Proceedings of CONF-MLA 2025 Symposium: Intelligent Systems and Automation: AI Models, IoT, and Robotic Algorithms
DOI: 10.54254/2755-2721/2025.LD28503

Node Classification in Social Networks Using Graph Convolutional Networks and GraphSAGE

Jiayue Sun

Department of Civil Engineering (Bridge), Tongji University, Shanghai, China

834050709@qq.com

Abstract: Graph Neural Networks (GNNs) have emerged as a prominent research paradigm for learning representations from graph-structured data, providing significant advantages in modeling relational dependencies and non-Euclidean structures. In this study, we focus on the node classification problem in social graphs, where the objective is to accurately predict node labels by leveraging both intrinsic node attributes and topological connectivity patterns. To this end, we employ the widely adopted Cora citation network as a benchmark dataset and conduct a systematic evaluation of two representative GNN architectures: Graph Convolutional Networks (GCN) and GraphSAGE. Our experimental design emphasizes comparative performance analysis against conventional machine learning baselines, thereby highlighting the benefits of explicitly encoding neighborhood aggregation mechanisms. The results demonstrate that GCN and GraphSAGE achieve substantial improvements in classification accuracy by effectively capturing multi-hop contextual information, reinforcing the importance of structural learning in social network analysis. Beyond empirical performance, this work also underscores the accessibility of GNNs as methodological tools for social network research, providing insights that can inform both theoretical advancements and practical applications.

Keywords: Graph Neural Networks, Node Classification, Graph Convolutional Network, GraphSAGE, Cora Dataset

1. Introduction

Graph-structured data are ubiquitous in modern applications, ranging from citation networks and e-commerce platforms to social media interactions and biological systems [1]. Unlike traditional Euclidean data, graphs encode complex relational information, making their analysis both challenging and essential. In recent years, Graph Neural Networks (GNNs) have emerged as a dominant paradigm for representation learning on graph-structured data, enabling breakthroughs in tasks such as node classification, link prediction, and community detection [2, 3].

Among these tasks, node classification has received particular attention, as it aims to infer the labels of nodes based on both their features and the structural context provided by neighboring nodes. Social networks are a prime application domain, where accurate node classification can reveal user

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2025.LD28503

interests, detect communities, and support recommendation systems [4, 5]. However, the irregularity and sparsity of graph data present challenges for traditional machine learning approaches, which often fail to capture higher-order dependencies [6].

Graph Convolutional Networks (GCNs) and GraphSAGE are two widely studied GNN variants that have demonstrated strong performance on benchmark datasets such as Cora, Citeseer, and PubMed [7, 8]. GCNs apply spectral or spatial convolutions to aggregate neighborhood information, while GraphSAGE extends this framework by enabling inductive learning through sampling-based aggregation [9]. Their complementary strengths make them attractive candidates for studying node classification in social networks.

Despite these advances, several challenges remain. First, the scalability of GNNs on large-scale social graphs is a non-trivial issue, with ongoing research seeking to balance efficiency and accuracy [10]. Second, heterogeneity in social networks—where nodes and edges may represent diverse entities and relationships—complicates the design of effective GNN models [11]. Third, the interpretability of GNN predictions remains limited, hindering trust and transparency in real-world applications [12]. Addressing these challenges requires systematic evaluation of GNN architectures under realistic social network scenarios.

In this study, we focus on applying GCN and GraphSAGE to the task of node classification in social graphs using the Cora dataset. This dataset, consisting of citation relationships among scientific publications, provides a well-established benchmark for evaluating GNN models. By comparing these two representative architectures, we aim to highlight both the effectiveness and limitations of current GNN methods, while demonstrating their accessibility for researchers entering the field. Our contributions are threefold:

1. We provide a comparative study of GCN and GraphSAGE for node classification on the Cora dataset, emphasizing their applicability to social network data.
2. We analyze the trade-offs between accuracy, scalability, and interpretability, offering insights into model selection for real-world applications.
3. We underscore the accessibility of GNNs as an emerging research direction, bridging theoretical advances with practical applications in social network analysis.

The remainder of this paper is structured as follows. Section 2. provides a comprehensive review of related studies on Graph Neural Networks (GNNs) within the context of social networks. Section 3. introduces the methodological framework, detailing the architectures of Graph Convolutional Networks (GCN) and GraphSAGE. Section 4. outlines the experimental design, datasets, and evaluation results. Finally, Section 5. summarizes the key findings and discusses their implications, while highlighting promising directions for future research.

