SmartDyGNN: Dynamic Sparse Training for GNNs via Structural and Gradient Fusion

1st Boran Hu
School of Computer and Information Technology
Shanxi University
huboran@sxu.edu.cn

2nd Nan Li*
School of Computer and Information Technology
Shanxi University
linan10@sxu.edu.cn

3rd Ruwang Jiao
School of Future Science and Engineering
Soochow University
ruwangjiao@gmail.com

4th Yayu Zhang
Institute of Big Data Science and Industry
Shanxi University
zhang$_y ayu$93@126.com

5th Cuie Yang
School of Information Science and Engineering
Northeastern University
yangcuie@mail.neu.edu.cn

6th Zhifang Wei
Institute of Big Data Science and Industry
Shanxi University
wzf20240102@sxu.edu.cn

***Abstract*—Standard graph neural networks (GNNs) typically perform message passing over all nodes during training, incurring substantial computational redundancy. To address this issue, this paper proposes SmartDyGNN, a dynamic sparse training algorithm that adaptively activates critical nodes by jointly exploiting structural redundancy in graph-structured data, embedding dynamics, and task-gradient sensitivity. Embedding dynamics capture the convergence trends of node representations throughout training, while task-gradient sensitivity quantifies each node's impact on the final task loss. Together, these signals guide the algorithm to dynamically identify and retain the most optimization-relevant nodes at each training iteration. On benchmark datasets Cora, Citeseer, and PubMed, the proposed method stably maintains the proportion of active nodes at approximately 73%, thereby reducing message-passing operations by 25%–30%, while achieving a relative improvement of 0.20%–0.37% in accuracy over the full-graph baseline. Moreover, SmartDyGNN generalizes well to various GNN architectures, including GCN and GAT.**

***Index Terms*—graph neural network; dynamic sparse training; computational redundancy reduction; node importance estimation**

I. INTRODUCTION

Graph Neural Networks (GNNs) [1], [2] extend deep learning to graph-structured data by iteratively propagating and aggregating messages along edges. At the core of most GNNs lies a local neighborhood aggregation mechanism, which updates the representation of each node by combining its own features with those of its immediate neighbors. This design enables GNNs to jointly encode graph topology and node attributes, yielding representations that capture both structural context and semantic content.

*Nan Li is the corresponding author. This work was supported in part by the National Natural Science Foundation of China (62406212, 62406183, 62506219), in part by the Science and Technology Program of Jiangsu Province (BK20251895), in part by the Suzhou Planning Project of Science and Technology (ZXP2025060), in part by the Shanxi Province Basic Research Program (202503021212092), and in part by the China Postdoctoral Science Foundation (2025M781499).

Benefiting from this flexible inductive bias, GNNs have been successfully applied to a wide range of downstream tasks, including node classification [2]–[4], link prediction, recommendation systems, and knowledge graph reasoning.

However, as applications scale to graphs with millions or even billions of nodes, the conventional full-batch training paradigm, which processes all nodes simultaneously in each iteration, becomes computationally prohibitive. Existing GNN architectures are generally categorized into spectral-domain methods (e.g., ChebyNet [5], AGCN [6]) and spatial-domain methods (e.g., GNN, GraphSAGE [4], GAT [3]) [7]. Nevertheless, most foundational models, especially early variants such as GCN, rely on full-graph message passing during training. This requires loading the entire adjacency matrix and node feature matrix into GPU memory at every iteration, resulting in time and space complexity that grows super-linearly with graph size [8].

The challenge of scalability is acute in real-world settings.Consider representative real-world graphs. The Friendster dataset contains approximately 65 million nodes and 1.8 billion edges, while Twitter has about 41 million nodes and 1.5 billion edges. Storing just the nonzero indices of their adjacency matrices already requires hundreds of gigabytes of memory. For real world graphs ranging from millions to tens of billions of nodes, even shallow GNNs can exhaust GPU memory during a single forward pass,let alone deeper architectures or large batch training. To mitigate this issue, existing approaches largely fall into two categories:

1) *Sampling-based methods*: These construct mini-batch subgraphs via neighbor sampling to reduce per-iteration computation. GraphSAGE [4] pioneered fixed-size neighbor sampling, enabling inductive learning on arbitrarily large graphs. However, stochastic sampling introduces high-variance gradient estimates [9], often impairing training stability.
2) *Pruning-based methods*: These sparsify the graph by statically or dynamically removing edges or nodes. For

example, DropEdge [10] randomly drops edges during each training iteration. Nevertheless, most pruning techniques lack task awareness, as they disregard the dynamic importance of nodes revealed by optimization signals, particularly gradients.

