A Hybrid Similarity-Aware Graph Neural Network with Transformer for Node Classification

MS(Research) Thesis

By

Aman Singh

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

May 2025

A Hybrid Similarity-Aware Graph Neural Network with Transformer for Node Classification

A THESIS

submitted to the

INDIAN INSTITUTE OF TECHNOLOGY INDORE

in fulfillment of the requirements for

the award of the degree

of

MS(Research)

By

Aman Singh

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

May 2025

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **A Hybrid Similarity-Aware Graph Neural Network with Transformer for Node Classification** in the partial fulfillment of the requirements for the award of the degree of **MS(Research)** and submitted in the **Department of Computer Science and Engineering, Indian Institute of Technology Indore,** is an authentic record of my work carried out during the period from August 2023 to May 2025. The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

[illegible] 14/05/2025

Signature of the Student with Date
(Aman Singh)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

[illegible]

Signature of Thesis Supervisor with Date 14/05/2025
(Dr. Ranveer Singh)

(Dr. Nagendra Kumar) Nagendra Kumar 14/05/2025

Aman Singh has successfully given his MS(Research) Oral Examination held on ________________

[illegible]

Signature of Head of Discipline

Date: 14/05/2025

ACKNOWLEDGEMENTS

I am deeply grateful for the opportunity to extend my sincere appreciation to several individuals whose contributions have made this journey both enriching and manageable. First, I would like to thank my supervisors **Dr. Ranveer Singh** and **Dr. Nagendra Kumar's** constant source of inspiration during my work. Without their constant guidance and research directions, this research work could not have been completed. Their continuous support and encouragement have motivated me to remain streamlined in my research work.

I would also like to acknowledge **Shahid Shafi Dar**, PhD Scholar at IIT Indore, for guiding me to do this dissertation. Without his precious support, it would not be possible to conduct this project.

My sincere acknowledgement and respect to **Prof. Suhas S. Joshi**, Director, Indian Institute of Technology Indore for providing me the opportunity to explore my research capabilities at Indian Institute of Technology Indore.

My deepest gratitude goes to my family and friends for their unwavering love and support throughout this process. Their encouragement and understanding during challenging times were invaluable.

Lastly, I extend my thanks to all those who have directly or indirectly contributed, assisted, and supported me on this path of academic pursuit.

Aman Singh

Abstract

Node classification has emerged as a critical task in graph deep learning, with diverse real-world applications such as recommendation systems, drug discovery, and citation networks. Although Graph Convolutional Networks and Graph Transformers have shown strong performance in this domain, they face inherent limitations. GCNs suffer from over-squashing, which restricts their ability to model long-range dependencies, while Graph Transformers encounter scalability issues when applied to large graphs. To address these challenges, we propose SIGNNet, A Hybrid **SI**milarity-aware **G**raph **N**eural **Net**work with Transformer for Node Classification. SIGNNet effectively captures both local and global structural information, enhancing the model's ability to learn fine-grained relationships and broader contextual patterns in graph data. The framework combines GCNs with a score-based similarity mechanism to improve local and global node interaction modeling while mitigating the effects of over-squashing. To tackle scalability, we introduce a Personalized PageRank-based node sampling strategy that enables efficient subgraph generation. Additionally, SIGNNet incorporates a novel Structure-Aware Multi-Head Attention (SA-MHA) mechanism that integrates structural features into the attention process, allowing the model to prioritize nodes based on their topological importance. We have conducted extensive experiments on both homophilic and heterophilic benchmark datasets to evaluate the effectiveness of our proposed method, SIGNNet. Our method consistently outperforms existing state-of-the-art approaches, demonstrating significant improvements across all datasets. Specifically, SIGNNet achieves average accuracy gains of 6.03%, 5.47%, and 4.78% on the homophilic datasets Cora, Citeseer, and CS, respectively. Even more substantial improvements are observed on the heterophilic datasets, with gains of 19.10% on Wisconsin, 19.61% on Texas, 19.54% on Cornell, 7.22% on Actor, and 14.94% on Chameleon. These results highlight the robustness and generalizability of SIGNNet in handling diverse graph structures.

List of Publications

1. **Aman Singh**, Shahid Shafi Dar, Ranveer Singh, and Nagendra Kumar (2025). "A hybrid similarity-aware graph neural network with a transformer for node classification." *Expert Systems with Applications.* https://doi.org/10.1016/j.eswa.2025.127292

Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

GCNs	Graph Convolutional Networks
GNNs	Graph Neural Networks
GATs	Graph Attention Networks
SIGNNet	A Hybrid Similarity-Aware Graph Neural Network with Transformer for Node Classification
SA-MHA	Structure-Aware Multi-Head Attention
PPR	Personalized PageRank
FFN	Feed-forward Network

Chapter 1

Introduction

1.1 Background

Graph structures provide powerful frameworks for modeling complex non-euclidean relationships between entities, making them invaluable across numerous real-world applications [1]. These graph relationships can be categorized as either homogeneous, where connections exist between nodes of identical types, or heterogeneous, where connections span across different node types. Graph-based representations have proven essential in diverse domains including social network analysis, recommendation systems, and biological network modeling [2]. Such representations facilitate enhanced relational data modeling while providing critical insights for community detection, node classification, and edge prediction tasks. The capacity to represent entities and their interactions in graph form is fundamental for capturing inherent relational structures across diverse domains.

Node classification the task of predicting labels for graph nodes by utilizing both node features and network topology presents distinct challenges in different graph structures. In homogeneous graphs, the primary challenge involves learning from uniform node and edge types, whereas heterogeneous graphs require models to capture complex multi-typed relationships and diverse node attributes. Graph Neural Networks (GNNs) have emerged as particularly effective models for addressing these challenges by leveraging both structural and feature information within graph data. GNNs excel at node classification by capturing local and global network patterns, making them well-suited for handling complex graph structures. Among various GNN architectures, Graph Convolutional Networks (GCNs) [3] have gained significant traction. GCNs function by propagating messages between neighboring nodes and aggregating features into unified representations. This localized feature aggregation enhances the expressiveness of node embeddings, enabling improved performance across node classification [4], community detection [5], recommendation systems [6], and graph classification [7] tasks. Despite their widespread success, GCNs perform optimally only on homophilic graphs where nodes share high feature similarity with their neighbors. They encounter significant challenges when applied to heterophilic graphs, where neighboring nodes exhibit substantial attribute differences. This limitation, known as over-squashing [8], occurs when the model struggles to propagate information across distant nodes, failing to capture the long-range dependencies crucial for effectively modeling complex heterogeneous graph relationships. The Graph Transformer architecture was introduced to address these GCN limitations, demonstrating superior performance in node classification tasks [9]. Graph Transformers capture information from the entire graph structure, enabling nodes to aggregate knowledge from all other nodes through an attention mechanism. This global attention capability proves advantageous for node classification by facilitating the capture of complex dependencies throughout the graph. However, the computational expense of calculating attention scores between each node and every other node in the graph presents a significant drawback, potentially capturing irrelevant information. This inefficiency can compromise node classification effectiveness, as not all nodes contribute meaningfully to target node representations. Our proposed framework addresses this limitation by selectively focusing on the most relevant nodes, thereby enhancing both classification performance and computational efficiency.

1.2 Motivation

Graph-structured data are used in various domains such as social networks, biological systems, recommendation engines, and citation networks. A central challenge in analyzing such data is the task of node classification, where the objective is to assign meaningful labels to graph nodes using both their attribute information and the relational information embedded in the graph. Traditional Graph Convolutional Networks have demonstrated remarkable success in this domain by leveraging neighborhood aggregation. However, they are fundamentally constrained by the over-squashing phenomenon, which hinders their ability to capture long-range dependencies in the graph. On the other hand, Graph Transformers offer a global view of the graph via attention mechanisms, but their high computational cost and lack of structural bias limit their scalability and effectiveness on large or complex graphs. The need for a unified, efficient, and structurally aware framework that overcomes the limitations of both GCNs and Graph Transformers provides the core motivation for this work. This thesis aims to address these challenges by proposing a hybrid architecture that combines the strengths of both paradigms while introducing novel components that improve node classification performance across a variety of homophilic and heterophilic graph datasets.

1.3 Challenges of Node Classification

Existing methods such as SAN [10] and SNGNN++ [4], offer state-of-the-art performance by addressing the common limitations of GCNs and transformers, ensuring accurate and contextually relevant outcomes. These methods use raw node features that contain only attribute information and lack local and global structural context. This limitation impairs the model's capacity to fully capture the underlying graph topology and the intricate relationships between nodes. Additionally, computing attention over the entire graph introduces scalability challenges due to the high computational cost of processing all node pairs, making these methods inefficient for large graphs. Another limitation is that these methods calculate attention scores based solely on node feature similarity, neglecting critical structural information. Attempts to capture the network using positional encodings, such as laplacian and spatial encodings, often fail to fully capture the graph structure. This restriction hampers the capture of crucial relationships between distant nodes, leading to less effective results.

1.4 Overview of Project

This thesis presents SIGNNet: a Hybrid Similarity-Aware Graph Neural Network with Transformer designed for scalable and accurate node classification. SIGNNet integrates structural and semantic information of the network by using three key strategies: (i) neighborhood-influenced feature learning, (ii) connection-aware feature representation, and (iii) class-centric feature representation.

To address scalability, the framework incorporates a Personalized PageRank-based node sampling technique that constructs informative subgraphs for each node, thereby reducing computational complexity without significant loss of structural information. Furthermore, the Structure-Aware Multi-Head Attention (SA-MHA) module is proposed to inject structural bias directly into the attention computation, allowing the model to focus on semantically and topologically relevant nodes.

Through extensive experimentation on eleven benchmark datasets spanning both homophilic and heterophilic graphs, SIGNNet demonstrates significant performance gains over existing state-of-the-art GNN and Transformer-based methods. The architecture effectively balances classification accuracy, scalability, and structural awareness, making it a robust solution for real-world node classification problems.

1.5 Problem Statement

Our research presents an innovative framework designed for node classification in graph-structured data. We aim to achieve precise node classification by systematically extracting, enriching, and utilizing intrinsic node features, effectively capturing the fundamental structural relationships and contextual information embedded within graph topologies. Mathematically, a graph is defined as $G = (V, E, X)$, where $V = \{v_1, v_2, \ldots, v_n\}$ represents a finite set of n nodes, E denotes the set of edges capture the relationships between nodes, and $X = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^{n \times d}$ is the node feature matrix, with d denoting the dimensionality of the feature space. The vector x_i is the feature vector of node v_i, which represents its characteristics.

The goal is to predict the node's label using a supervised learning framework. This framework will construct a function f that maps input features X to the corresponding node labels Y. The set Y includes all possible node labels, represented as $\{y_1, y_2, \ldots, y_m\}$. The framework is designed to improve feature representations, helping it to work well across different nodes and predict their labels correctly. The main goal is to create a mapping that is both accurate and flexible, ensuring better classification results for different graph structures.
This research aims to accomplish the following goals:

- Explore the use of Structure-Aware Multi-Head Attention mechanisms to directly incorporate topological information into the attention process, facilitating more effective integration of network structure during feature learning.

- Assess whether combining connection-aware features with class-centric representations enhances the model's capacity to embed complex higher-order network information within node representations.

- Evaluate the effectiveness of a strategic node-sampling approach based on Personalized PageRank for generating localized subgraphs, with the dual objectives of improving computational scalability and preserving critical structural characteristics.

- Determine if neighborhood-influenced feature aggregation techniques successfully capture and represent spatial dependencies among interconnected nodes within the graph.

Table 1.1: Overview of symbols

Notation	**Description**
$G = (V, E, X)$	Graph
n	Number of nodes
d	Dimension of node feature vector
$V = \{v_1, v_2, \ldots, v_n\}$	Set of nodes in the graph
$X = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^{n\times d}$	Node feature matrix
$P = [p_1, p_2, \ldots, p_n] \in \mathbb{R}^{n\times d}$	Convolutional feature amplification matrix
$X_{\text{final}} \in \mathbb{R}^{n\times d}$	Final enriched feature matrix
$X_{sim} \in \mathbb{R}^{n\times d}$	Similarity driven feature matrix
$X_{deg} \in \mathbb{R}^{n\times d}$	Connection-based convolutional feature matrix
$X_{comp} \in \mathbb{R}^{n\times n}$	Compatibility matrix
$Y = \{y_1, y_2, \ldots, y_m\}$	Set of node label/class
$A \in \mathbb{R}^{n\times n}$	Adjacency matrix
$N(i)$	Neighbour of node v_i
$C(i)$	Node similar to v_i
W, W_Q, W_V, W_K	Learnable weight matrices
A^T	Transpose of matrix A
$\bar{y}_k$	Representative of class k
$concat$	Concatenation

The Table 1.1 outlines the special symbols used in this thesis, providing clarity for the mathematical formulations discussed.

1.6 Contributions

The main contributions of SIGNNet are as follows:

[1] We introduce a novel framework that leverages both structural and semantic graph properties to enrich node features, facilitating a deeper understanding of node interactions and yielding enhanced model performance.

[2] We develop a dual approach combining class-centric analysis with connection score-based methods to effectively capture higher-order contextual information throughout the network, resulting in globally enriched node representations.

[3] We design a Structure-Aware Multi-Head Attention (SA-MHA) mechanism that

directly incorporates network topology into attention computations, thereby improving the model's ability to represent complex network dependencies.

[4] We formulate a specialized node-sampling technique utilizing the Personalized PageRank algorithm to generate node-specific localized subgraphs, enabling efficient training scalability while maintaining critical structural graph properties.

[5] We introduce neighborhood-influenced feature learning to aggregate spatial dependencies, facilitating more robust node feature representation. This approach improves the model's capacity for capturing localized structural patterns.

