Scalable Graph Learning with Graph Convolutional Networks and Graph Attention Networks: Addressing Class Imbalance Through Augmentation and Optimized Hyperparameter Tuning

Chaima Ahle Touate[1], Rachid El Ayachi[2], Mohamed Biniz[3]

Information Processing and Decision Support Laboratory-Department of Computer Science-Faculty of Sciences and Techniques, Sultan Moulay Slimane University, Beni Mellal, Morocco[1]

Department of Computer Science-Faculty of Sciences and Techniques, Sultan Moulay Slimane University, Beni Mellal, Morocco[2]

Department of Mathematics and Computer Science, Sultan Moulay Slimane University, Beni Mellal, Morocco[3]

***Abstract*—In this study, we propose a graph-based node classification to address challenges such as data scarcity, class imbalance, limited access to original textual content in benchmark datasets, semantic preservation, and model generalization in node classification tasks. Beyond simple data replication, we enhanced the Cora dataset by extracting content from its original PostScript files using a three-dimensional framework that combines in one pipeline NLP-based techniques such as PEGASUS paraphrase, synthetic model generation and a controlled subject aware synonym replacement. We substantially expanded the dataset to 17,780 nodes—representing an approximation of 6.57x scaling while maintaining semantic fidelity (WMD scores: 0.27-0.34). Our Bayesian Hyperparameter tuning was conducted using Optuna, along with k-fold cross-validation for a rigorous optimized model validation protocol. Our Graph Convolutional Network (GCN) model achieves 95.42% accuracy while Graph Attention Network (GAT) reaches 93.46%, even when scaled to a significantly larger dataset than the base. Our empirical analysis demonstrates that semantic-preserving augmentation helped us achieve better performance while maintaining model stability across scaled datasets, offering a cost-effective alternative to architectural complexity, making graph learning accessible to resource-constrained environments.**

Keywords—*Graph Convolutional Networks (GCN); Graph Attention Networks (GAT); hyperparameter tuning; data augmentation; PEGASUS; synonym replacement; optuna bayesian optimization; node classification; class imbalance*

I. Introduction

Text classification represents a foundational element within the domain of natural language processing (NLP), showcasing a broad scope of applications. Text classification systems must effectively map textual content to appropriate categories, a task complicated by the need to capture both local semantic patterns and global contextual relationships. In conventional methods, text is represented by hand-built features, typically lexical features (e.g., bags of words and n-grams) although they are efficient for particular tasks, semantic complex patterns are not well captured. The development towards deep learning models has been considerably applied to learn advanced textual representations, incorporating convolutional neural networks (CNNs) [1] and recurrent neural networks (RNNs) like long-term memory (LSTM) [2], such models perform well in terms of capturing semantic and syntactic insights within sequences of local consecutive words; however, they are still likely to disregard co-occurrence of the global words in a corpus that covers long-range, non-consecutive semantics dependencies that are important for overall text comprehension [3].

Lately, emerging graph-based approaches trends have gained significant attention to remedy these limitations, such as graph neural networks or graphical embeddings [4]. Unlike conventional sequence-based methods, GNNs allows for processing data in graphical interconnected network format. This paradigm shift enable the models to leverage both local characteristics of nodes and global graph topology, given that graphs are highly eloquent and boost calculation performance and extract semantic and syntactic information, innovative methods emerged for processing graph-structured data in machine learning, i.e., the graphical neural network, making its mark in numerous applications and proving itself as an effective and successful architecture.

Among the various GNN architectures, particularly Graph Convolutional Networks (GCNs) [5], shown remarkable success in node classification tasks. Based on the foundational work of spectral graph theory, the studies of (Bruna et al. 2014) [6], (Henaff et al. 2015) [7], and (Defferrard, Bresson, and Vandergheynst 2016) [8] set up the theoretical basis for carrying out convolution operations on graph-structured datasets, yielding competitive results in node classification seminal contribution introducing a localized, first-order method seminal contribution streamlined these spectral approaches by offering a localized, first-order approximation that maintains computational efficiency while achieving state-of-the-art performance across diverse graph-structured datasets, including the widely adopted Cora benchmark. Further applications include various domains from semantic role labeling in [9] to biomedical entity classification in [10].

Incorporating attention mechanisms into GNNs is yet a further significant advancement. The Graph Attention Network (GAT), introduced by Velickovic et al. (2018) [11], builds upon the concept of attention, thereby enabling the model to prioritize the most significant nodes during the aggregation process. Graph Attention Networks (GATs) have been highly influential in enhancing the performance of GNNs by dynamically assigning adaptive attention weights to neighboring nodes based on their relevance allowing them to allocate computational resources to the richest connections, resulting in more nuanced and effective feature representations, notably in heterogeneous graph settings in which node and edge types diverge considerably.

Although there have been significant architectural advances, critical challenges persist that impede the practical application and scalability of graph-based textual classification models such as data scarcity, class imbalance, and semantic coherence preservation. Benchmark datasets are typically provided in a format that limits their potential for these type of models, Cora [12] an example of a standard dataset, while academically valuable it comes often in pre-vectorized formats, which detaches the learning process from the original semantic complexity, restricts certain feature engineering introducing a barrier in research from flexibility in exploration of different preprocessing pipelines. In addition to its small size and class imbalance which further hinders generalization rare classes are not well represented. Furthermore, the lack of access to the original text prevents any attempt at implementing effective augmentation strategies. These common limitations prevent GNNs from fully using their ability to learn contextual representations from textual content.

Existing approaches exhibit a notable research gap: while important efforts have focused on architectural innovations to advance GNN capabilities, less attention has been paid to the research-based investigation of dataset quality and preparation. This gap can be seen in three practical challenges that limit how well we can evaluate performance. First, benchmark datasets like Cora are distributed in pre-vectorized formats, which limit access to original textual content and opportunities for controlled preprocessing or semantic augmentation. Second, the often-overlooked persistent class imbalance such as Neural Networks composed of 30% of samples while only 6% for Rule Learning can make the evaluation less accurate. Third, semantic preservation metrics for augmented graph data remain inconsistent, this makes it hard to make sure that synthetic relationships are meaningful. These data-related limitations suggest that focusing only on architectural complexity may not solve foundational challenges in graph-based text classification, underscoring the need for complementary data enhancement approaches. Although several studies acknowledge these issues, more careful investigations into semantic-preserving augmentation and class distribution mitigation remain limited. Addressing these challenges is essential to support and not replace architectural progress.