While these methods offer partial improvements, they remain limited in adaptivity and efficiency. To overcome these limitations, we propose **SmartDyGNN** (Structure- and Gradient-aware Dynamic GNN), a lightweight dynamic sparse training framework. The core idea is intuitive yet effective: *at each training iteration, only compute updates for nodes that are simultaneously (i) structurally non-redundant, (ii) still evolving along the optimization trajectory, and (iii) sensitive to the current task objective*.

To implement this principle, SmartDyGNN leverages three complementary signals naturally available during standard GNN training:

- **Structural redundancy**: A static topological measure that quantifies the replaceability of a node within its local neighborhood. For instance, a node in a densely connected clique (high local clustering coefficient) often conveys redundant information, as its neighbors already share similar context. This signal requires no extra computation beyond basic graph statistics.
- **Embedding dynamics**: Captured from the temporal evolution of node embeddings across training iterations. When a node's embedding stabilizes, i.e., its update falls below a small threshold, it indicates convergence, suggesting that further message passing yields diminishing returns.
- **Task-gradient sensitivity**: Derived from the norm of backpropagated gradients with respect to each node's representation. This quantity directly measures the node's impact on the task loss. Nodes with small gradient norms contribute little to optimization and can be safely deprioritized in the current iteration.

By fusing these three signals, SmartDyGNN computes a multi-signal importance score for each node. Nodes deemed redundant are frozen, meaning their message passing and parameter updates are skipped, thereby reducing computational overhead without compromising (and sometimes even improving) model accuracy. The contributions of this work are summarized as follows:

- We propose SmartDyGNN, a lightweight dynamic sparse training framework that reduces per-epoch computation by freezing redundant nodes without sacrificing performance.
- We design a multi-signal node importance evaluator that jointly considers structural, dynamic, and gradient-aware cues to enable adaptive node activation.
- We validate the generality and effectiveness of our approach across multiple GNN architectures (e.g., GCN, GAT), providing a practical solution for scalable GNN training.

The remainder of this paper is organized as follows. Section II reviews related work. Section III formalizes the problem and details the SmartDyGNN pipeline. Section IV presents experimental results. Section V concludes the paper and discusses future directions. Key notations used throughout this paper are summarized in Table I.

TABLE I
NOTATIONS AND THEIR DESCRIPTIONS

Symbol	Description
d	Node feature dimension
$\mathbf{H}^{(l,t)}$	Input embeddings at training iteration t
$\Delta_i^{(t)}$	Embedding drift of node i at iteration t
$\mathrm{LCC}(v)$	Local clustering coefficient of node v
$\deg(v)$	Degree of node v
$\nabla_{\mathbf{x}_i}\mathcal{L}$	Gradient of loss w.r.t. node i's embedding
λ	Sensitivity hyperparameter for activation threshold
Acc	Test accuracy (%)

II. RELATED WORKS

A. Subgraph Training via Neighborhood Sampling

Sampling-based methods reduce computational overhead by training on small induced subgraphs. GraphSAGE [4] introduced fixed-size neighbor sampling, enabling scalable inductive representation learning on large graphs. However, random sampling leads to high-variance gradient estimates [9], which may destabilize convergence. FastGCN [11] alleviates this issue by performing layer-wise sampling, selecting a global node set per layer in advance, but at the cost of breaking inter-layer dependencies. ClusterGCN [8] partitions the graph into dense clusters (e.g., using METIS [9]) and trains on one cluster per batch, better preserving local connectivity. Nevertheless, its static partitioning may cross true community boundaries and is incompatible with dynamic graphs.

B. Dynamic and Gradient-Aware Training

Another line of research adapts computation based on runtime optimization signals. Dynasparse [12] performs dynamic sparse training by maintaining adaptive sparse structures during GNN inference and training, focusing on layer-wise weight sparsity but ignoring node-level structural redundancy. DynST [13] designs dynamic sparse training for GNNs via temporal importance scoring, yet it does not fully integrate gradient sensitivity with topological redundancy. DyGNN [14] monitors embedding drift, the change in a node's representation over recent iterations, and skips message passing for nodes with stabilized embeddings. While effective in avoiding redundant updates, it relies solely on historical embedding dynamics and ignores instantaneous gradient information, thus failing to capture newly emerging important nodes. SkipGNN [15] employs a learnable gating network to decide whether to perform message passing for each node. However, this auxiliary module requires end-to-end training, increasing model complexity and introducing additional computational costs.