[6] We conduct comprehensive empirical evaluation across eleven diverse benchmark datasets, demonstrating that our approach consistently outperforms current state-of-the-art methods in node classification tasks.

1.7 Organizing the Thesis

This thesis is organized as follows:

- **Chapter 2** reviews existing literature in the field of node classification, GCN-based methods, and Graph Transformer-based methods.
- **Chapter 3** presents the proposed methodology, including model architecture, and algorithms.
- **Chapter 4** details the experimental setup, datasets used, evaluation metrics, performance results, and ablation studies.
- **Chapter 5** discusses the limitation, future work in this direction, and model complexity analysis.
- **Chapter 6** concludes the thesis by summarizing the key findings and contributions of the work.

Chapter 2

Literature Review

The exponential expansion of graph-structured data across diverse domains including social networks [11, 12] and biological systems [13, 14, 15] has catalyzed significant advancements in node classification techniques. This fundamental task involves predicting node labels by leveraging both node attributes and network connectivity patterns. As graphs continue to grow in complexity and scale, researchers have increasingly turned toward sophisticated deep-learning approaches. Graph Convolutional Networks (GCNs) [3] have demonstrated remarkable efficacy in capturing localized node dependencies, while more recent innovations [16, 17] have explored transformer-based architectures that employ attention mechanisms to enhance the interpretation of node relationships within extensive and intricate graph structures. For analytical clarity, we categorize the existing literature into two primary research directions: GCN-based methodologies and Graph Transformer-based approaches.

2.1 GCNs-based Methods

GCNs represent a sophisticated deep-learning architecture specifically engineered to transform complex graph structures into meaningful, low-dimensional representations that preserve essential topological and feature information. It systematically updates node representations through the aggregation of information from neighboring nodes and their associated features, as illustrated in Figure 2.1. The Graph Convolutional Network operates on two primary inputs: an adjacency matrix encoding the network's connectivity structure, and a feature matrix containing the intrinsic attribute information of individual nodes within the graph. The GCN processes these inputs to generate distinctive embeddings for each node in the network. These node embeddings encapsulate information from neighboring node features, making them particularly well-suited for node classification tasks [18]. The GCN operation can be conceptualized as a series of sequential layers, with each layer systematically refining node feature representations. Following the connectivity patterns defined in the adjacency matrix, nodes disseminate their feature information to adjacent neighbors within the network structure.

Subsequently, in each successive layer, the model aggregates these distributed features through a computational process typically involving linear transformation followed by non-linear activation functions. This iterative mechanism facilitates the accumulation of broader contextual information from throughout the network topology, a process that is repeated across multiple layers to capture increasingly complex structural patterns. Ultimately, the GCN generates refined node embeddings that provide semantically enriched representations, enabling more accurate and effective node classification.

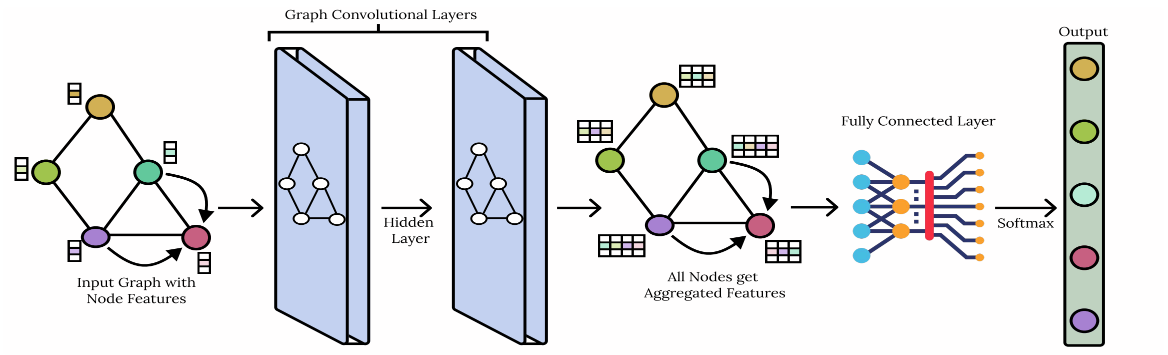

Graph Convolutional Layers
Input Graph with Node Features
Hidden Layer
All Nodes get Aggregated Features
Fully Connected Layer
Softmax
Output

Figure 2.1: Graph convolutional networks

GCNs have significantly advanced node classification tasks on graphs. The APPNP [19] employs a Personalized PageRank-based propagation scheme to enhance node classification by expanding the utilized neighborhood, thereby improving model efficiency. However, while APPNP leverages the Personalized PageRank (PPR) matrix for message passing, it can occasionally introduce redundant propagation, which may result in inefficiencies when capturing finer relationships within complex networks. GCNII [20] enhances traditional GCNs by integrating residual and identity mappings to tackle the over-smoothing problem, allowing for better preservation of feature information across multiple layers. While these improvements help maintain performance, GCNII can still encounter challenges in retaining initial node information, particularly in deeper layers and denser graphs, which may limit its ability to capture complex relationships effectively.

DisCo [21] is a graph-based disentangled contrastive learning framework for a cross-domain recommendation, capturing fine-grained user intents and filtering irrelevant source-domain information. It uses a multi-channel graph encoder and intent-wise contrastive learning guided by user similarity to enhance cross-domain knowledge transfer and mitigate negative transfer.

DisenSemi [22] is a semi-supervised framework for graph classification that learns disentangled graph representations using a graph encoder. It refines these representations through supervised objectives and mutual information-based constraints while enforcing consistency between supervised and unsupervised models via mutual information-based regularization. The Deep Graph Mutual Learning [23] framework is meant for Cross-domain Recommendation transfers knowledge using user-item interaction graphs. It builds domain-shared and domain-specific interaction graphs and uses parallel graph neural networks to capture user preferences.

HE-SNE [24] is a stream network embedding approach for modeling dynamic user behaviors by capturing their evolving nature over time. Unlike traditional static embedding methods, this approach represents user behaviors as temporal event sequences, updating node embeddings based on each event. Finally, GATE-GNN [25] employs a network ensemble to address imbalanced data, enhancing classification accuracy and stability in challenging environments. While this ensemble-based approach effectively handles class imbalance, the added complexity can reduce its efficiency on large-scale graphs, posing scalability and computational overhead challenges. Each method contributes distinctively to addressing the complexities of graph-structured data in supervised learning contexts. This research seeks to utilize deep learning approaches to create a sturdy and adaptable framework for node classification across diverse scenarios, addressing the complexities of graph-structured data. In the subsequent sections, we will discuss the Graph Transformer-based methods.

2.2 Graph Transformer-based Methods

The Graph Transformer represents a powerful and highly effective architecture for diverse graph-based tasks, including edge prediction [26], community detection [27, 28], and node classification, consistently demonstrating superior performance across these applications. At its architectural core, the Graph Transformer comprises two fundamental components: the Multi-Head Attention (MHA) mechanism and a position-wise feed-forward network. The sophisticated MHA mechanism processes input sequences of node features by projecting each node's feature representation into distinct query, key, and value spaces through learned weight matrices. This projection enables the model to compute attention by evaluating complex interactions between these transformed features, effectively capturing diverse relationship patterns among nodes within the graph structure. By simultaneously employing multiple attention heads, the MHA mechanism successfully captures a comprehensive spectrum of contextual information throughout the graph network [29, 30]. The outputs generated from these various attention heads undergo concatenation followed by a linear transformation, producing refined node feature embeddings. This architectural design equips Graph Transformers with the capacity to effectively learn intricate patterns and dependencies within complex graph structures, rendering them particularly powerful for node classification tasks.

Graph Transformer-based methods for node classification have introduced innovative ways to enhance node representation and scalability. Gapformer [31] integrates Graph Transformer with graph pooling, significantly reducing computational complexity while maintaining long-range interactions. However, while the pooling mechanism enhances efficiency, it may occasionally overlook critical local node information, which can impact performance in fine-grained node classification tasks. Dual-Encoding Transformer [32] utilizes both structural and semantic encoders to enhance representation learning by capturing comprehensive node interactions. However, the integration of dual encoders can introduce synchronization challenges, especially in highly connected or large-scale graphs where balancing structural and semantic information becomes complex. Spectral Attention Network (SAN) [10] leverages Laplacian-based positional encoding to mitigate over-squashing and enhance sub-structure detection, offering strong global context modeling. HEAL [33] is a deep learning model for protein function prediction. It utilizes a hierarchical graph transformer with super-nodes to capture structural semantics and employs graph contrastive learning to enhance graph representation by maximizing similarity between different views. GraphSite [34] is a DNA-binding residue predictor by leveraging protein structures predicted by AlphaFold2. It frames the binding site prediction as a graph node classification task and employs a transformer-based model to incorporate structural information. However, while utilizing the full Laplacian spectrum improves global representation, this reliance may limit its ability to capture localized structural details, particularly in graphs with distinct local variations.

Chapter 3

Proposed Method

3.1 Overview

This section focuses on the details of SIGNNet, as shown in Figure 3.1. The framework has three main parts: (a) feature augmentation and enrichment; (b) adaptive node sampling integrated with sequence module; and (c) attention-based node classification. In the feature augmentation and enrichment module, node features are enhanced by integrating both local and global network information. Local information of the network is incorporated into the node's feature by using neighborhood-influenced feature learning. Global information is captured in two ways: first, through connection score, which measures a node's importance in the network by counting its direct neighbors; and second, by concatenating the most similar class-centric feature to the node's feature. In the adaptive node sampling integrated with the sequence module, we construct node sequences by sampling relevant neighbors to capture the node's structural property and neighborhood information. The final enriched node features and node sequences are passed to the attention-based node classification module, including the Graph Transformer and a dense layer for final prediction.

3.1.1 Feature Augmentation and Enrichment

Traditional node classification methods [35, 36, 37] rely on raw node features, which can overlook complex relationships in graph structures. In the proposed method, node features are enhanced by integrating both local and global network information. Local information is aggregated by leveraging graph convolutional networks while global information is aggregated by connection score and class-centric mechanism, all the steps are defined in Algorithm 3.1. This makes the feature of each node more comprehensive, resulting in better classification performance. In the next section, we will discuss how local information is incorporated into node features by the message-passing technique.

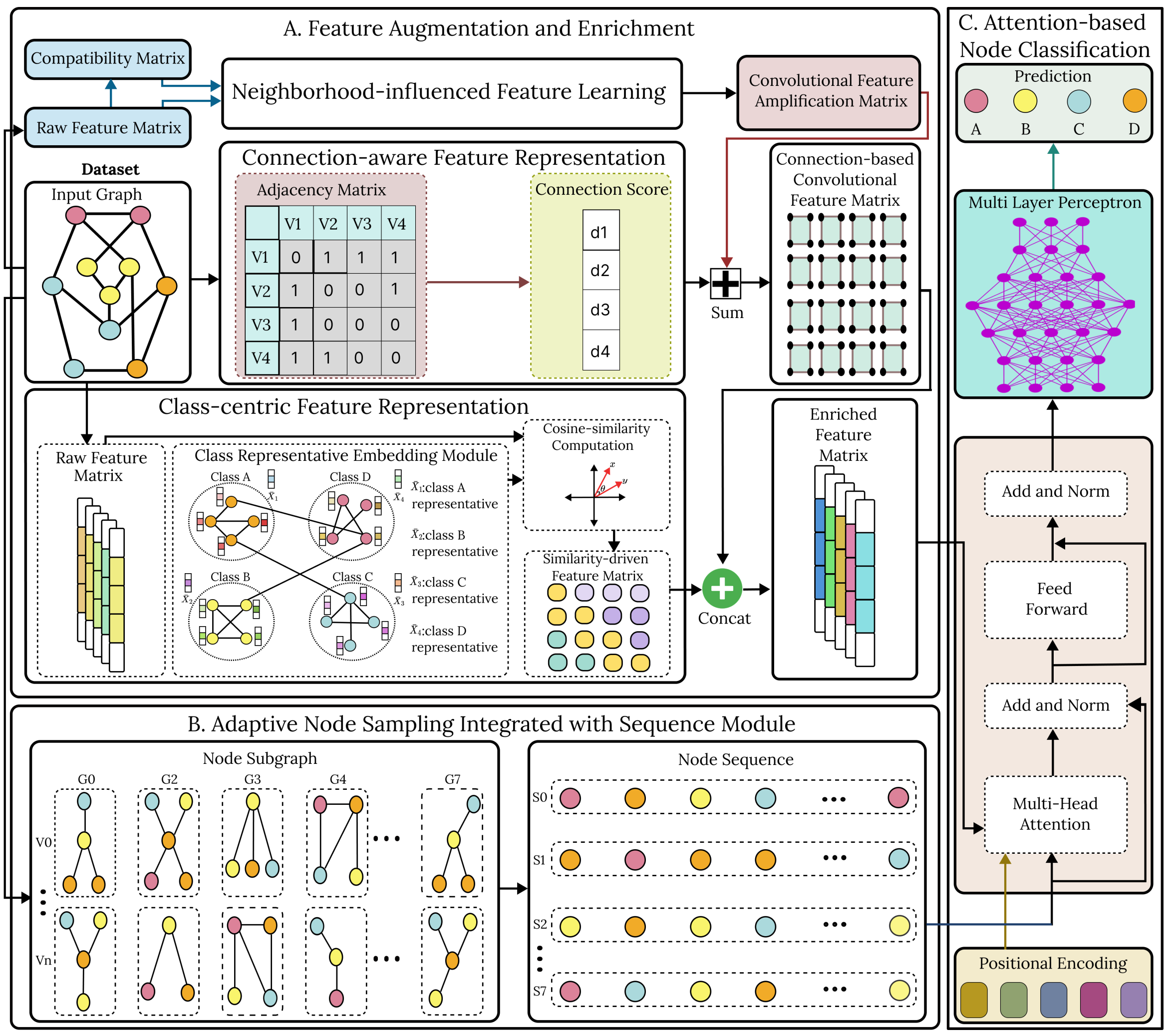

A. Feature Augmentation and Enrichment
Compatibility Matrix
Raw Feature Matrix
Neighborhood-influenced Feature Learning
Convolutional Feature Amplification Matrix
Dataset
Input Graph
Connection-aware Feature Representation
Adjacency Matrix
V1 V2 V3 V4
V1 0 1 1 1
V2 1 0 0 1
V3 1 0 0 0
V4 1 1 0 0
Connection Score
d1
d2
d3
d4
Sum
Connection-based Convolutional Feature Matrix
Class-centric Feature Representation
Raw Feature Matrix
Class Representative Embedding Module
Class A
Class D
Class B
Class C
$\bar{X}_1$:class A representative
$\bar{X}_2$:class B representative
$\bar{X}_3$:class C representative
$\bar{X}_4$:class D representative
Cosine-similarity Computation
Similarity-driven Feature Matrix
Concat
Enriched Feature Matrix
B. Adaptive Node Sampling Integrated with Sequence Module
Node Subgraph
G0
G2
G3
G4
G7
V0
Vn
Node Sequence
S0
S1
S2
S7
C. Attention-based Node Classification
Prediction
A
B
C
D
Multi Layer Perceptron
Add and Norm
Feed Forward
Add and Norm
Multi-Head Attention
Positional Encoding