To address this gap, this study explores the following research question: How can semantic-preserving dataset augmentation serve as a cost-effective complement to architectural innovations in graph-based text classification, while maintaining semantic consistency and boosting performance without increasing computational demands? This question is motivated by the practical need to make advanced graph learning accessible to resource-constrained environments while tackling fundamental data quality issues that architectural solutions alone cannot resolve. To answer this question, we present a scalable framework for dataset augmentation that preserves semantic consistency, aimed at improving graph-based text classification using graph neural networks (GNNs). Our work places particular emphasis on Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs). The framework responds to a frequent issue in this field—the discrepancy between the sophistication of theoretical models and the constraints placed on by real-world datasets. While much of the existing literature centers on architectural innovation, our findings suggest that focusing on data quality—specifically through targeted augmentation—can lead to performance gains that are comparable to those achieved through complex model redesigns. Our approach addresses several challenges:

1) *Cost-effective:* alternative to developing new architecture, making advanced graph learning accessible to researchers with limited computational resources.

2) *Data augmentation:* We introduce a hybrid augmentation pipeline that combines in a refined pipeline, NLP-based techniques like PEGASUS for text paraphrase [13] , controlled synonym replacement techniques [14], and a synthetic domain-specific text generation, we augment the dataset to enhance both the diversity and quantity of the training data and solve class imbalances while preserving semantic coherence while enabling in addition flexible and transparent text preprocessing.

3) *Semantic quality assurance:* We use semantic similarity checks employing the metric Word Mover's Distance (WMD) [15] to confirm that augmented samples maintain meaning and contextual relevance rather than depending just on cosine similarity operating on fixed vector representations and may overlook semantic changes introduced by paraphrasing.

4) *Model optimization and evaluation:* we use Bayesian-optimized Optuna [16] hyperparameter tuning, and k-fold cross-validation [17] to guarantee strong evaluation of model performance, providing a more reliable assessment of generalization across different data subsets. This method enhances the reliability of our systematically fair comparison of GCNs and GATs, in order to guarantee that the models reach their maximum potential on the augmented dataset.

Our approach yields GCN achieving an accuracy of 95.42% and GAT achieving 93.46% on 6.57x larger Cora dataset while supporting semantic fidelity (WMD scores: 0.27-0.34) showing the effectiveness of models with our flexible pipeline.

This paper is organized as follows: Section II reviews related work on GNNs and recent advancements in node classification. Section III details our methodologies. Section IV presents experimental results and analysis. Finally, Section V concludes with limitations and future research directions.

II. Related Works

Recent advancements in Graph Neural Networks (GNNs) have propelled the field of graph-based node classification forward in addressing various limitations. Key developments include:

1) Spectral graph convolutions: The groundwork for spectral graph convolutions was established by Bruna et al. (2014) [6], which was further advanced by Defferrard et al. (2016) [8] through ChebNet, utilizing Chebyshev polynomials to approximate convolutions on graphs. Kipf and Welling (2017) [5] simplified these methods by introducing Graph Convolutional Networks (GCNs), achieving state-of-the-art performance while maintaining computational efficiency.

2) Attention mechanisms in GNNs: Veličković et al. (2018) [11] introduced Graph Attention Networks (GATs), employing self-attention mechanisms to dynamically weigh the significance of neighboring nodes, facilitating more flexible feature aggregation. This model has shown particular efficacy in heterogeneous graph settings, enhancing model adaptability.

3) Scalability and efficiency: Addressing scalability issues, Chen et al. (2018) [18] proposed FastGCN, which employs importance sampling to minimize computational complexity. Huang et al. (2018) [19] further refined this with Adaptive Sampling GCN (AS-GCN), improving performance for large-scale graphs.

4) Handling heterogeneity: The Heterogeneous Graph Attention Network (HAN), proposed by Wang et al. (2019) [20], tackles challenges in heterogeneous graph data, demonstrating notable improvements in tasks like author identification and paper classification.

5) Mitigating over-smoothing: The issue of over-smoothing in deeper GNN architectures was addressed in study [21] with DeepGCNs, which integrated residual connections and dilated convolutions, facilitating the training of much deeper networks.

6) Data augmentation for graphs: Zhao et al. (2021) [22] proposed GraphSMOTE, a modification of the SMOTE algorithm tailored for graph-structured data to tackle class imbalance. Ding et al. (2022) [23] introduced a comprehensive data augmentation framework specifically for GNNs, showing significant improvements in performance across multiple tasks. Additionally, Zhang et al. (2020) [13] highlighted the effectiveness of PEGASUS for paraphrasing and synonym replacement in augmenting text data for GNN applications.

7) Hyperparameter optimization: Recent work by (Akiba et al., 2019) [16] emphasized automated hyperparameter tuning for GNNs, achieving notable enhancements in model performance across various datasets. R. Kohavi (1995) [17] further explored the application of k-fold cross-validation, emphasizing robust model evaluation practices necessary for ensuring generalizability in GNN contexts.

To overcome previously cited limitations and build upon these architectural advances. Our work focuses on combining architectural strength with data-centric approaches.

III. Methodology

A. Graph Convolutional Networks

GCN is a fundamental component of our approach to text classification using graph-based data. This section outlines the GCN architecture, explains the mathematical foundations, and highlights its role in our methodology.

1) Architecture overview: The GCN [1] operates like a layer that receives a set of input vectors representing the nodes, in conjunction with the graph structure, producing a new set of node mappings. In the context of a directed graph, denoted:

$$G = (V, E) \tag{1}$$

where V represents the set of vertices (nodes), while E the set of edges (see Fig. 1).

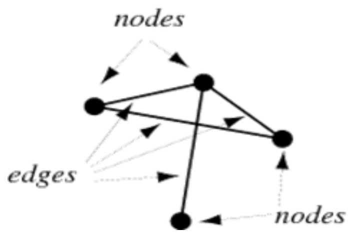

Fig. 1. Structure of a directed graph illustrating node-edge relationships in Cora dataset.

The purpose is to glean insights from the graph by the features function. This function uses as its input:

A description of the features xi for each node I, summed in a feature matrix N × D X (N: nodes, D: input features).

A graph structure description embodied in a matrix form; specifically, as an adjacency matrix A (or a function of A).

The output Z (an N × F matrix, F: output features at each node). Every layer unfurls as a unique non-linear function:

$$H^{(l+1)} = f(H^{(l)}, A) \tag{2}$$

H(0) initialized as X , as for I: the layers number. The patterns vary solely in choosing and parameterizing the function, so it can capture the transformation performed on the node representations in each layer, and A is a matrix attained through the normalization process applied to the adjacency matrix of the graph G.