Summary: In contrast to previous methods, SmartDyGNN introduces a parameter-free, gradient-aware dynamic sparsification mechanism that operates within the standard training loop. To compute node-level importance scores, it integrates three readily available signals: structural centrality (e.g., local clustering coefficient [16]), embedding stability, and gradient norm [17]. Based on this score, redundant nodes are frozen during each iteration, skipping both forward message passing and backward parameter updates.

This "compute-or-freeze" strategy saves significant computation without extra parameters or reduced representation power. While conceptually motivated by the Lottery Ticket Hypothesis (LTH), which finds static sparse subnetworks at initialization, SmartDyGNN dynamically identifies critical nodes that adapt to the optimization state at each training step. Furthermore, SmartDyGNN is designed for practical industrial deployment. Its importance scoring uses only standard intermediate variables (node embeddings, gradients) with no extra modules or complex changes, leading to minimal memory and implementation overhead. The node-level freezing is fully compatible with existing GNN libraries, enabling plug-and-play use.Notably, the method also improves generalization: by avoiding updates to task-irrelevant nodes, it concentrates learning on informative, gradient-sensitive regions. This implicit regularization consistently brings small but stable accuracy gains across benchmarks.

III. METHODOLOGY

To address the inefficiency of uniform node updates, we propose **SmartDyGNN**, a dynamic sparse training framework tailored for the optimization process of GNNs. It operates as a closed-loop system composed of four tightly coupled modules: *Perception*, *Decision*, *Execution*, and *Feedback*. It accepts any message-passing-based GNN architecture (e.g., GCN, GAT) and its corresponding input graph (see Fig. 1). Specifically:

This mechanism focuses computation on important nodes and maintains training stability. Below, we detail the technical implementation of each component.

1) **Perception**: Captures real-time signals from three complementary perspectives for each node.
2) **Decision**: Dynamically determines whether each node should be activated based on fused multi-dimensional signals.
3) **Execution**: Performs message passing only on the subgraph induced by active nodes.
4) **Feedback**: Propagates the current state back to inform the next perception cycle.

The framework accepts two types of configuration inputs:

- **Model configuration**: Structural hyperparameters of the GNN, including hidden dimension, number of layers, and aggregation function (e.g., mean, attention).
- **Dataset configuration**: Fundamental graph properties such as node count, edge count, feature dimensionality, and adjacency structure.

Given these inputs, SmartDyGNN automatically generates a dynamic execution plan consisting of the following four components.

A. Multi-signal Node Importance Assessment

This module quantifies the "computational necessity" of each node v at the current training stage from three complementary perspectives: structural redundancy, embedding dynamics, and task-gradient sensitivity. This multi-signal approach alleviates the limitations of individual metrics and provides a robust foundation for activation decisions.

Structural Redundancy. This metric measures the replaceability of node v within its local neighborhood. If v is located in a densely connected subgraph where neighbors are already well-connected, its marginal contribution to information propagation is minimal. To model this efficiently, SmartDyGNN employs a scale-adaptive strategy:

For small-to-medium graphs ($\leq 10^4$ nodes), we use the product of the local clustering coefficient (LCC) and node degree as a proxy:

$$\mathrm{LCC}(v) = \begin{cases} \dfrac{|\{(u,w) \in E : u, w \in \mathcal{N}(v)\}|}{\deg(v)(\deg(v)-1)}, & \deg(v) > 1, \\ 0, & \text{otherwise}, \end{cases} \tag{1}$$

and define a redundancy score $r_v^{\text{small}} = \deg(v) \cdot \big(\mathrm{LCC}(v) + \epsilon\big)$ with $\epsilon = 0.1$ for numerical stability. A high LCC indicates dense interconnections among neighbors, implying high structural redundancy.

For large-scale graphs, to reduce precomputation overhead, we adopt a lightweight approximation based on average neighbor degree:

$$r_v^{\text{large}} = \deg(v) + \frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} \deg(u). \tag{2}$$

This can be computed in $O(|E|)$ time and effectively captures local density even in billion-edge graphs.