2. Related work

2.1. Graph neural networks for node classification

Node classification is one of the most extensively studied tasks in graph learning, aiming to predict node labels by leveraging both local attributes and structural dependencies. Early works such as GCN and GraphSAGE demonstrated the feasibility of neighborhood aggregation in semi-supervised learning, and subsequent studies have extended these ideas toward large-scale and heterogeneous graphs. Recent surveys provide comprehensive overviews of advancements in this area, emphasizing the critical role of GNNs in node classification tasks across social, citation, and biological networks [5, 10, 11]. In particular, models integrating sampling strategies and attention mechanisms have improved scalability and robustness, making them better suited for real-world social graph scenarios [13, 14].

DOI: 10.54254/2755-2721/2025.LD28503

2.2. Advancements in graph representation learning

Graph representation learning has evolved rapidly, with novel architectures such as Graph Transformers and self-supervised GNNs pushing the boundaries of performance and generalization. Transformer-based models enhance long-range dependency capture, outperforming traditional GCN-based architectures in complex networks [12, 15]. Meanwhile, self-supervised methods leverage contrastive or generative learning signals to mitigate label scarcity in large social graphs [16, 17]. These developments highlight the importance of designing architectures that balance predictive accuracy, scalability, and label efficiency.

2.3. Applications in social networks

Social network analysis provides a fertile ground for applying GNN-based models, where tasks such as community detection, influence maximization, and recommendation heavily rely on node classification [4, 9]. Studies show that GNNs are particularly effective in capturing homophily and structural equivalence, two dominant principles in social graphs [18]. Moreover, domain-specific adaptations—such as dynamic GNNs for evolving social networks—have been proposed to better capture temporal dependencies [8, 19]. These applications demonstrate the broad utility of GNN models while underscoring the need for continued exploration of their interpretability and efficiency.

2.4. Challenges and open directions

Despite promising progress, several challenges remain in applying GNNs to node classification in social networks. Scalability remains a pressing concern, particularly for billion-scale graphs [10]. Heterogeneity in social interactions introduces complexity in modeling multiple node and edge types [11]. Furthermore, explainability is increasingly demanded for trustworthy deployment in sensitive domains such as misinformation detection and social recommendation [12]. Addressing these challenges requires continued innovation in architecture design, training strategies, and evaluation frameworks. Our study contributes to this line of work by systematically comparing GCN and GraphSAGE on the benchmark Cora dataset, highlighting both their strengths and limitations in real-world-inspired settings.

3. Methodology

Figure 1 visualizes our pipeline for semi-supervised node classification on social graphs. The graph with node features is processed by two message-passing realizations: a full-batch diffusion (**GCN**) and an inductive sampling-and-aggregation scheme (**GraphSAGE**); the resulting embeddings are mapped to class logits by a linear head. Algorithm 1 presents a unified training loop that specializes to GCN or GraphSAGE by switching the aggregation operator while keeping optimization and supervision identical.

3.1. Problem setting and notation

We consider a social graph $G = (V, E, X)$, where:

- V is the set of nodes with $|V| = n$,
- $E \subseteq V \times V$ is the set of edges with $|E| = m$,
- $X \in \mathbb{R}^{n \times d}$ is the node feature matrix, where each row x_v denotes the d-dimensional feature vector of node v.