2) Message passing: As its name suggests, it refers to the neighbor of the destination node. The general concept is to exchange message (insights) constantly with its neighbors to reach a steady balance. Fig. 2 exhibits an illustration of the message passing process that consists of two key steps:

a) Aggregation: each node transmits feature information (messages) to the target node from its neighbors.

b) Update: the functionality of each node according to the "Message" received to form an embedded representation (Embedded).

Message passing guarantees consistent dimensions for all representations, allowing easy further processing. A final integrated representation is achieved after many layers of message passing, capturing the complex relationships within the graph structure. Eq. (2) gives the rule for passing messages from a GCN to a layer for an undirected graph, G.

$$H = \sigma\,(AXW) \tag{3}$$

H represents the updated node representations, X the matrix that wraps the characteristics of the nodes, where W embodies weight parameters, and σ functions as the non-linear activation coordinator. Meanwhile, A appears as a matrix built using a process of normalization of the rows of the adjacency matrix of the complex graph to add richer feature representation of the model. The following Fig. 2 exhibits the message passing mechanism throw layers.

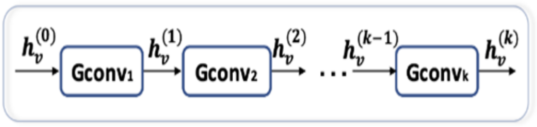

Fig. 2. Message passing mechanism.

In summary individual nodes aggregate information from their interconnected nodes, applying an aggregation function, which ensures that all representations result in equal size. Next, the vector undergoes a transformation through a dense neural network layer, together with a non-linear activation function to refine the vector's mapping. These steps are repeated according to how many layers are in the model. In particular, GCNs feature distinct representations at each layer:

At the zeroth layer, the representation is aligned with the node's specific characteristics.

At k layer, the node's mapping is calculated by going through its neighbors, aggregating their mappings from the preceding layer (k-1), averaging them and subjecting the result to a transformation via a parameter matrix. This process extends to include the node's proprietary messages from k-1 onwards. This value is then subjected to a non-linear function, such as ReLU in our case.

Eventually, when the node's mappings pass through the transformations in the hidden layers conclusive integration is achieved. In summary, GCNs work via iterative message transmission among nodes, allowing them to leverage representations capturing their structural context, both local and global, in the graph.

B. Graph Attention Networks

GATs offer an excellent opportunity for the development of graphical neural networks. With Graph Convolutional Networks (GCN), every neighbor has the same importance. Yet, some nodes are more essential than others. Graph Attention Networks solve this issue with the self-attention mechanism that regards significance of individual neighbors, an attention mechanism granting a weight coefficient per connection. The following section illustrates GAT architecture, provides an overview of its mathematical background, and outlines its contribution to our methodology.

1) Architecture Overview: GAT [11] architecture consists of various key elements, designed to contribute to its performance in graph-based neural networks:

Self-attention mechanism lies at the core of GAT, this mechanism assigns a weighting factor (attention score) to each connection, allowing the nodes to target the most relevant neighbors. GAT applies mathematical equations to aggregate node characteristics based on attention, the embedding of node 1 is calculated as the equation below shows, W: shared weight matrix:

$$h_1 = \alpha_{11} w\varkappa_1 + \alpha_{12} w\varkappa_2 + \alpha_{13} w\varkappa_3 + \alpha_{14} w\varkappa_4 \tag{4}$$

α_{ij} the attention scores across nodes i and j.

The calculation of these attention scores proceeds through three stages:

a) Linear transformation: To calculate the importance of each connection, pairs of hidden vectors are needed (see Fig. 3). A straightforward approach to forming these pairs involves concatenating the vectors of the respective nodes. Moving on from this step, a new linear transformation is applied using trainable attention vector a:

$$a_{ij} = a^t [W\, h_i || \, W h_j] \tag{5}$$

Where:

$W \cdot h_i$ and $W \cdot h_j$ are the transformed node features,

∥ Indicates vector concatenation,

a^t is a transposed attention vector (learned during training).

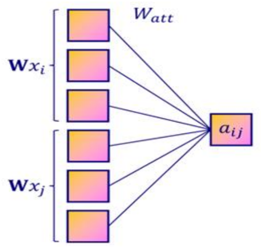

Fig. 3. Linear transformation process.

b) Activation function: Since the aim is to build a neural network, the activation is the second stage. In this context, the LeakyReLU function is added (see Fig. 4).

$$e'_{ij} = LeakyReLu(e_{ij}) \tag{6}$$

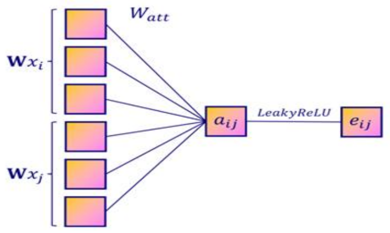

Fig. 4. Computing attention coefficient.

c) SoftMax normalization: The output of the neural network does not undergo normalization. To assess the relative relevance of node 2 to node 1 to node 3 ($\alpha_{12} > \alpha_{13}$), the same scale should be shared. A frequently employed method in neural networks involves the SoftMax function (see Fig. 5). The equation below shows how it is applied to every neighboring node:

$$\alpha_{ij} = softmax(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k \in N_i} exp(e_{ij})} \quad (7)$$

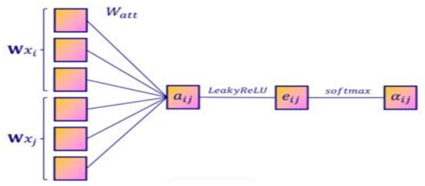

Fig. 5. SoftMax normalization.

We can calculate every α_{ij}. However, self-attention can exhibit instabilities. To enhance robustness, multi-head attention was introduced into the architecture of the transformer.

d) Multi-head attention: The multi-head attention (see Fig. 6) involves repeating three identical steps repeatedly, with the aim of calculating the average or recombine outputs. Here, rather than obtaining a unique hidden h_1, a separate hidden vector h_1^k is generated for each attention head. We can then apply one of two schemes:

Average: add together each of the h_i^k. Normalize via the total number n of heads of attention.

$$h_i = \frac{1}{n} \sum_{k=1}^{n} h_i^k \quad (8)$$

Concatenation: concatenate the different h_i^k.

$$h_i = ||_{k=1}^{n} h_i^k \quad (9)$$

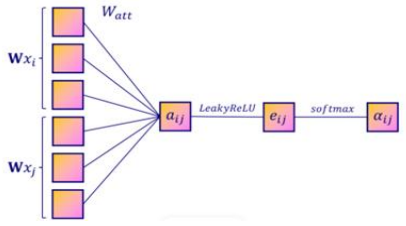

Fig. 6. Multi-head attention.