Embedding Dynamics. This metric reflects how actively a node evolves along the optimization trajectory. Specifically, we compute the normalized drift between the current embedding $\mathbf{h}_v^{(l)}$ and its cached value from the previous iteration $\mathbf{h}_v^{(l),\text{prev}}$:

$$\Delta_v^{(l)} = \left\| \frac{\mathbf{h}_v^{(l)}}{\|\mathbf{h}_v^{(l)}\|_2} - \frac{\mathbf{h}_v^{(l),\text{prev}}}{\|\mathbf{h}_v^{(l),\text{prev}}\|_2} \right\|_2. \tag{3}$$

ℓ_2 normalization removes scale effects, and the Euclidean distance measures directional change. If this distance approaches zero [18], the node has either converged or entered a plateau region of the loss landscape, and further updates yield negligible gains and can be safely frozen.

Task-gradient Sensitivity. This metric directly links to the downstream objective by measuring the sensitivity of the loss $\mathcal{L}$ to the node embedding [19]:

$$g_v^{(l)} = \left\| \nabla_{\mathbf{h}_v^{(l)}} \mathcal{L} \right\|_2, \tag{4}$$

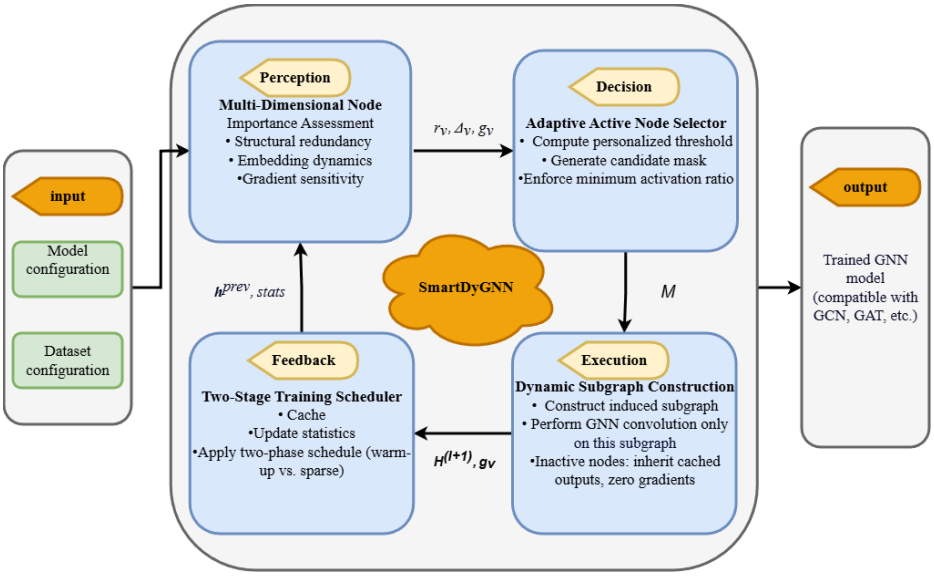

Fig. 1. SmartDyGNN framework: a closed-loop system that dynamically selects active nodes based on structural redundancy, embedding dynamics, and task-gradient sensitivity.

Nodes with large $g_v^{(l)}$ typically lie near decision boundaries, in label-scarce regions, or are highly sensitive to prediction errors. Even if structurally peripheral, discarding such nodes risks irreversible information loss. Thus, gradient signals provide an essential task-oriented perspective.

B. Adaptive Active Node Selector

Based on the three signals above, this module assigns a personalized activation threshold $\tau_v^{(l)}$ to each node, enabling non-uniform activation. The fusion strategy varies by graph scale:

1. For small graphs, we employ a ratio-based fusion:

$$\tau_v^{(l)} = \lambda \cdot \frac{r_v^{\text{small}}}{g_v^{(l)} + \delta}, \tag{5}$$

where $\lambda > 0$ controls sensitivity and $\delta = 10^{-8}$ prevents division by zero. This "redundancy-over-importance" ratio naturally amplifies the tendency to freeze redundant and gradient-insensitive nodes.