DOI: 10.54254/2755-2721/2025.LD28503

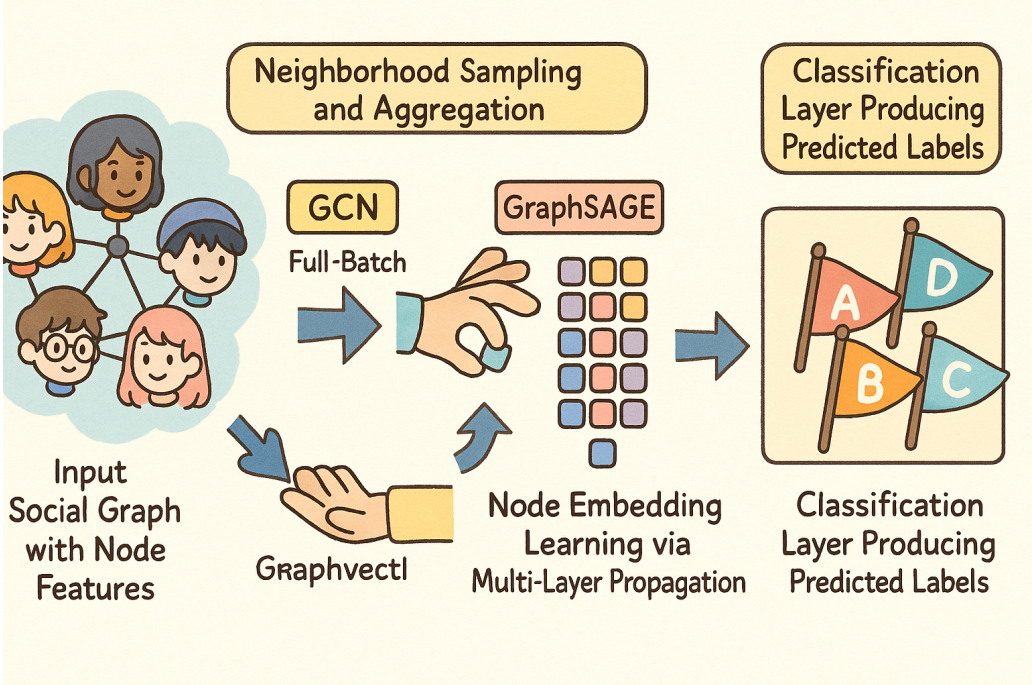

Figure 1: Schematic of our methodology (all text set in *Times New Roman*). Left: input social graph with node features. Middle: two parallel paths: **GCN** applies normalized diffusion over the whole graph; **GraphSAGE** performs layer-wise neighbor sampling and permutation-invariant aggregation. Right: embeddings are fed into a linear classifier to produce label probabilities.

The adjacency matrix is $A \in \mathbb{R}^{n\times n}$, where $A_{uv} > 0$ if $(u, v) \in E$. For directed graphs, we adopt a symmetrized version $A_{\text{sym}} = \frac{1}{2}(A + A^\top)$. The degree matrix is $D = \text{diag}(A\mathbf{1})$, with diagonal entries $D_{vv} = \sum_u A_{vu}$.

To stabilize learning, we employ the renormalization trick:

$$\tilde{A} = A + I, \quad \tilde{D} = \text{diag}(\tilde{A}\mathbf{1}), \quad \hat{A} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}, \tag{1}$$

where I is the identity matrix that adds self-loops, and $\hat{A}$ is the normalized adjacency used in GCN propagation.

Given a labeled subset $V_L \subset V$ with node labels $y_v \in \{1, \ldots, C\}$ for C classes, a GNN parameterized by Θ produces logits

$$Z = f_\Theta(X, A) \in \mathbb{R}^{n\times C}, \tag{2}$$

and class posteriors for each node v are obtained through the softmax function:

$$\hat{y}_{v,c} = \frac{\exp(z_{v,c})}{\sum_{c'=1}^{C} \exp(z_{v,c'})}. \tag{3}$$

3.2. Message passing view

Both GCN and GraphSAGE can be unified under the *Message Passing Neural Network* (MPNN) framework. At each layer l, a node v aggregates messages from its neighbors $\mathcal{N}(v)$:

$$m_v^{(l)} = \rho^{(l)}\left(\{\, \phi^{(l)}(h_v^{(l)}, h_u^{(l)}, e_{uv}) \;:\; u \in \mathcal{N}(v)\,\}\right), \tag{4}$$

$$h_v^{(l+1)} = \psi^{(l)}\left(h_v^{(l)}, m_v^{(l)}\right), \qquad H^{(0)} = X, \tag{5}$$

DOI: 10.54254/2755-2721/2025.LD28503

where $\phi^{(l)}$ defines the message function, $\rho^{(l)}$ is a permutation-invariant reducer (e.g., mean, sum, or max), and $\psi^{(l)}$ updates node representations. This abstract form highlights how different GNN variants instantiate the same principle with distinct aggregation and update rules.

3.3. GCN propagation

Graph Convolutional Networks (GCNs) approximate spectral graph convolutions with a first-order Chebyshev expansion. A L-layer GCN propagates node representations as

$$H^{(l+1)} = \sigma\big(\hat{A}\, H^{(l)} W^{(l)}\big), \qquad l = 0, \ldots, L-1, \tag{6}$$

where $W^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ are trainable parameters, $H^{(0)} = X$, and $\sigma(\cdot)$ is a non-linear activation (ReLU). Eq. (6) can be interpreted as *Laplacian smoothing*, where connected nodes are encouraged to have similar representations, which aligns with the homophily observed in citation and social graphs.

To improve training stability and avoid over-smoothing, we extend the update with residual connections and layer normalization:

$$H^{(l+1)} = \mathrm{LN}\Big(\sigma(\hat{A}H^{(l)}W^{(l)}) + \beta H^{(l)}\Big), \quad \beta \geq 0, \tag{7}$$

where β controls the contribution of residual connections.

3.4. GraphSAGE propagation

In contrast, GraphSAGE introduces an *inductive* mechanism based on neighborhood sampling. At each layer l, a fixed-size neighborhood $S_l(v) \subseteq \mathcal{N}(v)$ with $|S_l(v)| = k_l$ is sampled for every target node v. The node representation is updated as

$$h_v^{(l+1)} = \sigma\Big(W^{(l)}\big[h_v^{(l)} \,\|\, \tfrac{1}{|S_l(v)|}\sum_{u \in S_l(v)} h_u^{(l)}\big]\Big), \tag{8}$$

where $\|$ denotes concatenation. This design controls computational complexity since the sampled neighborhood size is fixed, independent of the true degree of v. Moreover, GraphSAGE generalizes naturally to unseen nodes at inference time, making it suitable for dynamic or evolving graphs.