Figure 3.1: Proposed architecture

3.1.1.1 Neighborhood-influenced Feature Learning

Graph Convolutional Networks (GCNs) is a neural network that works on graphs, capturing the relationship between connected nodes. Each node updates its features by gathering information from its neighbors. The input to the GCNs is feature matrix $X \in \mathbb{R}^{n \times d}$, where n is the number of nodes and d is the dimension of the feature vectors and a compatibility matrix $X_{\text{comp}} \in \mathbb{R}^{n \times n}$ encapsulates pairwise feature similarity, marking entries as 1 when similarity exceeds threshold and 0 otherwise. This approach enhances the representation of nodes by covering long-range dependencies and enriching node features with broader contextual information from the network. There are three major steps in the GCN: aggregation, combination, and updating as shown in Figure 3.2. In the aggregation step, each node v_i aggregates information from its similar nodes $C(i)$ and itself. This information is weighted based on the degree of the node v_i and v_j denoted by d_i and d_j respectively, adjusting how much each node contributes. In the combination step, the aggregated information is processed through a learnable weight matrix $W^{(l)}$, which refines the node features into more meaningful representations. Finally, in the updating step, a non-linear function is applied to update the features for the next layer as shown in Equation 3.1:

$$h_i^{(l+1)} = \sigma \left(\sum_{v_j \in C(i) \cup \{i\}} \frac{1}{\sqrt{d_i \cdot d_j}} h_j^{(l)} W^{(l)} \right) \tag{3.1}$$

where $h_j^{(l)}$ represents the features of node v_j at layer l and σ denotes sigmoid activation function. In summary, GCN is applied to the graph structure for the feature enrichment of each node. This leads to a convolutional feature amplification matrix $P = [p_1, p_2, \ldots, p_n] \in \mathbb{R}^{n \times d}$ that effectively captures semantic information. In the following section, we will explore the concept of Connection-aware feature representation.

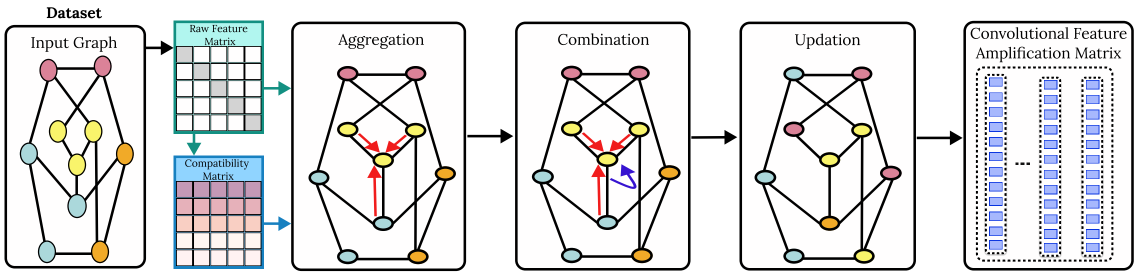


Figure 3.2: Neighborhood-influenced feature learning

3.1.1.2 Connection-aware Feature Representation

Features are further improved by connection score, a basic measure of a node's importance within the graph. Connection score $deg(v_i)$ evaluates the connectivity of a node by counting the number of adjacent nodes in the network. This metric is very helpful for determining how influential a node is in the network. Mathematically, the connection score for a node v_i is defined in Equation 3.2.

$$deg(v_i) = \sum_{j=1}^{n} A_{ij} \tag{3.2}$$

The adjacency matrix is a square binary matrix of size $A \in \mathbb{R}^{n \times n}$, where n is the total number of nodes in the graph. Each element A_{ij} in the matrix is equal to 1 if there is a direct edge between node v_i and node v_j otherwise, it is 0. This means that $deg(v_i)$ is calculated by summing up all the values in the row corresponding to node v_i, effectively counting the total number of edges connected to v_i. Including the connection score enriches the node's feature vector by adding information about how well-connected each node is in the graph. This enhancement provides the model with a clearer view of the node's role within the network. Once the connection score is computed for each node, it is integrated into the node's feature vector to form a connection-based convolutional feature matrix $X_{deg} \in \mathbb{R}^{n \times d}$. This integration is performed by adding the connection score $deg(v_i)$ directly to the feature vector $p(v_i)$ of each node v_i obtained by the message passing. The resulting feature vector $x_{deg}(v_i)$ is thus computed as shown in Equation 3.3.

$$x_{deg}(v_i) = p(v_i) + deg(v_i) \tag{3.3}$$

After adding the connection score to each node's feature, the updated node feature now reflects its importance within the network [38], which will contribute to accurate predictions. In the next section, we discuss about class-centric feature representation.

Algorithm 3.1 Node feature enhancement for graph transformer

Input: Adjacency matrix $A \in \mathbb{R}^{n \times n}$, Compatibility matrix $X_{comp} \in \mathbb{R}^{n \times n}$, Feature matrix $X \in \mathbb{R}^{n \times d}$, class labels $\{y_1, y_2, \ldots, y_m\}$

Output: Final feature matrix $X_{\text{final}} \in \mathbb{R}^{n \times d}$

```
1: for each node $v_i \in V$ do
2:     $h_i^{(l+1)} = \sigma(\sum_{v_j \in C(i) \cup \{i\}} \frac{1}{\sqrt{d_i \cdot d_j}} h_j^{(l)} W^{(l)})$
3: end for
4: $P = [p_1, p_2, \ldots, p_n] \in \mathbb{R}^{n \times d}$
5: for each node $v_i \in V$ do
6:     $\deg(v_i) = \sum_{v_j=1}^{n} A_{ij}$
7:     $X_{\deg}(v_i) = P(v_i) + \deg(v_i)$
8: end for
9: for $y_k \in \{y_1, y_2, \ldots, y_m\}$ do
10:     $\bar{x}_k = \frac{1}{|y_k|} \sum_{v_i \in y_k} x_i$
11: end for
12: for $v_i \in \{v_1, v_2, \ldots, v_n\}$ do
13:     for $y_k \in \{y_1, y_2, \ldots, y_m\}$ do
14:         $sim(x_i, \bar{x}_k) = \frac{x_i \cdot \bar{x}_k}{\|x_i\| \|\bar{x}_k\|}$
15:     end for
16:     $k = \arg\max_k sim(x_i, \bar{x}_k)$
17:     $X_{sim}(v_i) = [x(v_i), \bar{x}_k]$
18: end for
19: $X_{\text{final}} = [X_{\text{deg}}, X_{\text{sim}}]$
```

3.1.1.3 Class-centric Feature Representation

We compute class-centric features that encapsulate the features of all nodes belonging to the same class. This leads to including the global context of graph structure in the node feature. For each class y_k, we calculate a class representative $\bar{x}_k$, which is the average of the feature vectors of all nodes v_i that belong to class y_k. This computation is formalized in Equation 3.4:

$$\bar{x}_k = \frac{1}{|y_k|} \sum_{v_i \in y_k} x_i \tag{3.4}$$

where $|y_k|$ denotes the number of nodes in class k, and $x_i \in \mathbb{R}^d$ represents the feature vector of node v_i. The resulting class representative $\bar{x}_k$ serves as a centroid in the feature space, capturing the central tendency of the features within that class. For each node v_i, we calculate the cosine similarity $sim(x_i, \bar{x}_k)$ between its feature vector x_i and each class representative $\bar{x}_k$. This similarity measure, defined in Equation 3.5, shows how closely the node's features align with the characteristics of each class.

$$sim(x_i, \bar{x}_k) = \frac{x_i \cdot \bar{x}_k}{\|x_i\| \|\bar{x}_k\|} \tag{3.5}$$

After computing the similarities, the class representative that exhibits the highest similarity with the node v_i is identified. This most similar class representative is then concatenated with the original feature vector of the node to produce the similarity-enriched feature matrix $X_{sim} \in \mathbb{R}^{n \times 2d}$, as described in Equation 3.6.

$$x_{sim}(v_i) = [X(v_i) \parallel \bar{x}_k \text{ for } k = \arg\max_k sim(x_i, \bar{x}_k)] \tag{3.6}$$

This approach allows each node to inherit global information about its class, improving its feature representation by integrating not only its own attributes but also the typical characteristics of the class it is most similar to. This similarity-enriched feature matrix enhances the ability of the model to generate more correct predictions by considering both local node features and broader class-level patterns.

The final feature matrix $X_{\text{final}} \in \mathbb{R}^{n \times 3d}$ is initially formed by concatenating two distinct matrices: the connection-based convolutional feature matrix $X_{\text{deg}} \in \mathbb{R}^{n \times d}$ and the similarity-enriched feature matrix $X_{\text{sim}} \in \mathbb{R}^{n \times 2d}$. This process expands the dimensionality of the final feature matrix to encompass both local and global information, offering a robust representation that adeptly handles the complexities of graph structures. Following this expansion, X_{final} is processed through a fully connected layer that compresses its dimensionality down to d, resulting in a streamlined final features matrix $X_{\text{final}} \in \mathbb{R}^{n\times d}$. This compression step prepares the enriched features for classification tasks. These feature vectors are subsequently utilized as inputs in the classification model, enhancing the classification performance. In the next section, we discuss the node sampling technique.

The Algorithm 3.1 enhances node features in a graph by performing several operations sequentially. Initially, from lines 1 to 4, it aggregates weighted features from each similar node based on the Compatibility matrix, applies a non-linear transformation, and creates a convolutional feature amplification matrix $P \in \mathbb{R}^{n\times d}$. Subsequently, lines 5 to 8 compute each node's connection score and enhance its features by incorporating this score into the convolutional feature amplification matrix P, resulting in a connection-based convolutional feature matrix $X_{\text{deg}} \in \mathbb{R}^{n\times d}$. Lines 9 to 11 focus on deriving class-centric features for each class from the nodes within those classes. The final steps, lines 12 to 18, involve computing cosine similarities between each node's features and these class-centric features, concatenating with the most similar class-centric feature to form a similarity-driven feature matrix $X_{\text{sim}} \in \mathbb{R}^{n\times d}$, and combining $X_{\text{sim}} \in \mathbb{R}^{n\times d}$ and $X_{\text{deg}} \in \mathbb{R}^{n\times d}$ to produce the final feature matrix $X_{\text{final}} \in \mathbb{R}^{n\times d}$. This process effectively utilizes both local connectivity and global class information to refine node representations for improved performance in node classification.

Algorithm 3.2 Sampling matrix creation

```
Input: Adjacency matrix $A \in \mathbb{R}^{n\times n}$, c is damping factor and, $I \in \mathbb{R}^{n\times n}$ is identity matrix
Output: Sampling matrix $S \in \mathbb{R}^{n\times n}$
1: Initialize a matrix $D \in \mathbb{R}^{n\times n}$ with all zeros.
2: $d = A \times \mathbf{1}$
3: for $v_i \in \{v_1, v_2, \ldots, v_n\}$ do
4:     $D[v_i][v_i] = d[v_i]$
5: end for
6: $\hat{A} = A \times D^{-1}$
7: $S = c(I - (1-c)\hat{A})^{-1}$
```

3.1.2 Node Sampling and Sequence Construction

In traditional methods [10, 38], the Graph Transformer typically processes the entire graph, considering all nodes and edges together, which may raise scalability concerns. To address this our framework uses a node sampling technique based on the Personalized PageRank (PPR) algorithm [39] to create subgraphs around each node defined in Algorithm 3.3. Using the sampling matrix $S \in \mathbb{R}^{n \times n}$ where n is the number of nodes in the graph defined in Algorithm 3.2, we identify the most influential nodes relative to a given node v. The vector r is defined for each node as shown in Equation 3.7:

$$r = c(I - (1-c)\hat{A})^{-1} e_v \tag{3.7}$$

where $\hat{A}$ is the transition matrix calculated as,

$$\hat{A} = A \times D^{-1} \tag{3.8}$$

where $I \in \mathbb{R}^{n \times n}$ is the identity matrix, $D \in \mathbb{R}^{n \times n}$ is the degree matrix, $A \in \mathbb{R}^{n \times n}$ is the adjacency matrix, e_v is a one-hot vector representing the node v, and c is damping factor. The node sampling process begins by calculating scores from the sampling matrix $S \in \mathbb{R}^{n \times n}$, where each row reflects the relative influence of all nodes with respect to node v. To prevent the node from sampling itself, the diagonal entry is set to a large negative value. Next, the top k_1 neighbors of the node v are identified based on their PPR scores, selecting those with the highest influence within the local neighborhood.