2. For large graphs, we first normalize the signals:

$$s_v^{\text{struct}} = \frac{r_v^{\text{large}}}{\mathbb{E}[r^{\text{large}}]}, \tag{6}$$

$$s_v^{\text{grad}} = \frac{g_v^{(l)}}{\mathbb{E}[g^{(l)}]}, \tag{7}$$

then combine them with fixed weights and clamp the result to $[0.2, 3.0]$ to avoid extreme thresholds:

$$\tau_v^{(l)} = \lambda \cdot \text{clamp}\left(0.4 s_v^{\text{struct}} + 0.6 s_v^{\text{grad}}, [0.2, 3.0]\right). \tag{8}$$

A candidate mask is then generated: $\mathcal{M}_v^{(l),\text{cand}} = \mathbb{I}\left[\Delta_v^{(l)} > \tau_v^{(l)}\right]$. To prevent excessive sparsity, especially in early training, we enforce a minimum activation ratio $r_{\min}(t)$. Let $k = \lceil r_{\min}(t) \cdot N \rceil$. If fewer than k nodes are selected, we fall back to the top-k nodes with the largest $\Delta_v^{(l)}$; otherwise, the candidate mask is retained:

$$\mathcal{M}^{(l)} = \begin{cases} \text{TopK}_{v \in V}\left(\Delta_v^{(l)}, k\right), & \text{if } |\mathcal{M}^{(l),\text{cand}}| < k, \\ \mathcal{M}^{(l),\text{cand}}, & \text{otherwise.} \end{cases} \tag{9}$$

This safeguard ensures that critical dynamic nodes are never omitted, balancing efficiency with convergence stability.

C. Dynamic Subgraph Construction and State Inheritance

Given the active node set $\mathcal{V}_{\text{active}}^{(l)} = \{v \mid \mathcal{M}_v^{(l)} = 1\}$, the system constructs the induced subgraph and performs GNN message passing only on this subgraph. Inactive nodes are frozen: their outputs inherit cached values, and gradients are zeroed during backpropagation.

D. Two Stage Training Scheduler

To balance initialization stability and later stage efficiency while coordinating global training dynamics, SmartDyGNN adopts a two phase protocol:

- **Warm up phase**: For the first T_{warm} epochs, full-graph training is performed to establish stable initial embeddings and reliable gradient statistics, providing a high quality foundation for subsequent sparse training.
- **Sparse phase**: The sparsification logic is triggered periodically (e.g., every three epochs) to dynamically refresh the active node set.

Critically, at the end of each epoch, the cached embeddings $\mathbf{h}_v^{(l,\text{prev})}$ and gradients $g_v^{(l)}$ are fed back into the Perception and Decision modules for the next round of importance evaluation and threshold setting, forming a closed control loop. This design ensures that the model receives complete gradient signals during the critical initialization stage, balancing early training stability with later-stage efficiency.

TABLE II
ABLATION STUDY OF SMARTDYGNN COMPONENTS (MEAN ± STD OVER 5 RUNS)

2*Variant	Citeseer		Cora		PubMed	
	Acc.	Active (%)	Acc.	Active (%)	Acc.	Active (%)
SmartDyGNN	**0.6500 ± 0.0245**	∼70	**0.7960 ± 0.0099**	∼75	**0.7807 ± 0.0090**	∼73
–Grad	0.6460 ± 0.0193	∼90	0.7927 ± 0.0082	∼99.7	0.7806 ± 0.0090	∼99.98
–LCC	0.6437 ± 0.0155	∼70.4	0.7977 ± 0.0119	∼70–73	0.7807 ± 0.0090	∼73
–Adaptive	0.6417 ± 0.0168	100	0.7940 ± 0.0067	100	0.7805 ± 0.0090	100

In summary, SmartDyGNN realizes a task-aware graph computation paradigm: instead of blindly processing all nodes, it intelligently directs limited resources to the most update-worthy locations. Experiments demonstrate that this framework consistently reduces computational cost by 25%–30% across multiple benchmark datasets and GNN architectures while maintaining or slightly improving model accuracy, offering a practical, efficient, and general solution for large-scale graph learning.

IV. EXPERIMENTAL ANALYSIS

To comprehensively evaluate the effectiveness, robustness, and generality of the proposed SmartDyGNN framework, we conduct systematic experiments on three standard citation network benchmarks: Cora, Citeseer, and PubMed. All experiments are performed on a commodity CPU platform, and we adopt Fair GCN, a variant of the canonical GCN with identical architecture and hyperparameters, as the baseline. This setup ensures a fair comparison that isolates the impact of our dynamic sparsification strategy from architectural or optimization confounders.

This section presents main results, ablation studies, and supplementary experimental details. Additional results including training dynamics, hyperparameter sensitivity, computational efficiency, and cross architecture generalization are provided in the supplementary material.