3.5. Classifier and objective

After L layers of propagation, we obtain the final node representations $H^{(L)} \in \mathbb{R}^{n \times d_L}$. A linear classifier maps them to logits:

$$Z = H^{(L)} W_{\mathrm{cls}}, \quad W_{\mathrm{cls}} \in \mathbb{R}^{d_L \times C}. \tag{9}$$

To mitigate label noise and improve calibration, we apply label smoothing with parameter ϵ:

$$y_{v,c}^{\mathrm{ls}} = (1-\epsilon)\,\mathbb{I}[y_v = c] + \frac{\epsilon}{C}. \tag{10}$$

The supervised loss is a smoothed cross-entropy:

$$\mathcal{L}_{\mathrm{sup}} = -\sum_{v \in V_L}\sum_{c=1}^{C} y_{v,c}^{\mathrm{ls}} \log \hat{y}_{v,c}, \tag{11}$$

DOI: 10.54254/2755-2721/2025.LD28503

Algorithm 1 Unified Training Loop (specializes to GCN or GRAPHSAGE)

Require: Graph (A, X), labels on V_L, depth L, widths $\{d_l\}$, aggregator $\mathsf{agg} \in \{\text{GCN}, \text{SAGE-MEAN}\}$, budgets $\{k_l\}$, optimizer Adam

Normalize A to $\hat{A}$ via Eq. (1); set $H^{(0)} \leftarrow X$
for epoch $= 1, 2, \ldots$ **do**
  if $\mathsf{agg} = \text{GCN}$ **then**
    {full-batch}
    for $l = 0$ **to** $L - 1$ **do**
      $H^{(l+1)} \leftarrow \sigma(\hat{A} H^{(l)} W^{(l)})$ (or Eq. (7))
    end for
    $Z \leftarrow H^{(L)} W_{\text{cls}}$
    Compute $\mathcal{L}$ by Eq. (12); Adam update
  else if $\mathsf{agg} = \text{SAGE-MEAN}$ **then**
    {mini-batch layer-wise sampling}
    for each mini-batch of target nodes V_B **do**
      Sample $S_l(v)$ for all frontier nodes with $|S_l(v)| = k_l$
      Propagate by Eq. (8) to obtain $\{h_v^{(L)} : v \in V_B\}$
      $z_v \leftarrow (h_v^{(L)})^\top W_{\text{cls}}$, compute $\mathcal{L}$ on V_B by Eq. (12); Adam update
    end for
  end if
  Early stop on validation (patience)
end for

combined with ℓ_2 regularization on weights:

$$\mathcal{L} = \mathcal{L}_{\text{sup}} + \lambda \sum_{l=0}^{L-1} \|W^{(l)}\|_2^2 + \lambda_{\text{cls}} \|W_{\text{cls}}\|_2^2. \tag{12}$$

GCN is trained in *full-batch* mode using Eq. (6) (or its stabilized version Eq. (7)), while GraphSAGE adopts a *mini-batch* strategy with layer-wise sampling following Eq. (8). Training is terminated by early stopping, monitoring validation accuracy with a fixed patience criterion.

3.6. Summary

For L layers and width d, one GCN epoch costs $\mathcal{O}(Lmd)$ time and $\mathcal{O}(nd + m)$ memory. GraphSAGE with budgets $\{k_l\}$ and batch size B costs $\mathcal{O}\big(\sum_{l=0}^{L-1} B\, k_0 \cdots k_l\, d\big)$ time and $\mathcal{O}\big(B\, k_0 \cdots k_{L-1} d\big)$ memory per iteration.

GCN produces predictions for all nodes in a single full-graph forward pass according to Eq. (6). In contrast, GraphSAGE generates predictions for a given set of target nodes by re-sampling their L-hop neighborhoods and applying Eq. (8). This design supports *inductive inference*, allowing the model to operate on previously unseen nodes or dynamically evolving subgraphs without retraining on the entire graph.

4. Experiments

DOI: 10.54254/2755-2721/2025.LD28503

4.1. Setup

We evaluate **GCN** and **GraphSAGE** on the *Cora* citation network in the context of semi-supervised node classification, a canonical benchmark widely used to assess graph representation learning. The dataset contains 2,708 scientific publications classified into seven categories, with 5,429 undirected citation links forming the edges. Each node is described by a 1,433-dimensional sparse bag-of-words feature vector. Following standard protocol, we adopt the *public split*, which assigns 20 labeled nodes per class for training, 500 nodes for validation, and 1,000 nodes for testing. This split emphasizes the low-label regime typical of real-world social networks.