In summary, attention-graph networks define self-attention as a mechanism that allows nodes nearby to be accorded levels of importance. Mathematically, this self-attention is characterized by linear transformations, activation functions, and SoftMax normalization. Multi-headed attention adds to the model's performance by considering multiple perspectives during the aggregation process.

C. Data Preprocessing and Augmentation

1) Original Cora dataset description: The Cora dataset published by Andrew McCallum and his research group at the University of Massachusetts Amherst [12] (see Fig. 7) as a compressed zip file containing .ps files containing metadata, abstracts and citations for 2,708 academic publications distributed across seven classes. The accompanying citation grid comprises 5,429 links. Cora's established benchmark status in graph neural network literature enables meaningful performance contextualization, while its citation network structure provides the graph topology essential for GCN and GAT evaluation. We selected Cora for three reasons aligned with our objectives:

- Availability of original PostScript text to support sophisticated, meaning-preserving augmentation.
- Moderate size (2 708 nodes) that allows comprehensive k-fold validation without prohibitive compute demands.
- Pronounced class imbalance (Neural Networks: 30 %, Rule Learning: 6 %), making it an ideal benchmark for evaluating imbalance-aware graph learning strategies.

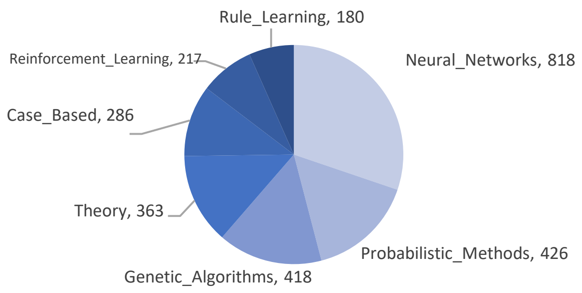

Fig. 7. Original Cora dataset classes distribution.

This dataset was subjected to a comprehensive preprocessing and augmentation protocol to enhance its utility and mitigate inherent limitations and was systematically expanded to a more robust dataset of 17,780 samples through a multifaceted augmentation strategy.

e) Semantic-preserving multi-modal augmentation framework: Theoretical motivation: data expansion traditional methods regularly introduce semantic leakage in which synthetic input samples deviate increasingly far from the domain's original pattern. This issue is also pronounced in datasets like Cora that are already heavily pre-processed to a simplified representation removing much of its contextual richness that GAT and GCN are meant to capture. To solve this problem, instead of developing a costly new augmentation algorithm we propose a combination of existing methods with a semantic control level mechanism following this pipeline:

- Preprocessing Cora original raw dataset: The initial phase was data preprocessing. It involved a meticulous examination and refinement of the raw Cora dataset, originally distributed in PostScript (ps) file format. This process was executed through a series of sophisticated steps to preserve data quality, relevance and semantic integrity.

Content Extraction and Metadata Removal: we isolate essential textual content directly from the PostScript file one by one using a custom rule-based string parser we implemented in

Python while systematically removing extraneous metadata, such as (e.g. author names and institutional affiliations, links, etc.). Abstracts were retained, as they serve as the primary textual representation of each paper.

Text Cleaning: A multi-tiered protocol to text cleaning has been used taking into consideration benchmark practices to not roughly eliminate semantic senses: custom regular expressions were employed to remove non-alphanumeric characters, normalize whitespace, and standardize formatting.

- Data augmentation Pipeline (Methodologies)

$$\text{Let } D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\} \quad (10)$$

D represent the original dataset, where x_i is the feature vector and y_i is the label that correspond to the feature in question. The generated D' is the result of the augmentation process such that $|D'| > |D|$ while preserving the label condition distribution $P(x', y') \approx P(x,y)$.

PEGASUS-based Textual Transformation: We employed the PEGASUS (Pre-training with Extracted Gap-sentences for Abstractive Summarization) model, specifically fine-tuned for paraphrasing [13], as the primary method for generating diverse textual variations. This approach allows us to create semantically equivalent but linguistically diverse versions of the original text:

$$P = f_{\theta_p}(x) \quad (11)$$

where P is the paraphrased text, x is the paper textual content, and f_(θ_p)

is the PEGASUS paraphrasing function with parameters. This technique effectively doubles the dataset size while introducing linguistic variations that can improve model robustness.

Subject-Aware Lexical Substitution via Synonym Replacement: We used synonym replacement algorithm [13] while enforcing controlled lexical variations through filtering according to subject domain, thereby maintaining thematic coherence while expanding vocabulary:

$$x_{\text{aug}} = h(x_{\text{orig}}, \lambda) \quad (12)$$

Where x_aug is the augmented paper content, x_orig is the original or previously augmented content, h is the synonym replacement function, and λ is a parameter controlling the degree of replacement.

Conditional restrictions are applied so that only substitutions that met semantic and label constraint were retained (Samples exceeding the WMD threshold were rejected)

$$\text{WMD}(x_{\text{orig}}, x_{\text{aug}}) \leq \tau_{\text{wmd}} = 0.35 \quad (13)$$

Class-balanced custom Generation Pattern**:** we devised a sophisticated syntactic generation framework characterized by a meticulously curated lexicon of action-oriented verbs and descriptive adjectives. Enabling the formulation of new sentences that are both semantically rich and consistent with the underlying class distributions of the original text, this pattern was applied specifically to limited set of isolated nodes with no connection in the network and particularly those with the shortest text content that lack sufficient presentation.

$$N = g(X, V) \quad (14)$$

where N is the new synthetic sample, X is the original text, V encapsulates our curated vocabulary and advanced terminological construct, and g is our custom generation function that generates suitable terms from V while respect part-of-speech to maintain correctness. This combinatorial approach allows for controlled growth of even smaller contributor nodes in the dataset with semantic check through the threshold τ_{wmd}.