Following this, probabilistic sampling is performed, selecting a subset of nodes with positive PPR scores, along with the top k_1 neighbors, to form the subgraph. The indices of these sampled nodes, together with node v itself, are then used to construct the subgraph, ensuring that each subgraph contains $k_1 + 1$ nodes. This approach effectively captures both highly connected and structurally diverse neighborhoods, allowing the model to focus on locally significant nodes while maintaining the overall graph structure. In the following module, we will explore the third component of our framework, which focuses on attention-based node classification.

In Algorithm 3.2, lines $1 - 6$ efficiently compute the transition matrix by Equation 3.8. This normalization balances node influence across the graph. This forms $\hat{A}$, a symmetric matrix that scales adjacency relations based on normalized degrees, enhancing graph structure representation. Lines 7 compute the sampling matrix S by incorporating the transition matrix and a damping factor c. It will be further used in finding the subgraph of each node in the graph explained in Algorithm 3.3.

- **selectNode**$(v_i, sample\_index, p)$**:** It takes input as a node, the number of nodes to be selected, and probability vector p. It randomly selects r number of nodes with weighted probability without repetition.
- **select**$(top\_neighbor\_index, k_1 - r)$**:** It takes input as the $top\_neighbor\_index$ and the number of nodes to be selected. It returns randomly selected $k_1 - r$ nodes from the $top\_neighbor\_index$.

In Algorithm 3.3, the initialization of $data\_list$ in line 2 sets up storage for subgraphs of each node. For each node v_i, its entry in the sampling matrix S is set to $-\infty$ in line 4 to prevent self-inclusion in subgraphs. The top k_1 influential neighbors are identified in line 5, and the entry for v_i is reset to zero in line 6. The positive entries in $S[v_i]$ are counted between lines $8 - 12$, followed through the calculation of a probability vector p for sampling in line 15. For each node, q subgraphs are generated by sampling nodes using p in line 18, with additional nodes selected from top neighbors to complete the subgraph size in line 22. The resulting subgraphs are stored in $data\_list$ in line 25. This approach efficiently combines matrix operations and stochastic sampling to generate localized subgraphs for each node.

3.1.3 Attention-based Node Classification

The drawbacks of Graph Transformer are discussed in this section along with solutions for better classification. Graph transformers use only the node features to calculate the attention score which ignores the structural property of the network. To address this, we introduce a structural encoding vector to incorporate structural details into the attention process.

Algorithm 3.3 Subgraph sampling for ensemble

```
Input: Sampling matrix $S$, nodes $v_1, v_2, \ldots, v_n$, number of nodes in each subgraph $k_1$, number of samples $q$
Output: Subgraphs of each node
1: $data\_list \leftarrow []$
2: for $v_i \in \{v_1, v_2, \ldots, v_n\}$ do
3:     $s = S[v_i]$
4:     $s[v_i] = -\infty$
5:     $top\_neighbor\_index = s.argsort()[-k_1 :]$
6:     $s = S[v_i]$
7:     $s[v_i] = 0$
8:     $count = 0$
9:     for $v_j \in \{v_1, v_2, \ldots, v_n\}$ do
10:        $s[v_j] = \max(s[v_j], 0)$
11:        if $s[v_j] > 0$ then
12:            $count = count + 1$
13:        end if
14:    end for
15:    $r = \min(count, k_1)$
16:    $sub\_data\_list \leftarrow []$
17:    $p = s/s.sum$
18:    for $m \in \{1, 2, \ldots, q\}$ do
19:        if $r > 0$ then
20:            $node\_sample\_id = \text{selectNode}(v_i, r, p)$
21:        else
22:            $node\_sample\_id = []$
23:        end if
24:        $remaining\_nodes = \text{select}(top\_neighbor\_index, k_1 - r)$
25:        $node\_sample\_id = \text{concatenate}(v_i, node\_sample\_id, remaining\_nodes)$
26:        $sub\_data\_list.\text{append}(node\_sample\_id)$
27:    end for
28:    $data\_list.\text{append}(sub\_data\_list)$
29: end for
30: Return $data\_list$
```

3.1.3.1 Structure-Aware Multi-Head Attention

Structure-Aware Multi-Head Attention (SA-MHA) is introduced to address above mentioned limitations. For a pair of nodes v_i and v_j, multiple views of structural information are captured in a structural encoding vector, denoted as ψ, which enhances the attention mechanism. The attention score β is computed as defined by Equation 3.9.

$$\beta = \frac{QK^T}{\sqrt{d}} + \psi \tag{3.9}$$

Here, d represents the dimension of the key vectors, while Q and K denote the learnable weight matrices for the query and key, respectively. The structural encoding ψ_{ij} is generated by concatenating multiple structural encoding functions represented by Equation 3.10.

$$\psi_{ij} = Concat(\alpha_m(v_i, v_j) \mid m \in \{0, 1, \ldots, M-1\}) \tag{3.10}$$

Here, $Concat$ is a concatenation operation. Each structural encoding function α_m captures a specific aspect of the relationship between the node pair defined in Equation 3.11. In our approach, we consider the structural order in which node pairs are connected. The structural encoding functions are defined in Equation 3.11:

$$\alpha_m(v_i, v_j) = \begin{cases} \tilde{A}_m[i, j], & \text{if } m < M-1 \\ 0, & \text{otherwise} \end{cases} \tag{3.11}$$

where $\tilde{A} = \text{Norm}(A + I)$ is the normalized adjacency matrix including self-connections. Thus, the first $M-1$ dimensions ofϕ_{ij} encode the probabilities of reachability from 0-order (identity relationship) to $(M-2)$-order between nodes v_i and v_j, preserving fine-grained structural information for each node pair.

3.1.3.2 Prediction

The prediction process in our framework leverages the attention mechanism to incorporate both the graph's structural information and the positional relationships between nodes. This process begins with encoding the input features $X_{\text{final}} \in \mathbb{R}^{n \times d}$ and applying a linear transformation to elevate them into a space with higher dimensionality defined in Equation 3.12:

$$H^{(0)} = Linear(X_{\text{final}}) \tag{3.12}$$

The initial encoded representation is refined using the SA-MHA, allowing the model to simultaneously capture different parts of the graph while incorporating structural biases. The query, key, and value matrices for the attention heads are computed as $Q^{(l)} = H^{(l-1)}W_Q^{(l)}, K^{(l)} = H^{(l-1)}W_K^{(l)}, V^{(l)} = H^{(l-1)}W_V^{(l)}$. The attention score for each head is computed using the scaled dot-product defined by Equation 3.13, enhanced by structure-aware encoding.

$$Attention(Q, K, V) = softmax\left(\frac{QK^\top}{\sqrt{d_k}} + \psi\right) V \tag{3.13}$$

where ψ represents the attention bias, capturing both structural and positional relationships between the nodes. After the attention scores are computed, the output passes through a feed-forward network (FFN), which applies two linear transformations separated by a GELU activation function as expressed by Equation 3.14.

$$FFN(H^{(l)}) = GELU\left(H^{(l)}W_1 + b_1\right) W_2 + b_2 \tag{3.14}$$

Residual connections and layer normalization are then applied to stabilize the training process as illustrated by Equation 3.15.

$$H^{(l+1)} = LayerNorm(H^{(l)} + FFN(H^{(l)})) \tag{3.15}$$

Finally, the hidden representation $H^{(l)}$ is used for prediction by applying a linear transformation followed by a softmax function to produce class probabilities as expressed by Equation 3.16.

$$T = softmax\left(H^{(L)}W_{out} + b_{out}\right) \tag{3.16}$$

The predicted class for each node is determined by choosing the class with the greatest probability in the output distribution $T \in \mathbb{R}^{1\times u}$ where u is a number of classes in the dataset. This method ensures that the final predictions fully use the structural and positional information embedded in the graph, allowing for accurate node classification within complex network systems.

Chapter 4

Experimentation and Result

This section presents the datasets used for the experiments and the comparison of our proposed method with existing Graph Transformer-based and GNN-based approaches. Then we discuss the hyperparameter information and perform the ablation studies.

4.1 Datasets

To validate our method's adaptability we experiment on both homophilic and heterophilic datasets summarized in Table 4.1. Homophilic datasets include Cora, Citeseer, and Pubmed, where nodes are likely to link with similar others. Heterophilic datasets such as Wisconsin, Texas, Cornell, Chameleon, Actor, CS, Photo, and Squirrel in which nodes belonging to different classes are more prone to being connected. To understand the structural properties of the datasets we compute the following graph metrics.

- Average Degree (AD) [40]: To understand the graph structure, we calculate the average degree of nodes in each dataset. A high degree generally means a dense graph, which can then influence the learning efficiency of graph-based deep learning models.

- Clustering Coefficient (CC) [41]: This metric helps to measure the tendency of nodes to cluster together, such that get a tightly connected cluster of nodes. In general, a graph with a very high clustering coefficient has a high probability that nodes in the community will be connected.

- PageRank Centrality (PRC) [42]: We calculate the PageRank centrality to the order of incoming links to calculate the importance of each node. In particular, it is useful in node classification tasks since it can be used to identify influential nodes that should be of vital importance when information flows through the network directly affecting the classification results.

- Homophily Ratio (HR) [43]: This metric helps to measure the feature similarity of labeled nodes that are connected in a network. High symmetry ratios mean that nodes are connected with nodes in the same class, which can greatly influence the efficiency of node classification algorithms.

- Triangles (Tri) [40]: To understand how different parts of the network are connected, the probability of forming dense clusters and to better understand local connectivity, we count of triangles for each node.

4.1.1 Cora, Citeseer, and Pubmed

The Cora, Citeseer, and Pubmed [44] datasets are traditional citation networks that are widely adopted in graph-based tasks. In these networks, the node represents the research paper while edges connect nodes in cases where one paper cites the other. The node features are sparse bag-of-words vectors obtained from the paper textual content, and each feature indicates the presence of specific text in the paper.

4.1.2 CS

The Coauthor CS [45] dataset models a graph where nodes represent individual authors connected by edges if they have co-authored a paper. This forms a network reflecting real-world academic collaborations. Node features often include embeddings from their publications, such as bag-of-words from paper keywords, offering semantic insights into their research domains.

4.1.3 Photo

The Amazon Photo [45] dataset forms a graph where each node represents a photography-related product, linked by edges when products are frequently purchased together, mirroring consumer buying behaviors on e-commerce sites. Node features include bag-of-words extracted from product reviews, offering valuable insights into consumer preferences and the attributes of the products.

Table 4.1: Summary of homophilic and heterophilic datasets

Dataset	# Nodes	# Edges	# Classes	# Features	Tri	AD	CC	PRC	HR
Homophilic Datasets									
Cora	2,708	5,429	7	1,433	1.81	3.90	0.240	0.000369	0.810
Citeseer	3,327	4,732	6	3,703	1.05	2.74	0.140	0.000301	0.736
PubMed	19,717	44,327	3	500	1.90	4.50	0.060	0.000051	0.802
CS	18,333	163,788	15	6,805	14.04	8.93	0.343	0.000055	0.808
Photo	7,650	238,162	8	745	281.33	31.13	0.404	0.000131	0.827
Heterophilic Datasets									
Wisconsin	251	499	5	1,703	1.41	2.05	0.208	0.003984	0.196
Texas	183	295	5	1,703	1.10	1.78	0.198	0.005464	0.108
Cornell	183	280	5	1,703	0.97	1.65	0.167	0.005464	0.131
Actor	7,600	26,752	5	931	2.81	3.95	0.080	0.000132	0.219
Squirrel	5,201	198,493	5	2,089	5534.86	41.74	0.422	0.000192	0.224
Chameleon	2,277	31,421	5	2,325	451.99	15.85	0.481	0.000439	0.235

Note: # Nodes = Number of nodes, # Edges = Number of edges, # Classes = Number of classes, # Features = Number of features per node, Tri = Average triangle count, AD = Average degree, CC = Clustering coefficient, PRC = PageRank centrality, HR = Homophily ratio.

4.1.4 Texas and Wisconsin

The Texas and Wisconsin [46] datasets are smaller-scale web networks, frequently utilized to test model performance on challenging graph structures. In these datasets, nodes represent web pages, and edges correspond to hyperlinks between them. The node features are binary vectors indicating the presence or absence of specific keywords on the web pages.

4.1.5 Actor

The Actor [47] dataset, derived from the film industry, features a unique graph structure where nodes are actors and there are edges between nodes if they have collaborated professionally. The characteristics of the actors including gender, birth year, and the genres of movies they have appeared in, provide a wide range of datasets for model evaluation.

4.1.6 Cornell

The Cornell [48] dataset is a popular website network in which nodes are web pages and if there is a hyperlink from one page to another then there is an edge between them. Node features are the bag-of-words representation of web pages. Web pages can be classified into the five classes student, project, course, staff, and faculty.

4.1.7 Squirrel and Chameleon

The Squirrel and Chameleon [49] dataset are popular Wikipedia page-page network. In which nodes are the articles from Wikipedia and if there are hyperlinks between the articles then there is an edge between them. Node features indicate the presence of particular nouns in the articles. The nodes were classified into 5 classes in terms of their average monthly traffic.

4.2 Comparison Methods

We compare SIGNNet with the existing methods to show its effectiveness. In this section, we summarize the various methods used for comparison in our experiment. We start with GNN-based methods and conclude with the graph transformer-based methods.