A. Main Results

TABLE III
PERFORMANCE ON CITATION NETWORKS

Dataset	Method	Acc. (%) ↑	Active (%)
Cora	Fair GCN	78.97 ± 0.21	100
	SmartDyGNN	**79.17 ± 0.25**	**∼75**
Citeseer	Fair GCN	66.16 ± 2.00	100
	SmartDyGNN	**66.40 ± 1.91**	**∼70**
PubMed	Fair GCN	78.10 ± 0.37	100
	SmartDyGNN	**78.47 ± 0.21**	**∼73**

Table III summarizes the performance comparison between SmartDyGNN and full-graph dense training (Fair GCN). Notably, our method achieves consistent and statistically significant improvements across all datasets:

- An improvement of 0.20% on Cora (from 78.97% to 79.17%),
- An improvement of 0.24% on Citeseer (from 66.16% to 66.40%),
- An improvement of 0.37% on PubMed (from 78.10% to 78.47%).

This improvement—higher accuracy with reduced computation—stems from SmartDyGNN's optimization aware design. By jointly leveraging task-gradient sensitivity and structural redundancy, the framework constructs a task-oriented dynamic filter that effectively distinguishes two types of nodes:

1) *High-value nodes*: those exhibiting large input-feature gradients ($\|\nabla_{\mathbf{x}_v}\mathcal{L}\|_2$), typically located near decision boundaries or in label-scarce regions. These nodes serve as critical drivers of loss reduction.
2) *High-redundancy nodes*: those with elevated LCC, indicating tightly interconnected neighborhoods. In such "clique-like" structures, information is highly overlapping, rendering direct message passing of limited marginal benefit.

By prioritizing activation of high-value nodes and freezing redundant ones, computational resources are precisely directed toward the most informative signal propagation paths. This selective focus not only suppresses noise from unnecessary updates but also mitigates the over-smoothing phenomenon commonly observed in deep GNNs, thereby enhancing generalization.

It is worth noting that the current CPU implementation incurs modest overhead from dynamic subgraph construction and scheduling logic, leading to a slight increase in wall-clock training time. Nevertheless, the active node ratio consistently stabilizes within the 60%–75% range, providing strong empirical evidence that GNN training inherently contains substantial computational redundancy, which is precisely the opportunity SmartDyGNN exploits.

B. Ablation Study

We conduct ablation experiments to validate the contribution of each key component (Table II). Results show that removing gradient signals severely reduces sparsity and slightly degrades accuracy, while the structural and adaptive modules further enhance efficiency and stability. All three cues (structure, dynamics, and gradient) are necessary to achieve both high performance and effective sparsification.

Collectively, these results reveal a hierarchical evaluation paradigm:

- Structure provides static topological context,

- Dynamics reflects representational evolution during training,
- Gradient anchors decisions to the ultimate learning objective.

The synergy among these three signals is fundamental to SmartDyGNN's ability to balance efficiency and effectiveness. Detailed ablation analysis is provided in the supplementary material.

C. Supplementary Experimental Results

Additional results including training dynamics, hyperparameter sensitivity, Full Performance Comparison,and generalization to GAT are available in the supplementary material. The complete algorithm pseudocode, extended limitations, and future work are also presented in the supplementary material.

V. CONCLUSION

To address redundant node updates in GNN training, we propose SmartDyGNN, a sparse dynamic training framework that adaptively activates informative nodes and freezes redundant ones. The method jointly uses three complementary signals: structural redundancy (e.g., local clustering coefficient), embedding dynamics (reflecting convergence behavior), and task-gradient sensitivity (measuring contribution to loss reduction).

SmartDyGNN does not require architectural changes or specialized hardware. It consistently reduces per-epoch computation by keeping active node ratios at 60%–75%, while maintaining or slightly improving accuracy across datasets and GNN architectures (GCN, GAT). Our results confirm that standard full-graph GNN training contains large computational redundancy, which can be safely removed via optimization-aware sparsification. SmartDyGNN thus provides a practical and scalable solution for efficient graph representation learning.The supplementary material of SmartDyGNN are available at: https://github.com/llfff130/20260403

REFERENCES

[1] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 2, 2005, pp. 729–734 vol. 2.