- Parameter Sensitivity: The choice of crucial parameters in our augmentation pipeline has an important impact on performance. While a lower Learning rate (Lr) produces nearly duplicate paraphrases with little advantage, a higher PEGASUS (Lr) introduces semantic drift that reduces classification accuracy. Raising the synonym rate beyond its optimal point disrupts domain-specific terminology, damaging coherence, while too low a rate limits new lexical variety. Temperature values above 0.8 produce semantically inconsistent text that misleads the classifier, and values below 0.6 result in repetitive, uninformative samples. While lowering the WMD threshold allows for excessive semantic divergence, raising it over-filters advantageous variations. These findings underscore how sensitive the augmentation process is to parameter choices and highlight the importance of systematic tuning, as even minor deviations from optimal settings can significantly degrade semantic quality.

- Methodological Validation: Semantic Preservation**:** For each newly augmented node inherits the original node's edges to preserve citation relationships so that the structural semantics of the graph remain intact after augmentation. To support the integrity of the augmented dataset, we implement: Automated intra-category semantic similarity checks between original and augmented document embeddings for each category. We computed the Word Mover's Distance (WMD) [15] and set threshold for strict filtering; this validation mechanism prevents the semantic drift commonly seen in synthetic data generation. In addition to analyzing repetitiveness in sentence structure and phrase.

Class Balance: The data were augmented to realize equal class distributions (approximately 14% each) drops the bias in the original Cora, where Neural Networks represented 30% of samples while Rule Learning only 6%, Neural Networks now present more equitable 16% reduced dominance. This balance guarantees fair model evaluation.

Computational Efficiency: compared to the improvement of ensemble methods or attention mechanism. This pipeline provides better graph learning preserved semantically, enabling models to achieve better performance without architectural modifications, reducing computational overhead and more accessible to any individual.

IV. Experimental Results

A. Dataset Description

In this study, we have created a customized augmented dataset from Cora [12] using various methods, each with critical parameter settings. The Pegasus model utilized a learning rate of 0.01 selected following several tests and monitoring continuously steady convergence without overshooting, a maximum input length of 512 tokens, a beam search size of 5, and was trained for 5 epochs to ensure effective learning. For synonym replacement, we set a replacement probability of 0.3 and a maximum of 3 synonyms per word, maintaining semantic integrity while promoting variability. In the text generation phase, we used a temperature of 0.7 for balanced creativity, top-k sampling with k=50 to restrict predictions to the most probable words, and to reduce repetitive phrases we used a repetition penalty of 1.2. These carefully chosen parameters, and many others significantly preserved the quality and diversity of the generated data.

The New customized Cora dataset (see Fig. 8) cumulative generated by the augmentation techniques resulted in the important expansion of our dataset from the original 2,708 samples to 17,780 samples, an increase in the factor of approximately 6,57x. This expansion not only increased the training data volume but also preserved the diversity of linguistic variations and synthetic samples, all while maintaining semantic coherence and domain.

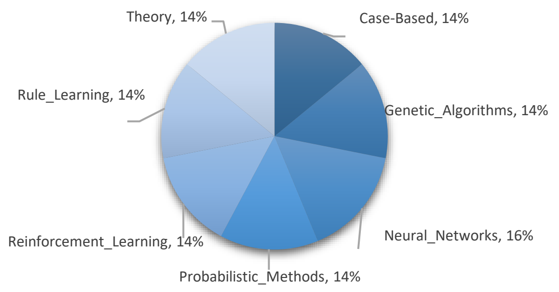

Fig. 8. Customized dataset classes distribution.

The quality and relevance of the augmented samples shows a similarity threshold of 0,8 was maintained. Additionally, WMD calculated scores intra classes (see Table I) values around 0.3 to 0.34 suggest moderate similarity between original and augmented texts within each category introducing meaningful variation.

TABLE I. WMD Scores

Category	WMD scores
Case-Based	0.3389
Genetic Algorithms	0.3049
Probabilistic Methods	0.3044
Reinforcement Learning	0.3309
Rule Learning	0.3283
Theory	0.2773

B. Hyperparameter Tuning Configuration

We adopted to conduct Naive Bayesian hyperparameters tuning with adopted Optuna [16], because of its empirical ability to efficiently navigate high-dimensional search spaces and typically generalize better, even in contexts like ours involving data imbalance and augmentation where stability is critical. The full set of optimized hyperparameter configurations for both GCN and GAT models is summarized in Table II, which lists ranges for learning rates, dropout, and architectural elements.

This probabilistic approach enabled us to model the relationship between hyperparameters and model performance. In addition to ensuring the robustness and generalizability of our hyperparameter configurations, we employed a k-fold cross-validation strategy. This approach partitioned our augmented dataset into k subsets, allowing for multiple training and validation cycles. The cross-validation procedure not only provided a more reliable estimate of model performance but also mitigated the risk of overfitting to specific data partition.

TABLE II. Optimized Hyperparameter Configurations

Elements	Values	Model
Optimization framework	100-400 trials	GCN-GAT
Hidden features	8 to 128	
Activation	["relu", "relu"], ["relu", "softmax"], ["elu", "softmax"]	
Dropout	0.1 to 0.8	
Learning rate	1e-5 to 1e-1 (Log-uniform)	
Weight decay	1e-6 to 1e-2 (Log-uniform)	
Cross-Validation	5-fold cross-validation	
Early Stopping	Patience of 20 epochs, monitored on validation set's F1 score	
Attn_heads	1 to 16	GAT
attn_dropout	0.2 to 0.8	

The hyperparameter tuning process aimed to maximize the mean cross-validated F1 score. The best-performing hyperparameters for each model were selected based on this metric. We then compared the optimized GAT and GCN models to determine which architecture was more suitable for our specific graph classification task.

C. Evaluation Metrics

Classification models are evaluated using well-established metrics to quantify their performance and effectiveness. These metrics are essential in determining how well a model can predict outcomes based on the given data. Below are the key metrics used in evaluating GCN and GAT classification [5] [11]:

True Positives (TP): Instances where the model correctly predicts positive outcomes (i.e., both the actual and predicted results are positive).

True Negatives (TN): Instances where the model correctly predicts negative outcomes (i.e., both the actual and predicted results are negative).

False Positives (FP): Occurrences where the model predicts a positive result, but the actual result is negative (also known as a Type I error).

False Negatives (FN): Occurrences where the model predicts a negative result, but the actual result is positive (also known as a Type II error).