4.2.1 GNN based Methods

We compare the performance of SIGNNet with the following GNN-based approaches.

[1] GCN [3] applies a scalable approach to graphs by using a simplified version of convolutional neural networks directly on graph data. It learns node representations that capture both local structure and node features.

[2] GAT [50] introduces graph attention networks, which use self-attention layers to allow nodes to focus on their most important neighbors.

[3] APPNP [19] is based on GCN and PageRank to improve node classification in graphs. It is called Personalized propagation of neural predictions (PPNP), uses a propagation scheme based on Personalized PageRank and its faster version is APPNP.

[4] GCNII [20] is an improved version of GCN that solves the over-smoothing problem. Two techniques are used such as initial residual connections and identity mapping. It was performing better than the existing methods present at that time.

[5] ACM-GCN [51] addresses the issue of heterophily in Graph Neural Networks (GNNs), where connected nodes significantly differ, often leading to underperformance. The Adaptive Channel Mixing (ACM) framework, leverages dynamic utilization of aggregation, diversification, and identity channels in each GNN layer.

[6] GATv2 [52] is an enhanced version of the original Graph Attention Network (GAT). To overcome the drawbacks of GAT it uses dynamic attention in place of static attention, which allows it to handle more complex graphs and perform better over a range of graph structures.

[7] Non-local GNNs [53] introduce a new framework for graph neural networks (GNNs) that utilizes non-local aggregation, which is essential for tasks on graphs

with low homophilic coefficients. The author suggests an effective attention-based technique for non-local aggregation and demonstrates how local aggregation can be harmful for certain kinds of graphs.

[8] FSGNN [37] is simple and improves performance by selecting only the most useful features from graph data. L2-Normalization and SoftMax are used to eliminate less informative features that were collected from neighbors.

[9] SNGNN++ [4] is a method created to overcome the difficulties in heterophilic graphs, in which nodes of the same class do not always need to be connected. To address this, a matrix is calculated to show the similarity between the nodes, helping with better aggregation and improving model performance.

[10] UniG-Encoder [54] uses a projection matrix to transform node connections into edge features, which are processed by a neural network. A reverse transformation is used to derive the node embeddings, and they are used for graph-based tasks. In contrast to existing methods, UniG-Encoder seamlessly couples node features and graph structures while performing well with both heterophilic and homophilic graphs.

[11] Geom-GNN [46] finds that message-passing neural networks generally do not adequately capture structural information and detect distant relationships in graphs. To deal with it, the geometric aggregation scheme employs the ideas from the network geometry to achieve worthwhile exploitation of the continuous space below the graph structure. The whole approach is based on node embedding, mapping of structural neighborhoods, and dual-level aggregation. These components are incorporated into the GeomGCN model to significantly improve direct graph learning.

4.2.2 Graph Transformer based Methods

We compare the performance of SIGNNet with the following Graph Transformer-based approaches.

[1] SAN [10] uses learned positional encoding (LPE) to capture the full laplacian spectrum, which supports the model to understand the position of the nodes in the graph. The LPE is added to the node features, and then the fully connected transformer network processes the aggregate features. This approach offers deeper insights into the graph structure and enhances model performance.

[2] UniMP [55] is a method that combines feature and label propagation for better classification from the graph convolutional network and the label propagation algorithm. It uses a graph transformer to process feature and label embeddings, and to reduce overfitting uses a masked label prediction strategy.

[3] DET [32] is designed to improve the scalability in large graphs. It uses two encoders first is the structural encoder, which aggregates the information from neighbors and second is the semantic encoder, which finds semantically related nodes. This method performs better for node classification tasks.

[4] Adaptive Graph Transformer AGT [40] is introduced to enhance node classification in graphs. To overcome the limitations of existing graph transformers, it incorporates trainable centrality encoding and kernelized local structure encoding to capture structural properties effectively. Additionally, it includes an adaptive transformer block, boosting performance on node classification tasks.

[5] NAGphormer [56] is designed to handle large graphs efficiently. It uses a Hop2Token module that turns neighborhood features from different hops into multiple tokens for each node, instead of treating each node as a single token. It also learns more informative node representations compared to advanced Graph Neural Networks (GNNs).

[6] Gapformer [31] is designed to improve node classification using Graph Transformers. It focuses on two issues: the first is irrelevant information from distant nodes, and the second is high computational cost. This method overcomes these issues by using graph pooling, which reduces the number of nodes for attention while keeping long-range information, thus reducing complexity.

[7] SoftGNN [57] Graph neural networks excel in networks where similar nodes connect but struggle in heterophilic networks with dissimilar connections. Introduces a label-guided GNN that leverages node labels to selectively aggregate neighborhood information across different classes. By incorporating an adaptive attention mechanism, this model enhances node representation distinguishability and generalizes better across varying network types.

4.3 Experimental Results

In this section, we present a comprehensive overview of the experimental setup, describing the datasets, model configurations, hyperparameter settings, and evaluation metrics. Following this, we discuss the performance comparison on both homophilic and heterophilic datasets.

4.3.1 Experimental Setup

To evaluate the performance of the SIGNNet, we run this model using the Adam optimizer for the GCN with a learning rate of 0.01 and a weight decay of 5e-4. The hidden dimensions are set to 128, and dropout of 0.3 to avoid overfitting. We used the AdamW optimizer for the Graph Transformer model with an initial learning rate of 2e-4 and an ending learning rate of 1e-9. The hidden dimensions are set to 128 with a dropout of 0.3. The weight decay is 0.01, and the batch size of 32. To evaluate the performance of SIGNNet, we use accuracy as an evaluation metric. We implement our method using PyTorch Geometric (PyG) [58] and PyTorch [59]. The datasets are publicly available on PyG, and we use the same node and label attributes as provided in these datasets. We divide each dataset into 60% for training, 20% for validation, and 20% for testing to maintain consistent evaluation across the model.

4.3.2 Performance on Homophilic Datasets

From the results presented in Table 4.2, it is clear that SIGNNet achieves state-of-the-art performance on three out of the five datasets, which underscores the effectiveness of our proposed method. Our analysis compares methods based on both graph neural networks and graph transformers. Unlike Non-local GNNs [53], which mainly focus on heterophilic datasets and tend to overlook local network information, our method enhances node features by aggregating both structural and semantic information of the network. This is achieved through the use of connection scores and class-centric features, leading to improved performance. Specifically, the accuracy of Non-local GNNs on the Cora, Citeseer, Pubmed, CS, and Photo datasets are 88.50%, 76.20%, 89.06%, 94.28%, and 94.18% respectively. The performance of the PubMed dataset, while competitive, reveals certain challenges specific to its structure. Its sparse structure, with few connections and low clustering, makes it challenging for our proposed method to effectively gather and spread information but less critical for baselines focused on overall graph properties. Also, the lack of influential nodes in PubMed makes it hard to capture wider network patterns, a limitation less significant for some baseline models. Moreover, the sampling strategy employed in our approach may hinder the effectiveness of subgraph generation, impacting overall performance.

In contrast, SIGNNet surpasses Non-local GNNs on four out of these five datasets, with improvements of 3.95%, 4.94%, 3.08%, and 0.66% for Cora, Citeseer, CS, and Photo respectively. These results demonstrate the enhanced capability of SIGNNet in handling datasets where similar nodes are more likely to be connected, thereby providing a more accurate classification performance. AGT [40] demonstrates strong performance in node classification but encounters scalability challenges because it processes the entire graph for each classification task. To address this, our proposed method employs a sampling technique that not only mitigates scalability issues but also emphasizes the importance of neighboring nodes. This focus enhances the understanding of local node interactions and improves performance on homophilic datasets, where similar nodes are more likely to be connected. The accuracies of AGT on

Table 4.2: Comparison of models on homophilic datasets (Accuracy in %)

Model	**Cora**	**Citeseer**	**PubMed**	**CS**	**Photo**
Graph Neural Network-based Methods					
GCN [3]	86.92±1.33	76.13±1.51	88.42±0.50	89.11±0.70	85.94±1.18
GAT [50]	87.34±1.14	75.75±1.86	86.33±0.48	88.53±0.54	87.13±1.00
APPNP [19]	87.75±1.30	76.53±1.61	86.52±0.61	94.49±0.07	94.32±0.14
GCNII [20]	86.08±2.18	74.75±1.76	**90.15±0.43**	84.23±0.78	67.06±1.74
ACM-GCN [51]	87.91±0.95	77.41±1.78	90.30±0.52	94.83±0.24	90.34±1.82
GATv2 [52]	87.25±0.89	75.72±1.30	85.75±0.55	88.46±0.61	81.52±3.23
Non-local GNNs [53]	88.50±1.80	76.20±1.60	89.06±0.27	94.28±1.82	94.18±0.19
FSGNN [37]	88.23±1.17	77.40±1.93	89.78±0.38	95.15±0.48	92.58±0.78
SNGNN++ [4]	88.13±0.98	77.75±1.75	87.23±0.42	92.26±1.28	93.67±0.97
UniG-Encoder [54]	88.49±1.10	77.78±1.70	89.76±0.46	94.57±1.93	92.92±1.27
Geom-GNN [46]	85.35±1.57	78.90±1.15	89.95±0.47	95.64±2.81	93.35±1.40
Graph Transformer-based Methods					
SAN [10]	81.91±3.42	69.63±3.76	81.79±0.98	94.51±0.15	94.86±0.10
UniMP [55]	84.18±1.39	75.00±1.59	88.56±0.32	94.20±0.34	92.49±0.47
DET [40]	86.30±1.41	75.37±1.41	86.28±0.44	93.34±0.31	91.44±0.49
AGT [40]	81.65±0.41	70.95±0.62	81.48±1.69	87.16±2.18	88.49±0.92
NAGphormer [56]	85.77±1.35	73.69±1.48	89.70±0.19	95.75±0.09	**95.49±0.11**
Gapformer [31]	87.37±0.76	76.21±1.47	88.49±0.44	94.48±0.36	92.34±0.63
SoftGNN [57]	86.34±0.85	76.91±1.57	89.45±0.35	95.38±1.27	92.17±0.54
SIGNNet	**92.45±1.13**	**81.14±0.52**	87.89±2.10	**97.36±0.81**	94.84±0.61

the Cora, Citeseer, Pubmed, CS, and Photo datasets are 81.65%, 70.95%, 81.48%, 87.16%, and 88.49%, respectively. In contrast, SIGNNet outperforms AGT across all five datasets with significant improvements. Specifically, SIGNNet achieves enhancements of 10.80% for Cora, 10.19% for Citeseer, 6.41% for Pubmed, 10.20% for CS, and 6.35% for Photo. These results highlight SIGNNet's superior capability in handling diverse dataset characteristics effectively.

4.3.3 Performance on Heterophilic Datasets

From the results presented in Table 4.3, it is evident that SIGNNet achieves state-of-the-art performance on five of the six datasets, underscoring the robustness of our proposed method. FSGNN [37] primarily utilizes local features and neglects the structural property of the network. This approach generally results in diminished performance on heterophilic graphs, where nodes belonging to different classes are more likely to be connected. To address this limitation, our

Table 4.3: Comparison of models on heterophily datasets

Model	**Wisconsin**	**Texas**	**Actor**	**Cornell**	**Chameleon**	**Squirrel**
Graph Neural Network-based Methods						
GCN [3]	52.55±4.27	60.81±8.03	28.73±1.17	60.54±5.30	64.82±2.24	53.43±2.01
GAT [50]	57.45±3.51	62.16±4.52	28.33±1.13	61.89±5.05	60.26±2.50	40.72±1.55
APPNP [19]	55.29±3.90	61.62±5.37	29.42±0.81	73.51±0.37	54.30±0.77	32.37±0.51
GCNII [20]	52.54±7.32	58.91±4.32	25.40±0.97	77.86±3.79	63.56±3.04	38.47±1.58
ACM-GCN [51]	88.28±3.64	87.99±4.64	36.08±1.03	85.14±6.07	66.93±1.85	54.40±1.88
GATv2 [52]	52.74±3.96	60.54±4.55	28.79±1.47	50.27±8.97	62.20±2.11	50.80±3.01
Non-local GNNs [53]	87.30±4.30	85.40±3.80	37.90±1.30	84.90±5.70	70.10±2.90	59.00±1.20
FSGNN [37]	88.43±3.22	87.57±4.86	35.75±0.96	87.84±6.19	78.95±0.86	74.10±1.89
SNGNN++ [4]	89.02±3.36	88.65±4.90	34.20±1.26	87.84±4.81	80.83±0.94	**75.48±1.98**
UniG-Encoder [54]	87.84±3.90	85.95±3.90	35.79±0.39	86.75±6.56	81.06±1.21	74.39±2.17
Geom-GNN [46]	64.51±3.58	66.76±2.78	31.46±1.25	60.54±3.67	60.00±2.81	38.15±0.92
Graph Transformer-based Methods						
SAN [10]	51.37±3.08	60.17±6.66	27.12±2.59	50.85±8.54	48.74±5.25	46.36±3.27
UniMP [55]	79.60±5.41	73.51±8.44	35.15±0.84	66.48±12.5	69.17±0.18	65.84±2.37
DET [32]	54.90±6.56	56.76±4.98	28.94±0.64	72.18±7.14	67.94±3.71	64.26±0.87
AGT [40]	83.20±4.10	79.14±4.29	35.74±0.51	73.28±4.82	75.83±3.84	73.67±3.84
NAGphormer [56]	62.55±6.22	63.51±6.53	34.33±0.94	56.22±8.08	75.17±3.85	68.27±3.91
Gapformer [31]	83.53±3.42	80.27±4.01	36.90±0.82	77.57±3.43	81.04±1.65	70.98±2.31
SoftGNN [57]	88.63±3.37	88.11±4.39	36.68±0.86	78.92±3.78	81.67±1.81	74.16±0.39
SIGNNet	**90.20±1.66**	**92.24±2.81**	**39.82±1.25**	**91.35±0.48**	**83.97±0.37**	73.49±0.51