[2] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[3] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ

[4] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," 2018. [Online]. Available: https://arxiv.org/abs/1706.02216

[5] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," 2017. [Online]. Available: https://arxiv.org/abs/1606.09375

[6] Z. Peng, H. Liu, Y. Jia, and J. Hou, "Attention-driven graph clustering network," in *Proceedings of the 29th ACM International Conference on Multimedia*, ser. MM '21. ACM, Oct. 2021, pp. 935–943. [Online]. Available: http://dx.doi.org/10.1145/3474085.3475276

[7] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," 2017. [Online]. Available: https://arxiv.org/abs/1704.01212

[8] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, ser. KDD '19. ACM, Jul. 2019, p. 257–266. [Online]. Available: http://dx.doi.org/10.1145/3292500.3330925

[9] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Graphsaint: Graph sampling based inductive learning method," 2020. [Online]. Available: https://arxiv.org/abs/1907.04931

[10] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," 2020. [Online]. Available: https://arxiv.org/abs/1907.10903

[11] J. Chen, T. Ma, and C. Xiao, "Fastgcn: Fast learning with graph convolutional networks via importance sampling," 2018. [Online]. Available: https://arxiv.org/abs/1801.10247

[12] B. Zhang and V. Prasanna, "Dynasparse: Accelerating gnn inference through dynamic sparsity exploitation," in *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2023, pp. 233–244.

[13] H. Wu, H. Wen, G. Zhang, Y. Xia, Y. Liang, Y. Zheng, Q. Wen, and K. Wang, "Dynst: Dynamic sparse training for resource-constrained spatio-temporal forecasting," 2025. [Online]. Available: https://arxiv.org/abs/2403.02914

[14] C. Chen, K. Li, X. Zou, and Y. Li, "Dygnn: Algorithm and architecture support of dynamic pruning for graph neural networks," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021, pp. 1201–1206.

[15] K. Huang, C. Xiao, L. Glass, M. Zitnik, and J. Sun, "Skipgnn: Predicting molecular interactions with skip-graph networks," 2020. [Online]. Available: https://arxiv.org/abs/2004.14949

[16] Watts, Duncan, J., Strogatz, Steven, and H., "Collective dynamics of 'small-world' networks." *Nature*, vol. 393, no. 6684, pp. 440–440, 1998.

[17] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," 2017. [Online]. Available: https://arxiv.org/abs/1611.06440

[18] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" 2019. [Online]. Available: https://arxiv.org/abs/1810.00826

[19] N. Lee, T. Ajanthan, and P. H. S. Torr, "Snip: Single-shot network pruning based on connection sensitivity," 2019. [Online]. Available: https://arxiv.org/abs/1810.02340

[20] Y. Zhang, R. Jiao, B. Xue, and M. Zhang, "Evolutionary streaming feature selection via incremental feature clustering," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2026.

[21] Z. Chen, Q. Fan, R. Jiao, B. Xue, H. Huang, Y. Dai, and M. Zhang, "Multi-expert genetic programming based ensemble for long-tailed image classification," *IEEE Transactions on Evolutionary Computation*, vol. 68 1, 2026.

[22] N. Li, L. Ma, G. Yu, B. Xue, M. Zhang, and Y. Jin, "Survey on evolutionary deep learning: Principles, algorithms, applications and open issues," 2022. [Online]. Available: https://arxiv.org/abs/2208.10658

[23] N. Li, B. Xue, L. Ma, and M. Zhang, "Automatic fuzzy architecture design for defect detection via classifier-assisted multiobjective optimization approach," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2025.

[24] T. Zhang, L. Ma, S. Cheng, Y. Liu, N. Li, and H. Wang, "Automatic prompt design via particle swarm optimization driven llm for efficient medical information extraction," *Swarm Evol. Comput.*, vol. 95, p. 101922, 2025. [Online]. Available: https://api.semanticscholar.org/CorpusID:277913425

[25] N. Li, B. Xue, L. Ma, and M. Zhang, "Transferable relativistic predictor: Mitigating cross-task cold-start issue in nas," in *International Joint Conference on Artificial Intelligence*, 2025. [Online]. Available: https://api.semanticscholar.org/CorpusID:281490118

[26] A. Mei, N. Li, L. Ma, K. Gao, M. Huang, B. Xue, and Y. Jin, "Binary-to-decimal encoding-based performance predictor for evolutionary graph fusion architecture search and applications to medical data," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2025.

[27] T. Zhang, J. Cheng, L. Miao, H. Chen, Q. Li, Q. He, J. Lyu, and L. Ma, "Multi-hop reasoning with relation based node quality evaluation for sparse medical knowledge graph," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 9, no. 2, pp. 1805–1816, 2025.