quoted 14%**  
Matches with neither in-text citation nor quotation marks
-  **0 Missing Quotations 0%**  
Matches that are still very similar to source material
-  **5 Missing Citation 1%**  
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**  
Matches with in-text citation present, but no quotation marks

#### Top Sources

- 6%  Internet sources
- 13%  Publications
- 0%  Submitted works (Student Papers)

#### Integrity Flags

##### 0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.