method incorporates a compatibility matrix that facilitates information sharing among similar nodes and integrates class-centric features to provide a comprehensive view of each class. This dual strategy enhances our understanding of the network and significantly improves performance. The accuracies of FSGNN on the Wisconsin, Texas, Actor, Cornell, Chameleon, and Squirrel datasets are 88.43%, 87.57%, 35.75%, 87.84%, 78.95%, and 74.10%, respectively. In stark contrast, SIGNNet demonstrates superior performance over FSGNN on five of these six datasets, with accuracy improvements of 1.77% for Wisconsin, 4.67% for Texas, 4.07% for Actor, 3.51% for Cornell, and 5.02% for Chameleon. These enhancements highlight SIGNNet's effective adaptation to the challenges posed by heterophilic datasets and its ability to leverage both local and global network properties for improved classification results. Gapformer [31] faces challenges like quadratic complexity and noise from irrelevant nodes in node classification, which SIGNNet addresses effectively. SIGNNet reduces complexity through a Personalized PageRank-based node sampling method that limits the number of nodes processed. It also incorporates a Structure-Aware Multi-Head Attention (SA-MHA) mechanism, which integrates structural information into the attention process, focusing on topologically significant nodes and reducing noise from irrelevant ones. Additionally, SIGNNet enhances node representation by employing connection scores and class-centric features, allowing for a better capture of both local and global graph properties. These improvements enable SIGNNet to handle the graph structure more adeptly, thereby improving its performance in node classification tasks across various datasets. The accuracies of Gapformer on the Wisconsin, Texas, Actor, Cornell, Chameleon, and Squirrel datasets are 83.53%, 80.27%, 36.90%, 77.57%, 81.04%, and 70.98%, respectively. In clear contrast, SIGNNet significantly outperforms Gapformer across all these datasets, demonstrating notable improvements. Specifically, SIGNNet achieves enhancements of 6.67% for Wisconsin, 11.97% for Texas, 2.92% for Actor, 13.78% for Cornell, 2.93% for Chameleon, and 2.51% for Squirrel, underlining its superior performance across a range of graph environments. Furthermore, SIGNNet is compared against NAGphormer[56], SoftGNN [57], SAN [10], and many other advanced methods in node classification. SIGNNet consistently outperforms these methods by significant margins across all evaluation metrics. This thorough comparison emphasizes the robustness and effectiveness of SIGNNet specifically in node classification tasks. By surpassing existing benchmarks, SIGNNet proves its capability to effectively utilize graph structural and feature information, thus improving node classification performance across diverse datasets.

4.4 Ablation Studies

To show the effectiveness of individual components employed in our proposed method, we perform ablation studies by removing one component at a time from our proposed framework and evaluate how each contributes to performing on three datasets: Cora, Citeseer, and Actor. It provides insight into the role any individual component plays in the feature representation and the classification performance of the model. Additionally, we assess the behavior of the model with a different number of epochs, visualize learned embeddings, and examine the effects of varying transformer layers and subgraph counts to gain insights into the adaptability and effectiveness of the framework.

4.4.1 Modal Component Analysis

To understand the importance of each module in the framework, we remove one module at a time and investigate its effect on the framework, comparing the results on the homophily and heterophily datasets as shown in Table 4.4 and Table 4.5. Next, we discuss the effect of each module in the framework.

a) Impact of Neighborhood Influence Learning Feature: It works by using message passing to add local features from neighboring nodes into each node's feature, making the node features more informative. By including this local information, we enhance the overall understanding of the node's position in the network. If we remove this module, the node features will not include any local context.

b) Impact of Connection-aware Feature Representation: We incorporate broader information into each node's feature by adding its connection score, which measures how influential a node is within the network. Connection score helps capture the importance of a node in its immediate surroundings. If this part is removed, the node features will lose their global context, reducing their ability to reflect the node's influence within the network.

c) Impact of Class-centric Feature Representation: We incorporate global information into each node by aggregating class-centric features. The class-centric information is calculated by taking the average of the feature vectors for all nodes that belong to the same class. If we remove this module, the node features will be missing the broader contextual understanding of the graph structure, and this would negatively impact the method's performance.

Table 4.4: Performance on homophilic datasets across different configurations

Configuration	**Cora**	**Citeseer**	**Pubmed**	**CS**	**Photo**
W/o GCN	87.45±1.10	74.44±0.57	83.54±1.50	94.64±0.78	93.74±0.51
W/o C_{score}	91.95±0.85	80.97±0.45	87.12±1.24	96.81±0.65	94.17±0.49
W/o CR	90.77±1.07	78.79±0.35	86.14±2.07	96.24±0.62	93.88±0.55
SIGNNet	**92.45±1.13**	**81.14±0.52**	**87.89±2.10**	**97.36±0.81**	**94.84±0.61**

W/o GCN = Without graph convolutional network, W/o C_{score} = Without connection score, W/o CR = Without class representative

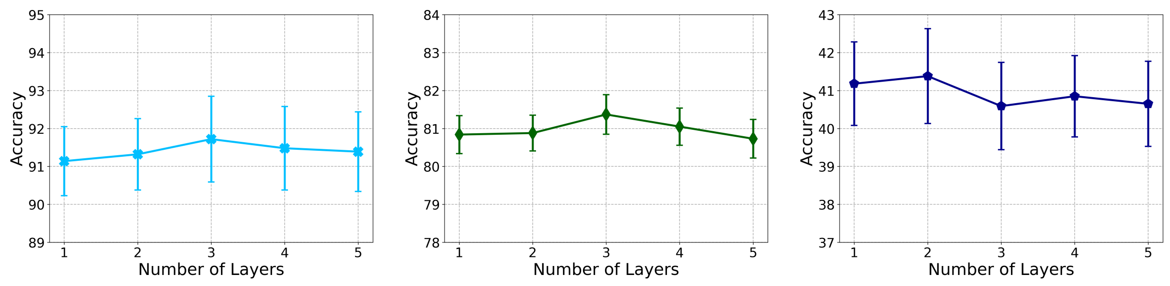
Table 4.5: Performance on heterophilic datasets across different configurations

Configuration	**Actor**	**Wisconsin**	**Texas**	**Cornell**	**Chameleon**	**Squirrel**
W/o GCN	36.15±1.17	84.79±1.42	87.05±2.53	86.84±0.35	80.28±0.29	70.68±0.44
W/o C_{score}	38.14±0.95	89.57±1.37	90.87±2.63	91.15±0.37	83.17±0.32	72.78±0.37
W/o CR	37.15±1.15	89.46±1.47	91.89±2.61	89.75±0.41	83.21±0.28	73.29±0.46
SIGNNet	**39.82±1.25**	**90.20±1.66**	**92.24±2.81**	**91.35±0.48**	**83.97±0.37**	**73.49±0.51**

W/o GCN = Without graph convolutional network, W/o C_{score} = Without connection score, W/o CR = Without class representative

4.4.2 Performance Comparison across Various Layers

In this subsection, we evaluate the classification performance of our method by varying the number of transformer layers from 1 to 5. As the number of layers in-



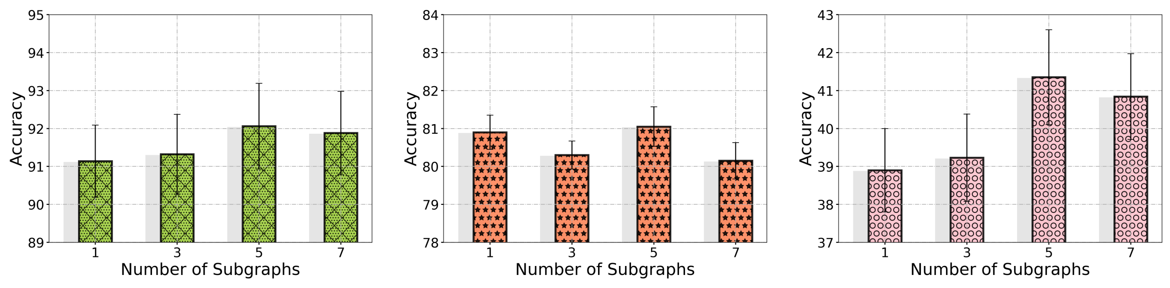
(a) Cora (b) Citeseer (c) Actor

Figure 4.1: Performance analysis across varying transformer layers

creases, the transformer captures more structural information from the network, generally improving classification performance. However, it can also cause overfitting. For example, Cora and Citeseer achieve better results with 3 layers, while the Actor dataset performs better with 2 layers. Thus, the selection of layers must be done carefully, taking into account the structural characteristics of the dataset for better performance. The following graphs Figure 4.1(a), Figure 4.1(b), and Figure 4.1(c) shows how the number of layers affects the performance of the Cora, Citeseer, and Actor datasets respectively.

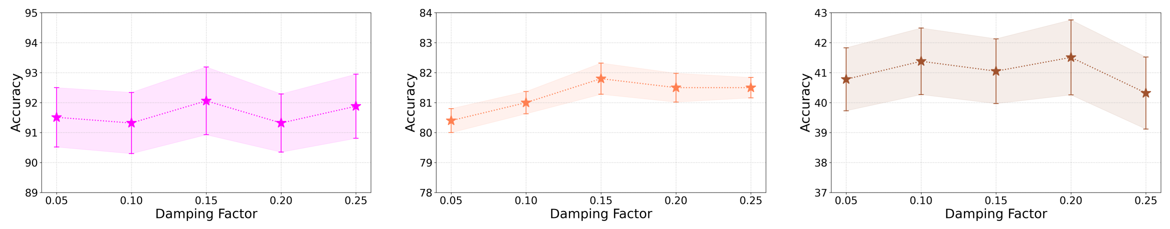
4.4.3 Performance Comparison across Various Subgraphs

In this section, we analyze how changing the number of subgraphs affects the SIGN-Net performance. The model exhibits the lowest performance with a single subgraph, while performance steadily improves as the number of subgraphs increases, reaching its peak at five subgraphs. Further increases lead to a decline in performance. Figure 4.2(a), Figure 4.2(b), and Figure 4.2(c) display the results for subgraph counts of 1, 3, 4, and 7 across the Cora, Citeseer, and Actor datasets respectively, highlighting significant variations in performance.



(a) Cora (b) Citeseer (c) Actor

Figure 4.2: Performance analysis with varying subgraph counts

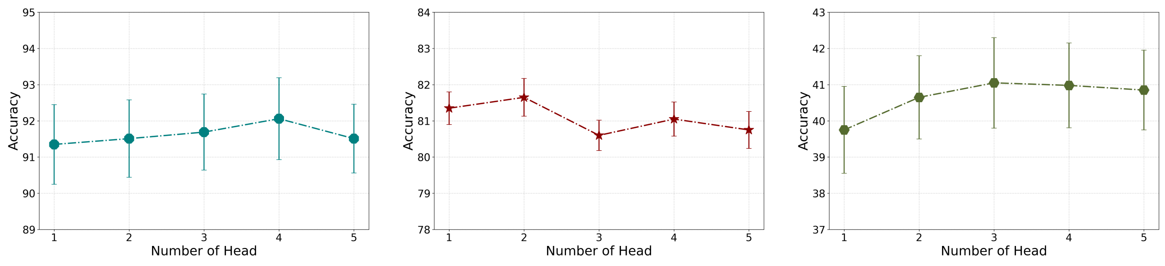


(a) Cora (b) Citeseer (c) Actor

Figure 4.3: Performance analysis with varying damping factor value

4.4.4 Performance Comparison across Various values of Damping Factor

We analyze the role of the damping factor in the Personalized PageRank (PPR) algorithm and its impact on model performance across different datasets. The restart probability controls the balance between local and global exploration in a graph. To guarantee a bias toward nodes nearby, the random walker either returns to a predetermined starting node at each step or keeps investigating nearby nodes. When the value of restart probability is small then it encourages a more comprehensive examination of the graph, and when the number is high then it more focuses on local neighbors. Cora's performance improves with a higher restart chance as shown in Figure 4.3(a), and Citeseer gives the best performance when the value is 0.15 as shown in Figure 4.3(b). The Actor's accuracy, on the other hand, varies, suggesting a more intricate link as shown in Figure 4.3(c). These results show that the damping factor is an important parameter.