Accuracy: Conversely, false negatives occur if the model predicted a negative result, while the real result proved positive**.**

$$\text{Accuracy} = \frac{\text{TP}_\text{i}+\text{TN}_\text{i}}{\text{TP}_\text{i}+\text{TN}_\text{i}+\text{FP}_\text{i}+\text{FN}_\text{i}} \tag{15}$$

Loss: Loss measures the discrepancy between predicted values and actual values. Cross-entropy is a common loss function in deep neural networks. It quantifies how well the predicted probabilities match the actual class labels.

$$\textbf{C}\text{rossEntropy} = -\sum_{i=1}^{n} log\left(P_{i,i}\right)y_{i,j} \tag{16}$$

where yi,j is the true value, i.e. 1 if sample I is in class j and 0 otherwise. Pi,j the likelihood forecast via the model that sample I is part of class j.

Precision: is the ratio between the True Positives and all the Positives.

$$\text{Precision} = \frac{\text{TP}_\text{i}}{\text{TP}_\text{i}+\text{FP}_\text{I}} \tag{17}$$

Recall: the extent to which our model correctly identifies true positives.

$$\text{Recall} = \frac{\text{TP}_\text{i}}{\text{TP}_\text{i}+\text{FN}_\text{I}} \tag{18}$$

F1 Score is the harmonic mean of Precision and Recall. It provides a balanced measure that considers both false positives and false negatives, making it particularly useful when dealing with imbalanced datasets. The F1 Score ranges from 0 to 1, with 1 indicating perfect precision and recall.

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{19}$$

D. *Results and Discussion*

1) Model settings: In this experiment we adapted across all models the same following configurations:

a) Data preparation: The textual preprocessed data was transformed into TF-IDF vectors, with a maximum of 1000 features for each record. Class labels were encoded using LabelEncoder, ensuring numerical representation for classification. We created an adjacency matrix from citation data, representing relationships between papers, and normalized it to ensure proper scaling in the graph model.

Stratified Data Splitting We employed stratified k-fold cross-validation with 5 splits to ensure balanced class distributions across training and validation sets. A random state with a fixed seed of 42 was used for reproducibility throughout the data splitting and model training. For each fold, data was split into training and validation sets, without using a k-fold throught cross-validation.

Following comprehensive hyperparameter tuning and testing, we identified a set of configurations for the foundational models as detailed in Table III. These configurations were meticulously selected to enhance the performance of the respective models.

The results yielded by the individual models, GCN and GAT, were rigorously analyzed. Fig. 9 depicts the outcomes of numerous testing and fine-tuning iterations, providing a summary of the average performance metrics from the 5 folds that are broken down per model, and summarized in Table IV including key metrics: validation accuracy, precision, recall, and F1 score.

The results of our analysis reveal that GCN is consistently better than GAT across all evaluated metrics. Specifically, GCN achieves a validation accuracy of 95.42%, which is a higher overall correct classification rate than GAT's 93.46%. In terms of precision, GCN achieves a score of 95.45%, compared to GAT's 93.50%, suggesting a lower false positive rate for GCN. In addition, GCN's recall of 95.42% above GAT's 93.46%. This shows that GCN's improved capability in identifying true positives. The F1 score shows GCN at 95.42%, while GAT scores 93.47% which reflects the balance between precision and recall.

TABLE III. GRAPH CONVOLUTIONAL NETWORK AND GRAPH ATTENTION NETWORK SETTINGS

Model	**GCN Instance Model**	**GCN Average Model**	**GAT Instance Model**	**GAT Average Model**
Model architecture	layer_sizes =108 activated by ReLU	layer_sizes = 116 activated by ReLU	layer_sizes = 106 activated by ELU	layer_sizes = 106 activated by ELU
Epochs	200	200	200	200
Optimizer	Adam	Adam	Adam	Adam
Learning Rate	0.04401160472	0.06507184668	0.00066108710 step_size=50, gamma=0.5	0.000661087 Step_size=50, gamma=0.5
Early Stopping	Stagnant F1 score validation for 200 epochs	Stagnant F1 score validation for 200 epochs	Stagnant F1 score validation for 200 epochs	Stagnant F1 score validation for 200 epochs
Patience	20	20	20	20
Activation	ELU (hidden layers), Log Softmax (output)	ELU (hidden layers), Log Softmax (output)	ELU (hidden layers), Log Softmax (output)	ELU (hidden layers), Log Softmax (output)
Dropout Rate	0.44934028857 (45%)	0.53708280317 (50%)	0.29109068673 (30%)	0.2910906867 (30%)
Weight Decay	3.602702935945202e-05	3.602702935945202e-05	0.00010625787460334993	0.00010625787460334993
Attention Heads	None	None	16	16

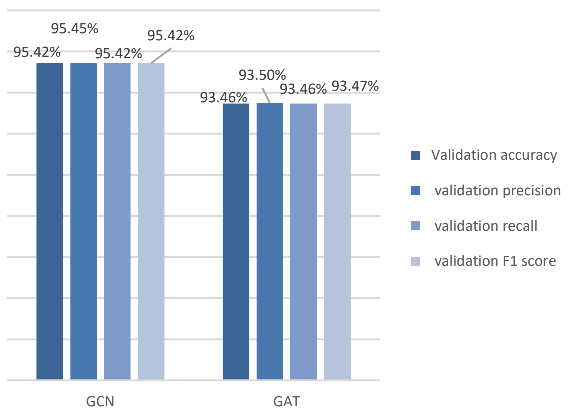

Fig. 9. Performance metrics comparison.

The GCN model performed better than the GAT model across all evaluation measures, with accuracy being approximately 1.96% higher (95.42% vs. 93.46%) and statistical significance with the paired t-tests showing ($p < 0.05$, $t = 2.89$, $df = 4$) also shows a 95% confidence interval: (94.77%, 96.07%) vs. (92.83%, 94.08%). The strong performance of both models, with low standard deviations, indicating stable learning over different data distributions.

TABLE IV. COMPARATIVE PERFORMANCE ANALYSIS OF GCN AND GAT MODELS ACROSS KEY EVALUATION METRICS

Model	Folds	Validation accuracy	Validation precision	Validation recall	Validation F1 score
GCN	F1	95,05%	95,06%	95,05%	95,04%
	F2	96,26%	96,27%	96,26%	96,26%
	F3	95,02%	95,09%	95,02%	95,03%
	F4	95,61%	95,61%	95,61%	95,60%
	F5	95,16%	95,23%	95,16%	95,18%
GAT	F1	92,69%	92,73%	92,69%	92,70%
	F2	93,64%	93,69%	93,64%	93,65%
	F3	93,25%	93,30%	93,25%	93,27%
	F4	93,98%	94,01%	93,98%	93,99%
	F5	93,73%	93,75%	93,73%	93,74%

Based on cross-validation findings, detailed performance showed that the GCN maintains an accuracy of 95.42% (±0.47%) with a precision of 95.45% (±0.45%), a recall of 95.42% (±0.47%) and an F1 score of 95.42% (±0.47%). While GAT also manifests the corresponding performance levels of 93.46% (±0.45%), 93.50% (±0.45%), 93.46% (±0.45%) and 93.47% (±0.45%). We kept track of validation loss across folds to determine model convergence and confidence level were balanced. GCN showed a consistently lower average validation loss of ~0.18 compared to GAT's ~0.21. Both models demonstrated small standard deviations, indicating a good level of stability as well as reproducibility across different data partitions.