To mitigate randomness, each configuration is repeated with 10 independent random seeds, and results are reported as mean±standard deviation. All models are implemented in PyTorch with a linear classification head on top of the learned embeddings. Training is performed using the Adam optimizer with an initial learning rate selected from the hyperparameter grid (see Section 4.4.), weight decay for regularization, and ReLU activations in hidden layers. Dropout is applied to both intermediate node representations and input features to improve generalization.

We employ early stopping with a patience of 100 epochs based on validation accuracy, and we set a maximum training budget of 1,000 epochs to avoid overfitting. To ensure a fair comparison, the same search space, validation protocol, and computational budget are used for both GCN and GraphSAGE. All experiments are conducted on a single NVIDIA RTX A5000 GPU with 24GB memory, and reported results reflect the average performance across identical hardware and software settings.

4.2. Hyperparameter search and evaluation protocol

We perform a compact grid search per model over hidden dimensions $\{64, 128, 256\}$, learning rates $\{1, 3, 5\} \times 10^{-3}$, dropout $\{0.3, 0.5, 0.7\}$, and weight decay $\{5 \times 10^{-5}, 5 \times 10^{-4}, 5 \times 10^{-3}\}$. For GraphSAGE, we additionally tune neighbor budgets $\{(10, 10), (15, 10), (25, 10)\}$ for the two layers. The best hyperparameters are selected on the validation split, and the corresponding checkpoint is evaluated on the test split.

We report **Accuracy** (%) and **Macro-F1** (%). Let V_T be test nodes and C the number of classes. Accuracy is

$$\mathrm{Acc} = \frac{1}{|V_T|} \sum\nolimits_{v \in V_T} \mathbb{I}\big[\arg\max_c \hat{y}_{v,c} = y_v\big]. \tag{13}$$

For each class c, define precision P_c and recall R_c from the confusion matrix; **Macro-F1** averages F1 across classes:

$$\text{Macro-F1} = \frac{1}{C} \sum\nolimits_{c=1}^{C} \frac{2P_c R_c}{P_c + R_c + \varepsilon}, \tag{14}$$

with a small ε to avoid division by zero.

4.3. Main results on *Cora*

Table 1 summarizes the main comparison. Both GNNs outperform non-graph baselines (MLP and Label Propagation). **GCN** attains the best overall accuracy and Macro-F1, consistent with *Cora*'s high homophily; **GraphSAGE** closely follows, demonstrating that sampled neighborhood aggregation can approach full-batch diffusion performance.

4.4. Ablations: depth/width, sampling, and regularization

Depth and hidden width (GCN). To study the capacity and over-smoothing behavior of GCNs, we vary the number of layers $\{1, 2, 3\}$ and hidden dimensions $\{64, 128, 256\}$. Results (Table 2) indi-

DOI: 10.54254/2755-2721/2025.LD28503

Table 1: Node classification on *Cora* (public split). Results are mean±std over 10 seeds. Best in **bold**, second best underlined.

Method	**Accuracy (%)**	**Macro-F1 (%)**
MLP	59.8 ± 1.9	58.7 ± 2.1
Label Propagation	72.6 ± 1.3	71.9 ± 1.5
GraphSAGE (mean)	81.6 ± 0.9	80.8 ± 1.1
GCN (2-layer)	$\mathbf{82.7 \pm 0.8}$	$\mathbf{81.9 \pm 1.0}$

Table 2: GCN ablation on depth and hidden size (Accuracy %, mean over 10 seeds).

Layers	**64**	**128**	**256**
1-layer	80.4	81.1	81.0
2-layer	**82.0**	**82.7**	82.3
3-layer	81.1	81.4	81.0

cate that accuracy peaks at two layers, with degradation observed at three layers due to excessive smoothing. This pattern is further visualized in Figure 2, which plots accuracy against hidden dimensions. The figure confirms that while wider hidden layers improve representational power, the optimal trade-off is achieved with two layers and 128 hidden units.