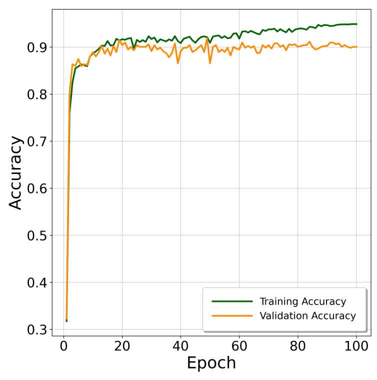


(a) Cora (b) Citeseer (c) Actor

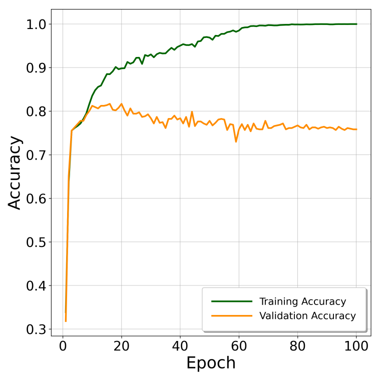
Figure 4.4: Performance analysis with varying number of heads in graph transformer

4.4.5 Performance Comparison across Various Numbers of Attention Heads

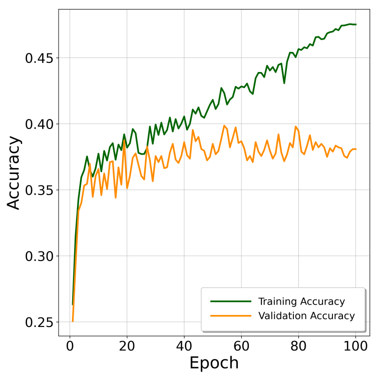
To evaluate the performance gain achieved by employing different numbers of attention heads. We carried out our experiments on three benchmark datasets such as Cora, Citeseer, and Actor to show the significance. The number of attention heads in a graph transformer is crucial because it enables the model to focus on various node-to-node interaction characteristics by capturing several attention patterns at once. The large number of attention heads helps to capture complex patterns and identify more intricate dependencies within the graph. However, too many heads can raise computational overhead and model complexity, which could result in overfitting or performance degradation. The accuracy varies with the number of attention heads, highlighting dataset-specific trends. We observe a different trend in all the datasets as the number of attention heads increases. In Cora, a general trend is observed, the performance of the model increases as the number of heads increases as shown in Figure 4.4(a), while Citeseer reaches its peak performance at a specific value before stabilizing as shown in Figure 4.4(b). Due to the dense nature of Actor dataset has a more complex pattern, where accuracy increases at a smaller number of heads and then gradually declines with larger values as shown in Figure 4.4(c). These results indicate that the optimal number of attention heads is dataset-dependent and is required to achieve good performance.



(a) Cora



(b) Citeseer



(c) Actor

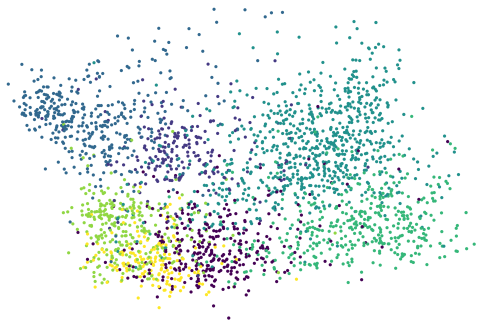
Figure 4.5: Test and validation accuracy over epochs

4.4.6 Accuracy Comparison Over Epochs

In this subsection, we illustrate the performance of the method over time. In the initial phase of learning on the Cora dataset, the training accuracy increases rapidly as the model understands the dataset. Validation accuracy is also increasing but at a slower pace shown in Figure 4.5(a). As the training proceeds, both the training and validation accuracies converge, indicating that the model learns well and does not overfit. This trend indicates that the model handles the dataset well and achieves good classification results after several epochs. From Figure 4.5(b), we can clearly see that the Citeseer dataset is a little bit more complex than the Cora dataset as during the training validation accuracy is less steady, going up and down across the epochs, which shows that the model has a harder time generalizing to new data compared to Cora dataset. The Actor dataset is a heterophilic graph, and the Figure 4.5(c) clearly shows how difficult it is to find nodes correctly for the model. Both training and validation accuracy start at low points, while training accuracy improves steadily, validation accuracy changes do not fully stabilize, reflecting the difficulties of the Actor dataset, where the model is challenged to always achieve better results.

4.4.7 Cora Dataset Visualization

In this subsection, we present a wide view of the Cora dataset after the model has learned through structural and feature-based transformations. This visualization


(a) Before Training

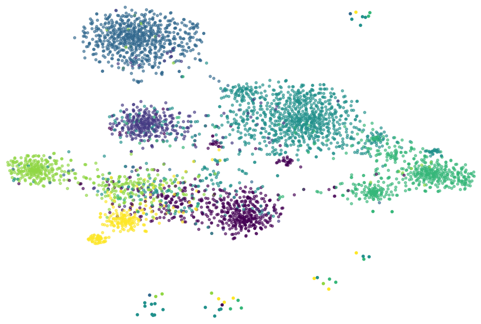

(b) After Training

Figure 4.6: Cora dataset visualization before and after training

captures the initial representation of the node embeddings before training shown in Figure 4.6(a), showing minimal separation between classes. After training with the proposed method, the node embeddings are improved by capturing both the structure and semantic contexts of the network, highlighting how the method enhances the embeddings such that nodes that belong to the same class are closer to each other shown in Figure 4.6(b). This progression illustrates the method's ability to effectively organize and classify the nodes, leading to more accurate node classification.

Chapter 5

Discussion

We present SIGNNet, a novel framework designed for node classification by integrating both semantic and structural properties of the network into the node features. SIGNNet leverages a Personalized PageRank algorithm to generate subgraphs specific to each node which addresses scalability issues associated with Graph Transformers. Additionally, we propose a structure-aware mechanism that aggregates the structural information in the attention process of the network. Our proposed method offers significant advancements in node classification by employing some novel techniques.

5.1 Limitations and Future Work

The proposed framework shows promising outcomes. However, some technical limitations and future directions are described below.

5.1.1 Limitations

- **Dependency on Labeled Graphs:** This method is not suitable for unsupervised and semi-supervised tasks as we are dealing with the dataset in which nodes are labeled.

- **Partial Loss of Global Structural Information:** The node sampling method preserves key features for efficiency while improving the scalability of the graph transformer with little effect on larger network information.

- **Disconnected Graphs:** A significant challenge arises when dealing with disconnected graphs, as the absence of connections between components restricts the flow of information. Since graph convolutional networks rely on neighborhood aggregation for feature propagation, nodes in different disconnected components remain isolated, limiting the model's ability to learn meaningful representations across the entire graph. Additionally, our sampling strategy may inadvertently select nodes only from the same component, leading to biased subgraph formation, which affects generalization.

- **Edge Directions and Temporal Information:** Our framework does not explicitly incorporate edge directions or temporal dependencies, which may limit its applicability to dynamic or directed graphs. In real-world scenarios, edge directions are crucial in representing asymmetric relationships, such as citation networks or knowledge graphs, where information should flow in a specific direction. Similarly, time-evolving graphs require models that account for the temporal ordering of edges to capture evolving node interactions. The lack of these mechanisms may hinder the model's performance on tasks requiring sequential or directional information processing.

- **Class Imbalance:** The presence of a class imbalance in node classification tasks can negatively impact model performance. When certain classes are underrepresented, the model may struggle to learn discriminative features for minority classes, leading to biased predictions. Prior studies [60, 61, 62] have explored strategies to address class imbalance in GNNs, but our approach does not include explicit mechanisms for handling skewed class distributions. This remains an important direction for future work.

5.1.2 Future Work

SIGNNet primarily focuses only on node classification tasks. However, our proposed framework can be expanded to additional graph tasks such as:

[1] **Community Detection:** relies on identifying densely connected substructures within a graph. The graph convolutional network of our framework inherently preserves homophily by aggregating local neighborhood features, making it well suited to detect community structures [63]. By stacking multiple GCN layers, the model captures higher-order dependencies, which enhances the detection of hierarchical communities. Additionally, our class-centric feature aggregation can be adapted to generate community-level embeddings. When combined with graph-theoretic measures such as modularity and degree centrality, these embeddings can aid in detecting influential nodes or community hubs [64], improving the framework's applicability to community detection tasks.

[2] **Link Prediction:** Our framework can be extended to link prediction by modifying the subgraph extraction process to generate an induced subgraph centered around a node pair. Instead of sampling multiple subgraphs, we construct a single localized subgraph that encapsulates the structural context between the two nodes, ensuring that their direct and indirect connections are preserved. To model pairwise interactions effectively, the Graph Transformer's structure-aware attention mechanism is adapted to compute edge-level attention scores, allowing the model to dynamically assess the importance of potential links based on surrounding connectivity patterns [26]. This mechanism ensures that structurally significant edges contribute more to the learned representations. Additionally, our framework integrates positional encodings to capture the global structural roles of nodes within the induced subgraph. These encodings enable the model to retain essential topological information, ensuring that it can distinguish between existing and potential links based on their relative positions within the graph. While our framework exhibits strong potential for these tasks, we acknowledge that further empirical validation is necessary. Consequently, our thesis explicitly outlined this as a future research direction.

[3] **Graph Classification:** Our current framework employs the Graph Transformer primarily for node-level representations. To extend it for graph-level classifica-

tion, future work will explore hierarchical pooling techniques (e.g., DiffPool, TopK pooling) to generate compact graph embeddings. Additionally, incorporating global self-attention mechanisms could enhance structural awareness. Benchmarking on standard graph classification datasets will further assess its efficacy.

However, it is worth noting that not all components of our proposed method apply to every task; however, certain components can effectively address specific challenges. Further research and experimentation can help overcome these limitations, enhancing the method's effectiveness and suitability for real-world node classification tasks.

5.2 Model Complexity Analysis

In this section, we have conducted a comprehensive complexity analysis, evaluating both the time complexity and parameter count of our proposed method in comparison with multiple state-of-the-art methods. All experiments were conducted on a Linux server equipped with an AMD Ryzen 5 5600X 6-core processor (3.70 GHz), 32 GB RAM, and an NVIDIA GeForce RTX 3050 GPU. The detailed comparison of computation time and parameter counts across different models is presented in Table 5.1. Our analysis reveals interesting trade-offs between model complexity and performance. When compared to GNN-based methods, SIGNNet has a larger parameter count but achieves superior accuracy compared to other models. Notably, while GCNII [20] demands more computational time than SIGNNet, it fails to match SIGNNet's performance benchmarks. In the context of graph transformer-based approaches, SIGNNet's parameter count exceeds most comparative methods, with Gapformer [31] being the sole exception. Despite its larger architecture, Gapformer requires significantly more computation time than SIGNNet while achieving lower performance. SIGNNet maintains competitive computation times relative to other models in the benchmark suite. In contrast, the SIGNNet framework optimally balances computational efficiency and model complexity. Despite having a moderate number of parameters (0.700320 million), SIGNNet delivers a competitive inference time of 0.009067

Table 5.1: Inference time and number of parameters of various methods

Models	**Time (seconds)**	**Parameters (millions)**
GNN-based Methods		
GCN [3]	0.007458	0.184
GAT [50]	0.007691	0.185
GCNII [20]	0.043489	0.354
UniG [54]	0.001305	0.369
FSGNN [37]	0.001446	0.646
Graph Transformer-based Methods		
Gapformer [31]	0.014067	0.786
SAN [10]	0.008996	0.509
DET [32]	0.008757	0.490
UniMP [55]	0.001662	0.333
SIGNNet	**0.009067**	**0.700**

seconds. Overall, SIGNNet achieves superior performance with a balanced level of computational complexity, making it well-suited for real-world deployment scenarios with limited computational resources. Additionally, we provide a detailed breakdown of the computational complexity of our proposed framework's key components such as SA-MHA and Personalized PageRank sampling. We break down the computational cost of its key components as follows:

1. Neighborhood Influence Feature Learning Module: The complexity of this module is $\mathcal{O}(l|E|d^2)$, where l represents the number of layers, $|E|$ is the number of edges, and d is the feature dimension. This reflects the computational cost of aggregating neighborhood information across multiple layers.

2. Connection Score Computation: Calculating the connection score for each node involves iterating through all edges, leading to a complexity of $\mathcal{O}(n^2)$, where n is the number of nodes in the graph. Once these scores are computed, integrating them into node features requires additional $\mathcal{O}(nd)$ operations.

3. Class-centric Feature Computation: Extracting Class-centric features involves two main steps:

- Computing the initial feature representation incurs a complexity of $\mathcal{O}(nd)$.
- Generating the similarity-driven feature matrix requires $\mathcal{O}(ndm)$ operations, where m is the number of classes. Since m is a constant, the overall complexity for this step simplifies to $\mathcal{O}(nd)$.

4. Personalized Page Rank Sampling Technique: The sampling operation within SIGNNet involves an exhaustive search across node relationships, which can be computed in time $\mathcal{O}(\text{mul}(n))$, where $\text{mul}(n)$ denotes the time complexity of matrix multiplication for matrices of order $n \times n$.

5. Structural-aware Attention Mechanism: The attention mechanism operates with a complexity of $\mathcal{O}(nqk_1)$, where q is the number of sampled nodes, and k_1 represents the number of nodes in each subgraph. Since q and k_1 are constants and significantly smaller than n, this component simplifies to $\mathcal{O}(n)$, ensuring scalability concerning the sampled subgraph size.
The computational demands of SIGNNet are comparable to those of other state-of-the-art methods. Furthermore, our evaluations indicate that SIGNNet consistently achieves high-performance metrics across most datasets.

Chapter 6

Conclusion

We propose SIGNNet, a novel framework for node classification that effectively integrates both structural and semantic information into node features, which initially contain only attribute-level data. Our approach enriches node representations through three key mechanisms: neighborhood-influenced feature learning, connection-aware representation, and class-centric feature encoding. The neighborhood-influenced feature learning component utilizes Graph Convolutional Networks (GCNs) to embed local structural context into node features, allowing the model to better differentiate between classes based on local connectivity patterns. Meanwhile, the connection-aware and class-centric modules encode broader network context and node relevance, resulting in richer and more discriminative feature representations. To address the computational limitations associated with Graph Transformers, SIGNNet incorporates a Personalized PageRank-based node sampling strategy, which efficiently selects the most influential nodes within a subgraph. This approach significantly reduces computational overhead without sacrificing performance. Furthermore, we introduce a Structure-Aware Multi-Head Attention mechanism, which combines feature similarity with graph structural information to enhance the effectiveness of the attention process. Experimental results demonstrate that SIGNNet consistently outperforms existing state-of-the-art models across both homophilic and heterophilic graph datasets. Compared to Gapformer, SIGNNet achieves notable accuracy improvements on homophilic datasets: 5.08% on Cora, 4.93% on Citeseer, 2.88% on CS, and 2.5% on

Photo. On heterophilic datasets, it outperforms Gapformer by 6.67% on Wisconsin, 11.97% on Texas, 2.92% on Actor, 13.78% on Cornell, 2.93% on Chameleon, and 2.51% on Squirrel. In comparison with SNGNN++, SIGNNet achieves gains of 4.32% on Cora, 3.39% on Citeseer, 0.66% on Pubmed, 5.10% on CS, and 1.17% on Photo among homophilic datasets. For heterophilic datasets, it surpasses SNGNN++ by 1.18% on Wisconsin, 3.59% on Texas, 5.62% on Actor, 3.51% on Cornell, and 3.14% on Chameleon. These results clearly demonstrate the strength of our approach. By integrating both local and global contextual information into node features while addressing the scalability limitations of Graph Transformers, SIGNNet provides a robust, efficient, and accurate solution for node classification in diverse graph settings.