However, there is a variation between the study datasets and data refinement methodologies, confronted with these limitations the establishment of internal reference measures was undertaken. To this end, we trained GCN and GAT on the original Cora dataset (2,708 samples) using identical architecture but without augmentation. The resulting accuracy rates were 83.39% and 82.24%, respectively, aligning with existing literature on the subject such as [5][11], where standard Cora benchmarks rarely go above 85% in the absence of architectural refinement. While this comparison does not represent a strictly controlled ablation, it is still useful to understand the potential impact of our semantic preservation, class balanced augmentation pipeline. The augmented and optimized models exhibited an accuracy rate of over 95%, these results show how important it is to preserve the semantics of the dataset during the engineering process. The findings of this study provide an indicative—but not strictly controlled—comparison, which is developed in the limitations section.

- Class-wise Performance Analysis: To further investigate per-class performance, we computed F1-scores for each class across all folds (see Table V).

TABLE V. AVERAGED F1-SCORES RESULTS ACROSS THE FIVE FOLDS FOR EACH CLASS WITH STANDARDS DEVIATIONS

Class	*GCN F1-Score*	*GAT F1-Score*	*(GCN-GAT)*	*Support*
Case-Based	0.9575 ± 0.0073	0.9378 ± 0.0029	+0.0197	500
Genetic_Algorithms	0.9726 ± 0.0037	0.9584 ± 0.0039	+0.0142	500
Neural_Networks	0.9194 ± 0.0097	0.8898 ± 0.0115	+0.0296	556
Probabilistic_Methods	0.9504 ± 0.0081	0.9284 ± 0.0054	+0.0220	500
Reinforcement_Learning	0.9733 ± 0.0057	0.9608 ± 0.0135	+0.0125	500
Rule_Learning	0.9737 ± 0.0068	0.9590 ± 0.0081	+0.0147	500
Theory	0.9364 ± 0.0096	0.9140 ± 0.0086	+0.0224	500
Weighted Avg	0.9542 ± 0.0047	0.9347 ± 0.0045	+0.0195	3556

As detailed in Table V the analyses by class highlight notable aspects. Overall, all seven classes are consistently well managed by the GCN model, with a performance variance between 1.25% (Reinforcement Learning) and 2.96% (Neural Networks). The Neural Networks category shows the most difficulties in classification with both models, particularly due to the conceptual overlap and semantic ambiguity inherent in the literature on neural networks affecting many areas of artificial intelligence. Probabilistic Methods and Theory exhibit relatively significant performance differences (2.20% and 2.24%), suggesting that GCN's aggregation process is effective in capturing the particularities of these domains. However, the relatively smaller difference between the two models in reinforcement learning (1.25%) clearly illustrates how well the two models deal with this specific domain.

The plots shown in Fig. 10 and Fig. 11 provide an overview of the learning history that gives a further indication of the convergence and stability of GCN and GAT performance over the course of the learning process with second fold as the representative fold. We can observe that the learning process for both models GAT and GCN presents some notable differences, particularly during the early training stages. GAT, for instance, is more unstable at the beginning, especially in the initial starting epochs. This instability likely comes from the way attention mechanism that GAT uses, as it requires more time to adjust and optimize the importance of edges within the graph. Learning how to prioritize different node connections adds more complexity, which in turn causes performance to initially fluctuate.

How GAT manages isolated nodes is one of its main challenges. While GCN works best in treating all neighboring nodes equally, GAT's effectiveness is more dependent on the topology of the graph. The presence of isolated nodes, which lack neighboring information, makes it harder for GAT to fully benefit from its attention mechanism. This has likely contributed to the early performance issues and slower convergence compared to GCN.

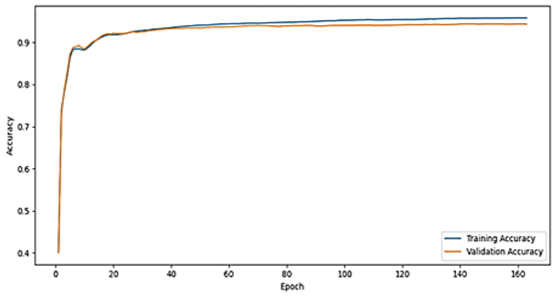

Fig. 10. GAT Fold 2 training and validation accuracy fit history.

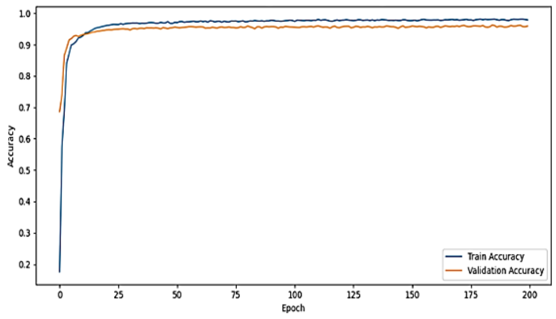

Fig. 11. GCN Fold 2 training and validation accuracy fit history.

Despite these challenges at the beginning, both models eventually ended up achieving high-performance levels. While it takes longer to converge for GAT, that frequently matches or even outperforms GCN in the end. This suggests that although GAT works in a more complex optimization environment, its ability to learn which edges are most important can offer important advantages as training continues.

Overall, the consistent results across the five cross-validation folds show that both models are robust, especially GCN, which was less variable across different data splits. The slight differences in GAT's performance could be due to its sensitivity to specific graph structures within each fold. Compared to baseline models, both GCN and GAT perform very well on the expanded dataset, suggesting their architecture's strength in handling complex graph data. This analysis underlines the unique strengths of each model and the importance of giving GAT sufficient training time to fully optimize its performance, particularly in scenarios with diverse graph connectivity.