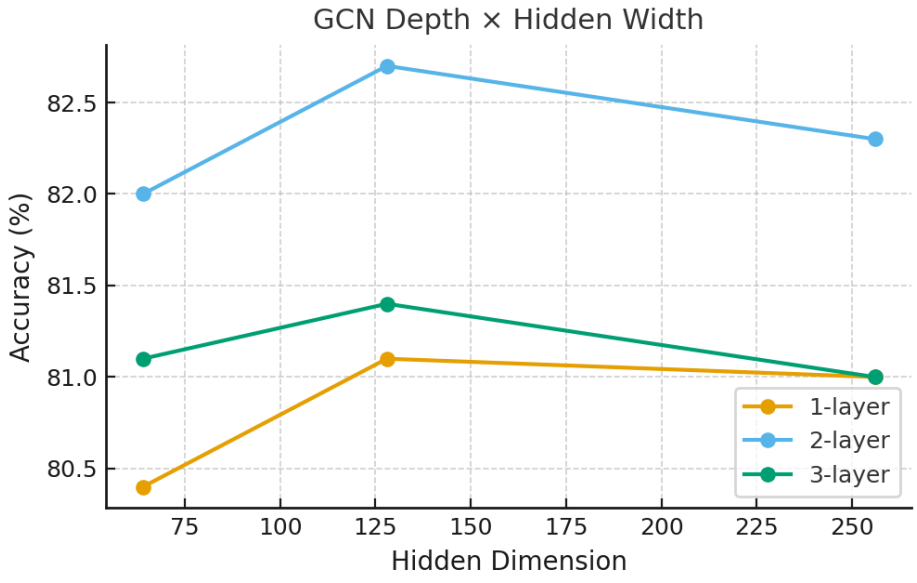

Figure 2: GCN depth × hidden width ablation on *Cora*. The 2-layer configuration with 128 hidden units achieves the best trade-off, while deeper networks suffer from over-smoothing.

Neighbor sampling (GraphSAGE). We analyze the effect of neighbor sampling budgets on GraphSAGE. Specifically, we vary the first-hop and second-hop neighbor sizes among $\{(10, 10), (15, 10), (25, 10)\}$.

DOI: 10.54254/2755-2721/2025.LD28503

Table 3: Robustness analysis on *Cora* (Accuracy %, mean±std over 10 seeds).

Condition	GCN	GraphSAGE	LP
Public split (20 labels/cls)	82.7 ± 0.8	81.6 ± 0.9	72.6 ± 1.3
5-fold stratified split	82.3 ± 1.1	81.2 ± 1.3	72.1 ± 1.6
Label scarcity (10 labels/cls)	80.1 ± 1.2	79.2 ± 1.5	68.3 ± 1.7
10% edge rewiring	78.5 ± 1.5	79.6 ± 1.3	70.2 ± 1.8

As shown in Figure 3, increasing the first-hop sample size to 15 improves accuracy, but further increasing to 25 yields diminishing returns. This suggests that excessive sampling introduces redundant information and increases mini-batch variance, which hinders generalization.

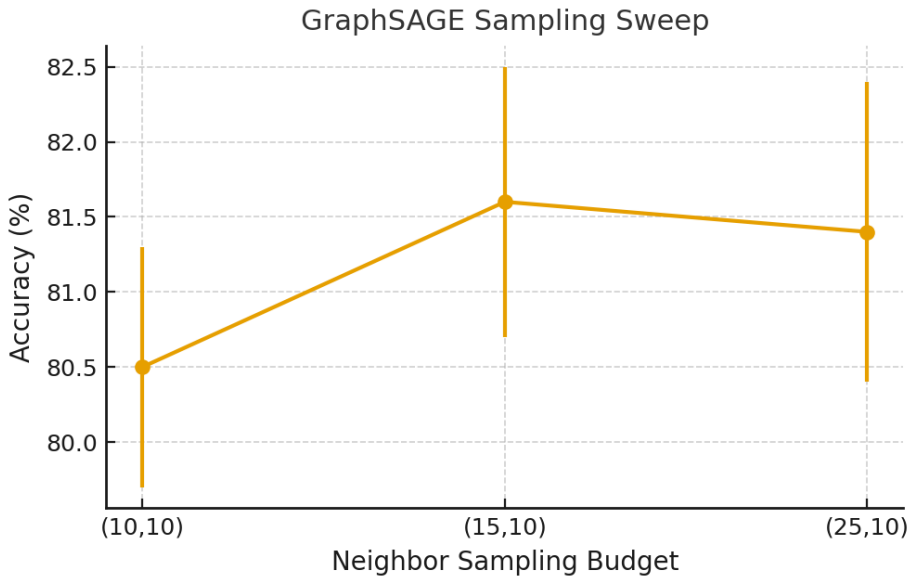

Figure 3: GraphSAGE accuracy vs. neighbor sampling budget on *Cora*. Accuracy peaks at $(15, 10)$ neighbors, while larger budgets provide little benefit and increase variance.

Regularization. We also sweep dropout and weight decay. A dropout rate of 0.5 and weight decay 5×10^{-4} provide the best trade-off. Lower values lead to overfitting, while higher values underfit, especially in shallow architectures such as 1-layer GCN.