Bibliography

[1] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," 2020.

[2] G. Nikolentzos and M. Vazirgiannis, "Learning structural node representations using graph kernels," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, 2021.

[3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017, pp. 424–457.

[4] M. Zou, Z. Gan, R. Cao, C. Guan, and S. Leng, "Similarity-navigated graph neural networks for node classification," *Information Sciences*, vol. 633, 2023.

[5] S. Yuan, H. Zeng, Z. Zuo, and C. Wang, "Overlapping community detection on complex networks with graph convolutional networks," *Computer Communications*, vol. 199, pp. 62–71, 2023.

[6] B. Li and D. Pi, "Learning deep neural networks for node classification," *Expert Systems with Applications*, vol. 137, pp. 324–334, 2019.

[7] Y. Xie, Y. Liang, M. Gong, A. K. Qin, Y. S. Ong, and T. He, "Semisupervised graph neural networks for graph classification," *IEEE Transactions on Cybernetics*, vol. 53, 2023.

[8] J. H. Giraldo, K. Skianis, T. Bouwmans, and F. D. Malliaros, "On the trade-off between over-smoothing and over-squashing in deep graph neural networks," in

International Conference on Information and Knowledge Management, Proceedings, 2023, pp. 566–576.

[9] S. Yun, M. Jeong, S. Yoo, S. Lee, S. S. Yi, R. Kim, J. Kang, and H. J. Kim, "Graph transformer networks: Learning meta-path graphs to improve gnns," *Neural Networks*, vol. 153, 2022.

[10] D. Kreuzer, D. Beaini, W. Hamilton, V. Létourneau, and P. Tossou, "Rethinking graph transformers with spectral attention," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 21 618–21 629.

[11] T. Zhao, X. Zhang, and S. Wang, "Graphsmote: Imbalanced node classification on graphs with graph neural networks," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, ser. WSDM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 833–841.

[12] B. Molokwu, S. B. Shuvo, N. C. Kar, and Z. Kobti, "Node classification and link prediction in social graphs using rlvecn," in *Proceedings of the 32nd International Conference on Scientific and Statistical Database Management*, ser. SSDBM '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 833–841.

[13] G. Li, M. Muller, G. Qian, I. C. Delgadillo, A. Abualshour, A. Thabet, and B. Ghanem, "Deepgcns: Making gcns go as deep as cnns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, 2023.

[14] R. Burkholz, J. Quackenbush, and D. Bojar, "Using graph convolutional neural networks to learn a representation for glycans," *Cell Reports*, vol. 35, 2021.

[15] C. S. Raghaw, A. Sharma, S. Bansal, M. Z. U. Rehman, and N. Kumar, "Cotconet: An optimized coupled transformer-convolutional network with an adaptive graph reconstruction for leukemia detection," *Computers in Biology and Medicine*, vol. 179, p. 108821, Sep. 2024. [Online]. Available: http://dx.doi.org/10.1016/j.compbiomed.2024.108821

[16] S. Bansal, N. S. Singh, S. S. Dar, and N. Kumar, "Mmcfnd: Multimodal multilingual caption-aware fake news detection for low-resource indic languages," 2024. [Online]. Available: https://arxiv.org/abs/2410.10407

[17] M. Z. U. Rehman, S. Zahoor, A. Manzoor, M. Maqbool, and N. Kumar, "A context-aware attention and graph neural network-based multimodal framework for misogyny detection," *Information Processing & Management*, vol. 62, no. 1, p. 103895, Jan. 2025. [Online]. Available: http://dx.doi.org/10.1016/j.ipm.2024.103895

[18] S. Xiao, S. Wang, Y. Dai, and W. Guo, "Graph neural networks in node classification: survey and evaluation," *Machine Vision and Applications*, vol. 33, 2022.

[19] J. Klicpera, A. Bojchevski, and S. Gunnemann, "Combining neural networks with personalized pagerank for classification on graphs," *arXiv*, 2018.

[20] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 1725–1735.

[21] H. Li, Y. Wang, Z. Xiao, J. Yang, C. Zhou, M. Zhang, and W. Ju, "Disco: Graph-based disentangled contrastive learning for cold-start cross-domain recommendation," *CoRR*, vol. abs/2412.15005, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2412.15005

[22] Y. Wang, X. Luo, C. Chen, X.-S. Hua, M. Zhang, and W. Ju, "Disensemi: Semi-supervised graph classification via disentangled representation learning," *IEEE transactions on neural networks and learning systems*, vol. PP, 08 2024.

[23] Y. Wang, Y. Li, S. Li, W. Song, J. Fan, S. Gao, L. Ma, B. Cheng, X. Cai, S. Wang, and M. Zhang, "Deep graph mutual learning for cross-domain recommendation," in *Database Systems for Advanced Applications: 27th International Conference, DASFAA 2022, Virtual Event, April 11–14, 2022,*

Proceedings, Part II. Berlin, Heidelberg: Springer-Verlag, 2022, p. 298–305. [Online]. Available: https://doi.org/10.1007/978-3-031-00126-0_22

[24] Y. Wang, J. Shen, Y. Song, S. Wang, and M. Zhang, "He-sne: Heterogeneous event sequence-based streaming network embedding for dynamic behaviors," in *2022 International Joint Conference on Neural Networks (IJCNN)*, 2022, pp. 1–8.

[25] A. J. Fofanah, D. Chen, L. Wen, and S. Zhang, "Addressing imbalance in graph datasets: Introducing gate-gnn with graph ensemble weight attention and transfer learning for enhanced node classification," *Expert Systems with Applications*, vol. 255, p. 124602, 2024.

[26] H. Shomer, Y. Ma, H. Mao, J. Li, B. Wu, and J. Tang, "Lpformer: An adaptive graph transformer for link prediction," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 2686–2698.

[27] M. Li, Y. Zhang, W. Zhang, S. Zhao, X. Piao, and B. Yin, "Csat: Contrastive sampling-aggregating transformer for community detection in attribute-missing networks," *IEEE Transactions on Computational Social Systems*, vol. 11, 2024.

[28] G. N. Chandrika, K. Alnowibet, K. S. Kautish, E. S. Reddy, A. F. Alrasheedi, and A. W. Mohamed, "Graph transformer for communities detection in social networks," *Computers, Materials and Continua*, vol. 70, 2022.

[29] S. S. Dar, M. Z. U. Rehman, K. Bais, M. A. Haseeb, and N. Kumar, "A social context-aware graph-based multimodal attentive learning framework for disaster content classification during emergencies," *Expert Systems with Applications*, vol. 259, p. 125337, 2025.

[30] S. S. Dar, M. K. Karandikar, M. Z. U. Rehman, S. Bansal, and N. Kumar, "A contrastive topic-aware attentive framework with label encodings for post-disaster resource classification," *Knowledge-Based Systems*, vol. 304, p. 112526, 2024.

[31] C. Liu, Y. Zhan, X. Ma, L. Ding, D. Tao, J. Wu, and W. Hu, "Gapformer: Graph transformer with graph pooling for node classification," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, E. Elkind, Ed. International Joint Conferences on Artificial Intelligence Organization, 8 2023, pp. 2196–2205, main Track.

[32] L. Guo, Q. Zhang, and H. Chen, "Unleashing the power of transformer for graphs," *arXiv preprint arXiv:2202.10581*, 2022.

[33] Z. Gu, X. Luo, J. Chen, M. Deng, and L. Lai, "Hierarchical graph transformer with contrastive learning for protein function prediction," *Bioinformatics (Oxford, England)*, vol. 39, 06 2023.

[34] Q. Yuan, S. Chen, J. Rao, S. Zheng, H. Zhao, and Y. Yang, "Alphafold2-aware protein-dna binding site prediction using graph transformer," *Briefings in bioinformatics*, vol. 23, 01 2022.

[35] Z. Huang, Y. Tang, and Y. Chen, "A graph neural network-based node classification model on class-imbalanced graph data," *Knowledge-Based Systems*, vol. 244, 2022.

[36] J. Wu, J. He, and J. Xu, "Demo-net: Degree-specific graph neural networks for node and graph classification," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019, pp. 406–415.

[37] S. K. Maurya, X. Liu, and T. Murata, "Simplifying approach to node classification in graph neural networks," *Journal of Computational Science*, vol. 62, 2022.

[38] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T. Y. Liu, "Do transformers really perform bad for graph representation?" in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 28 877–28 888.

[39] M. Yang, H. Wang, Z. Wei, S. Wang, and J. R. Wen, "Efficient algorithms for personalized pagerank computation: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, 2024.

[40] X. Ma, Q. Chen, Y. Wu, G. Song, L. Wang, and B. Zheng, "Rethinking structural encodings: Adaptive graph transformer for node classification task," in *Proceedings of the ACM Web Conference 2023*, ser. WWW '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 533–544.

[41] J. Saramäki, M. Kivelä, J.-P. Onnela, K. Kaski, and J. Kertész, "Generalizations of the clustering coefficient to weighted complex networks," *Phys. Rev. E*, vol. 75, p. 027105, Feb 2007.

[42] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," *World Wide Web Internet And Web Information Systems*, vol. 54, 1998.

[43] M. Mcpherson, L. Smith-Lovin, and J. Cook, "Birds of a feather: Homophily in social networks," *Annu Rev Sociol*, vol. 27, pp. 415–444, 08 2001.

[44] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, 2008.

[45] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," *CoRR*, vol. abs/1811.05868, 2018.

[46] H. Pei, B. Wei, K. C. C. Chang, Y. Lei, and B. Yang, "Geom-gcn: Geometric graph convolutional networks," in *8th International Conference on Learning Representations, ICLR 2020*, 2020, pp. arXiv–2002.

[47] J. Tang, J. Sun, C. Wang, and Z. Yang, "Social influence analysis in large-scale networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009, pp. 807–816.

[48] A. R. Benson, R. Abebe, M. T. Schaub, A. Jadbabaie, and J. Kleinberg, "Simplicial closure and higher-order link prediction," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 115, 2018.

[49] B. Rozemberczki, C. Allen, and R. Sarkar, “Multi-scale attributed node embedding,” *Journal of Complex Networks*, vol. 9, 2021.

[50] P. Veličković, A. Casanova, P. Liò, G. Cucurull, A. Romero, and Y. Bengio, “Graph attention networks,” in *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018, pp. 10–48 550.

[51] S. Luan, C. Hua, Q. Lu, J. Zhu, M. Zhao, S. Zhang, X. Chang, and D. Precup, “Is heterophily A real nightmare for graph neural networks to do node classification?” *CoRR*, vol. abs/2109.05641, 2021.

[52] S. Brody, U. Alon, and E. Yahav, “How attentive are graph attention networks?” in *ICLR 2022 - 10th International Conference on Learning Representations*, 2022, pp. 357–362.

[53] M. Liu, Z. Wang, and S. Ji, “Non-local graph neural networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, 2022.

[54] M. Zou, Z. Gan, Y. Wang, J. Zhang, D. Sui, C. Guan, and S. Leng, “Unig-encoder: A universal feature encoder for graph and hypergraph node classification,” *Pattern Recognition*, vol. 147, 2024.

[55] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun, “Masked label prediction: Unified message passing model for semi-supervised classification,” in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Z.-H. Zhou, Ed. International Joint Conferences on Artificial Intelligence Organization, 8 2021, pp. 1548–1554, main Track.

[56] J. Chen, C. Liu, K. Gao, G. Li, and K. He, “Tokenized graph transformer with neighborhood augmentation for node classification in large graphs,” *arXiv*, 2023.

[57] H. Yang, J. Fang, J. Wu, D. Li, Y. Wang, and Z. Zheng, “Soft label enhanced graph neural network under heterophily,” *Knowledge-Based Systems*, vol. 309, p. 112861, 2025.

[58] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019, p. 10.

[59] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, 12 2019.

[60] L. Qu, H. Zhu, R. Zheng, Y. Shi, and H. Yin, "Imgagn: Imbalanced network embedding via generative adversarial graph networks," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ser. KDD '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1390–1398.

[61] Z. Wang, X. Ye, C. Wang, J. Cui, and P. S. Yu, "Network embedding with completely-imbalanced labels," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, 2021.

[62] T. Zhao, X. Zhang, and S. Wang, "Graphsmote: Imbalanced node classification on graphs with graph neural networks," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, ser. WSDM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 833–841. [Online]. Available: https://doi.org/10.1145/3437963.3441720

[63] N. S. Sattar and S. Arifuzzaman, "Community detection using semi-supervised learning with graph convolutional network on gpus," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 5237–5246.

[64] R. K. Behera, D. Naik, B. Sahoo, and S. K. Rath, "Centrality approach for community detection in large scale network," in *Proceedings of the 9th Annual ACM India Conference*, ser. COMPUTE '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 115–124.