V. Conclusion

This study represents a notable advancement in graph-based node classification, effectively addressing key challenges related to data quality in preserving semantic aspect, class imbalance, and model performance generalization constraints. Our approach enhanced the Cora dataset by extracting content from its original PostScript files and applying a systematic implementation of a tripartite augmentation framework combination of benchmark techniques: a combination of textual transformation PEGASUS-based, subject aware synonym replacement and customized class-balanced generation pattern function on isolated nodes to expand the dataset to 17,780 nodes, achieving a 6.57x scaling factor while maintaining semantic fidelity as evidenced by Word Mover's Distance scores ranging from 0.27 to 0.34. In addition the use of Optuna-driven hyperparameter optimization using Naïve Bayesian with k-fold cross-validation, to make sure achieving optimal model performance. Both the GCN and GAT models performed effectively, achieving accuracy of 95.42% and 93.46% respectively. This analysis highlights how is consistent is GCN alongside GAT more intricate learning pattern, where early fluctuation gave way to competitive results. These results show the potential of attention mechanisms to capture complex graph dynamics with sufficient training. The models' resilience, validated across five folds, further demonstrates motivation of our approach. Importantly, this work illustrates how advanced data augmentation and optimization techniques can boost the performance and scalability of graph-based models by preserving important semantic aspect that can be lost in rough engineering, also shows the ability of these models on larger datasets without compromising accuracy.

Despite overall good classification performance, there are some limitations that should be acknowledged, particularly in distinguishing between overlapping classes such as neural networks. The attention mechanism of GAT also introduced computational costs without corresponding gains. In conclusion, results demonstrate that performance obtains equivalent to architectural complexity alone can be obtained through strategic dataset augmentation, this principle provides a cost-effective alternative, making advanced graph neural networks accessible to researchers with limited computational resources.

A. Limitations and Future Work

While our approach has shown good results as discussed earlier in the previous section, we acknowledge different limitations that present opportunities for future improving:

1) *Domain specificity:* Our validation focuses on academic paper classification that the Cora dataset provides. Future work should explore the practical applicability across variety of text domains and languages.

2) *Comparative evaluation:* Direct comparison with existing methods is challenging due to the characteristics of the new dataset and the difference in engineering approaches. Future work should establish a more standardized evaluation protocol.

References

[1] Y. Kim, "Convolutional neural networks for sentence classification," arXiv preprint arXiv:1408.5882, 2014.

[2] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, and R. Ward, "Semantic modelling with long-short-term memory for information retrieval," arXiv preprint arXiv:1412.6629, 2014.

[3] H. Peng, J. Li, Y. He, Y. Liu, M. Bao, L. Wang, and Q. Yang, "Large-scale hierarchical text classification with recursively regularized deep graph-cnn," in Proc. 2018 World Wide Web Conf., 2018, pp. 1063-1072.

[4] H. Cai, V. W. Zheng, and K. C. C. Chang, "A comprehensive survey of graph embedding: problems, techniques, and applications," IEEE Trans. Knowl. Data Eng., vol. 30, no. 9, pp. 1616-1637, Sept. 2018.

[5] T. N. Kipf and M. Welling, "Semi-supervised classification with Graph Convolutional Networks," arXiv preprint arXiv:1609.02907, 2017.

[6] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," arXiv preprint arXiv:1312.6203, 2014.

[7] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," arXiv preprint arXiv:1506.05163, 2015.

[8] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in Advances in Neural Information Processing Systems, vol. 29, 2016.

[9] D. Marcheggiani and I. Titov, "Encoding sentences with graph convolutional networks for semantic role labeling," arXiv preprint arXiv:1703.04826, 2017.

[10] Y. Li, R. Jin, and Y. Luo, "Classifying relations in clinical narratives using segment graph convolutional and recurrent neural networks (Seg-GCRNs)," J. Amer. Med. Inform. Assoc., vol. 26, no. 3, pp. 262-268, 2019.

[11] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," arXiv preprint arXiv:1710.10903, 2018.

[12] McCallum, A., Rennie, J., & Nigam, K. (2000). Cora Research Paper Classification Dataset. University of Massachusetts Amherst. Available at: https://people.cs.umass.edu/~mccallum/data/cora-classify.tar.gz

[13] J. Zhang, Y. Zhao, M. Saleh, and J. Liu, "PEGASUS: pre-training with extracted gap-sentences for abstractive summarization," in Proc. 37th Int. Conf. Machine Learning (ICML), 2020.

[14] J. Wei and K. Zou, “EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks,” in Proc. EMNLP IJCNLP, Hong Kong, Nov. 2019, pp. 6382–6388.

[15] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger, “From Word Embeddings to Document Distances,” in Proc. 32nd Int. Conf. Machine Learning (ICML), 2015, pp. 957–966.

[16] T. Akiba, S. Sano, T. Yanase, T. Ohta, and Z. Kira, "Optuna: a next-generation hyperparameter optimization framework," in Proc. 25th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining, 2019.

[17] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," Proc. 14th Int. Joint Conf. Artificial Intelligence (IJCAI), vol. 2, pp. 1137–1143, 1995.

[18] J. Chen, T. Ma, and C. Xiao, “FastGCN: Fast learning with graph convolutional networks via importance sampling,” arXiv preprint arXiv:1801.10247, 2018.

[19] M. Yoon Huang T. Zhang, Y. Rong, J. Huang "Performance-Adaptive sampling towards fast and accurate graph neural networks," in Proc. 27th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining (KDD), 2021, pp. 2046–2056.

[20] X. Wang, H. Ji, C. Shi, B. Wang, P. Cui, P. Yu, Y. Ye, "Heterogeneous graph attention network," in Proc. World Wide Web Conf., 2019.

[21] G. Li, M. Müller, G. Qian, C. Delgadillo, A. Abualshour, A. Thabet, B. Ghanem, "DeepGCNs: Making GCNs Go as Deep as CNNs" in Proc. IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, South Korea, Oct. 2019, pp. 9266–9275. DOI: 10.1109/ICCV.2019.00936

[22] T. Zhao, X. Zhang, S. Wang, "GraphSMOTE: imbalanced node classification on graphs with graph neural networks" in *Proc. ACM WSDM*, 2021, pp. 833–841. DOI: 10.1145/3437963.3441720 (arXiv:2103.08826)

[23] K. Ding, Z. Xu, H. Tong, and H. Liu, “Data Augmentation for Deep Graph Learning: A Survey,” arXiv preprint arXiv:2202.08235, 2022.