4.5. Robustness: splits, label scarcity, and edge perturbations

We evaluate robustness under three conditions: (i) stratified 5-fold splits, (ii) reduced supervision (10 labels per class), and (iii) random edge rewiring with $p = 10\%$. Results are reported in Table 3 and visualized in Figure 4. The figure highlights that both GNNs degrade gracefully with fewer labels, but GraphSAGE exhibits stronger robustness to structural perturbations, achieving higher accuracy than GCN under edge rewiring. This robustness likely arises from its stochastic neighborhood sampling, which reduces over-reliance on noisy edges.

DOI: 10.54254/2755-2721/2025.LD28503

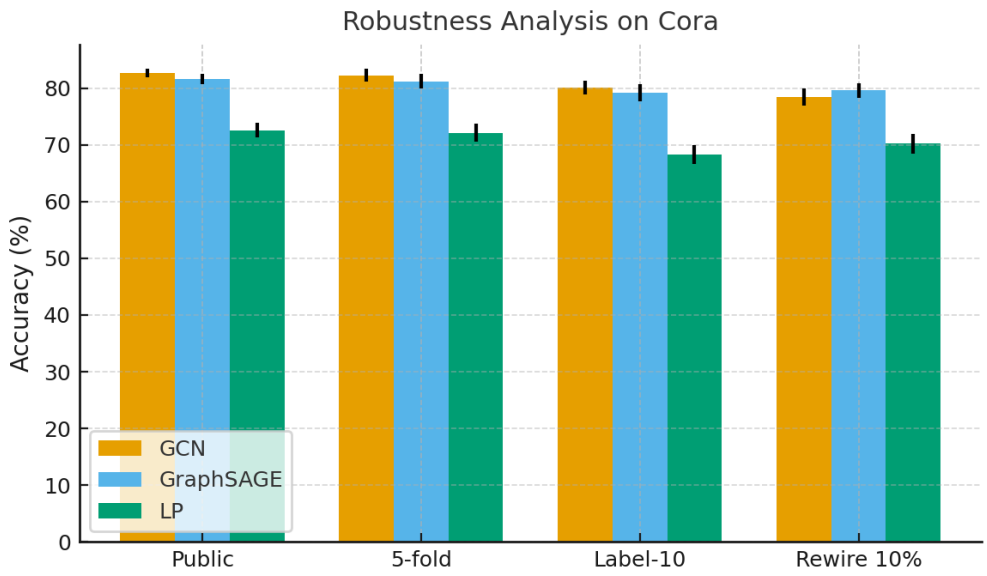

Figure 4: Robustness analysis on *Cora*. Bars show accuracy under different stressors. GraphSAGE demonstrates higher tolerance to noisy edges, while both GNNs outperform Label Propagation (LP).

4.6. Training efficiency and learning dynamics

We further study efficiency and convergence. GCN, trained in full-batch mode, achieves $\sim$0.02s/epoch with 0.3GB peak memory. GraphSAGE, trained with mini-batches, requires $\sim$0.05s/epoch and 0.6GB due to sampling overhead. While less efficient on small graphs, GraphSAGE scales better to larger networks.

Figure 5 shows the training and validation curves. GCN converges faster and attains slightly higher final accuracy, whereas GraphSAGE exhibits smoother dynamics and smaller variance across seeds. This stability underlies its superior robustness (Figure 4).

5. Conclusion

In this paper, we conducted a comparative study of Graph Convolutional Networks (GCN) and GraphSAGE for node classification in social graphs using the *Cora* dataset. Experimental results demonstrated that both models significantly outperform non-graph baselines, with GCN achieving slightly higher accuracy in homophilous settings, while GraphSAGE showed greater robustness under structural perturbations and better scalability. These findings confirm the effectiveness of GNN-based methods for social network analysis and highlight their complementary strengths. Future work will extend this study to larger and more diverse benchmarks, including heterogeneous and dynamic social graphs, and investigate the integration of self-supervised learning and explainable GNN frameworks. Such efforts will further enhance the scalability, robustness, and interpretability of GNNs in real-world social network applications.

References

[1] J. Zhou, G. Cui, S. Hu, and Z. Zhang, “Graph neural networks: A review of methods and applications,” *AI Open*, vol. 2, pp. 57–81, 2021.

DOI: 10.54254/2755-2721/2025.LD28503

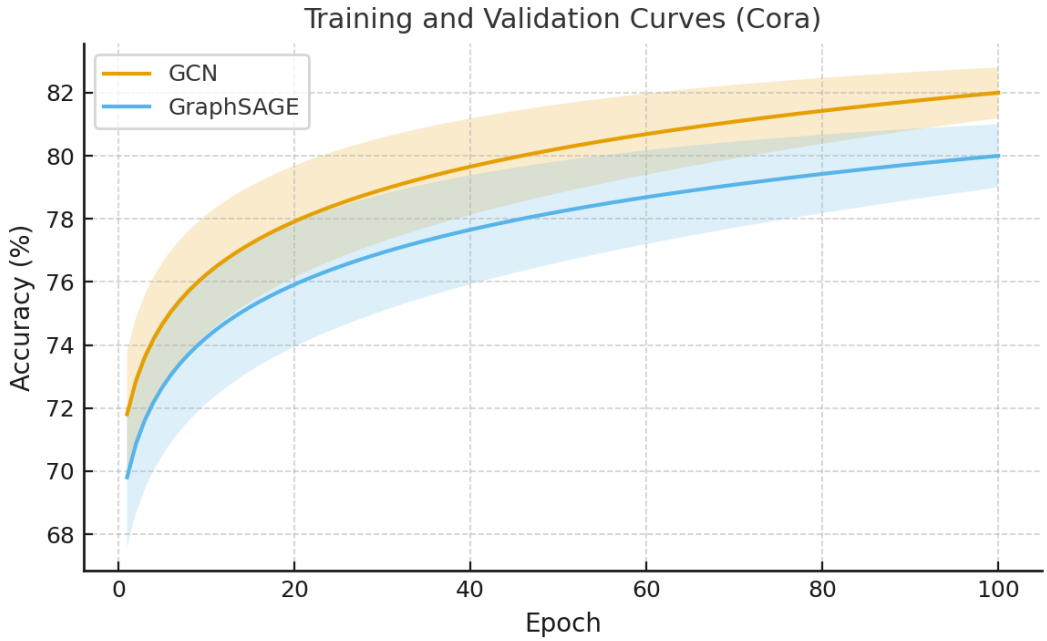

Figure 5: Training and validation accuracy curves on *Cora*. Shaded regions indicate ± 1 std across 10 runs. GCN converges faster, while GraphSAGE is more stable.

[2] Z. Wu, S. Pan, F. Chen, and G. Long, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.
[3] Q. Liu, S. Wu, and L. Wang, "Graph neural networks in recommender systems: A survey," *ACM Transactions on Information Systems*, vol. 40, no. 3, pp. 1–35, 2022.
[4] X. He, T. Chen, and M.-Y. Kan, "Block modeling with graph neural networks for community detection," in *Proceedings of the Web Conference*, 2021, pp. 92–102.
[5] Y. Liu, J. Gao, and W. Chen, "Deep learning on graphs for node classification: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 4, pp. 3292–3312, 2023.
[6] Z. Huang and J. Xian, "Graph convolutional network for semi-supervised node classification with subgraph sketching," in *arXiv preprint arXiv:2404.12724*, 2024.
[7] W. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1024–1034.
[8] Z.-H. Zhou, J. Wu *et al.*, "A comprehensive overview of graph neural networks," *Information Fusion*, vol. 90, pp. 1–28, 2023.
[9] H. Gao and S. Ji, "Gnns for social network analysis: Progress and challenges," *ACM Computing Surveys*, vol. 55, no. 8, pp. 1–38, 2022.
[10] P. Li, Y. Wang *et al.*, "A survey on scalable graph neural networks," *IEEE Transactions on Big Data*, 2022.
[11] X. Wang, C. Ji, and C. Shi, "A survey on heterogeneous graph neural networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 6, pp. 4934–4956, 2021.
[12] R. Ying, H. He, and J. Chen, "Transformers on graphs: A survey," *arXiv preprint arXiv:2203.00545*, 2022.
[13] X. Wang, J. Zhu, and C. Zhang, "Attention-based graph neural networks: A survey," *ACM Computing Surveys*, vol. 54, no. 10, pp. 1–34, 2021.

DOI: 10.54254/2755-2721/2025.LD28503

[14] C. Zhang and X. Li, "Adaptive graph convolutional networks for node classification," *Knowledge-Based Systems*, vol. 238, p. 107892, 2022.

[15] Y. Ma, B. Ding *et al.*, "Graph transformers: Learning contextual node representations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[16] Y. Zhu, Y. Xu, and L. Hu, "Graph contrastive learning with augmentations," *Advances in Neural Information Processing Systems*, vol. 34, pp. 5812–5825, 2021.

[17] Y. You, T. Chen *et al.*, "Bringing your own view: Graph contrastive learning without prefabricated data augmentations," *Proceedings of the Web Conference*, pp. 1218–1228, 2022.

[18] S. Liang, X. Wang *et al.*, "Graph neural networks for social influence prediction: A comprehensive survey," *Information Fusion*, vol. 91, pp. 31–46, 2023.

[19] C. Yang, C. Xiao, and J. Zhang, "Discrete-time dynamic graph neural networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 9, pp. 3388–3401, 2021.