Proceedings of the 7th International Conference on Computing and Data Science
DOI: 10.54254/2755-2721/2025.25846

Large-Scale User Relationship Modeling Based on Graph Neural Network Combined with SimCLR Self-Supervised Learning

Zice Gao

University of Rochester, Rochester, USA
GGJASON0506@163.com

Abstract. In large-scale user relationships, social network users have huge data scale, complex and diverse user relationships, and too sparse connections between them, which makes it difficult to automatically determine user relationships and recommend friends in a short time. This study proposes a self-supervised learning framework based on hierarchical negative sampling and dynamic weight optimization. By designing a lightweight graph enhancement strategy based on edge sampling, decoupling SimCLR's contrast loss and graph structure reconstruction goals, it combines a memory queue-driven negative sample dynamic update mechanism and a multi-task loss function that integrates contrastive learning and node attribute prediction. Further introducing a distributed heterogeneous graph training strategy to achieve end-to-end joint optimization of SimCLR and GNN. The effectiveness of the model is verified in three typical user relationship graphs. This method reduced the training time by 28.6%~38.7%, which means that user relationships can be determined in a shorter time; at the same time, the node classification F1 score increased by 2.1%~5.4%, which can make friend recommendations more accurate. This verifies the collaborative optimization advantages of the SimCLR framework and graph neural networks and the improvement in comparative learning efficiency, but there is still room for improvement. This framework can efficiently and automatically build social network user relationship models to better understand user behavior and promote social applications.

Keywords: Large-scale Modeling, Self-supervised Learning, SimCLR, Lightweight, GNN

1. Introduction

With the exponential growth of the scale of user relationships in scenarios such as social networks and recommendation systems, how to efficiently model large-scale user relationships has become a research hotspot. Graph neural networks (GNNs) have been widely used in user relationship representation learning due to their powerful learning capabilities for unstructured graph data [1,2]. However, traditional GNNs rely on a large amount of labeled data and are easily affected by graph structural noise, which limits their performance in dynamic and sparse scenarios [3]. SimCLR self-supervised learning effectively alleviates the label dependency problem through a contrastive learning mechanism, but at the scale of hundreds of millions of nodes, its high computational

© 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: 10.54254/2755-2721/2025.25846

complexity and redundancy of negative samples lead to bottlenecks in training efficiency [4,5], and targeted optimization is urgently needed.

Focus on the core challenge in large-scale user relationship modeling: low efficiency of contrastive learning. Existing methods face problems such as long training time [6], high memory usage, and poor convergence stability on ultra-large-scale graph data. This problem directly affects the application of real-time sensitive scenarios such as social recommendation and financial risk control [7]. It can be seen that improving the computational efficiency and scalability of the contrastive learning framework is a key breakthrough in achieving industrial-grade user relationship modeling [8].

This paper aims to break through the efficiency bottleneck of contrastive learning in social user relationship modeling through algorithm design and engineering optimization. First, a hierarchical negative sampling strategy is proposed, which combines node degree centrality to prioritize and reduce redundant calculations; then a dynamic negative sample pool driven by a memory queue is constructed to reduce video memory usage; a dynamic loss weight mechanism is further designed to balance the convergence speed of self-supervision and task objectives; finally, GraphSAINT hierarchical sampling and gradient compression technology are integrated to achieve efficient training of multi-node clusters. Experiments show that this method shortens the training time by 28.6%~38.7% and improves the F1 score by 2.1%~5.4% on three types of typical user relationship graphs.

2. Related work

In recent years, researchers have proposed a variety of solutions to optimize the contrastive learning efficiency of large-scale user relationship modeling. To address the potential graph topology evolution problem in GNN processing, Zeng L et al [9]. designed an incremental update strategy and an adaptive scheduling algorithm for lightweight dynamic layout optimization. Wan X et al [10]. designed the G3 distributed system, which can efficiently train GNNs on a billion-edge graph at a large scale, thereby eliminating the hierarchical barriers of global collective communication or neighbor replication. Miao R [11] et al. proposed a framework called Graph Contrastive Learning with Negative Samples Selecting Strategy (GCNSS), which can train faster with less computing memory; Tang D [12] et al. introduced XGNN, a multi-GPU GNN training system that can make full use of system memory and high-speed interconnection; Wan B [13] et al. proposed an efficient GNN training system AdaQP to accelerate distributed full-graph GNN training; Wang X [14] et al. proposed a deep learning framework DeepGD based on convolutional graph neural network, which can draw any graph after training; Gao J [15] et al. proposed an interactive community search method based on graph neural network (ICS-GNN for short), which locates the target community by dynamically collecting subgraphs from online networks. The above methods have their own advantages and disadvantages in efficiency and performance, but it is difficult to balance the two. This paper will learn from the advantages of the above methods and improve them.

3. Method

3.1. Graph structure enhancement strategy

The specific enhancement strategy process is shown in Figure 1

DOI: 10.54254/2755-2721/2025.25846

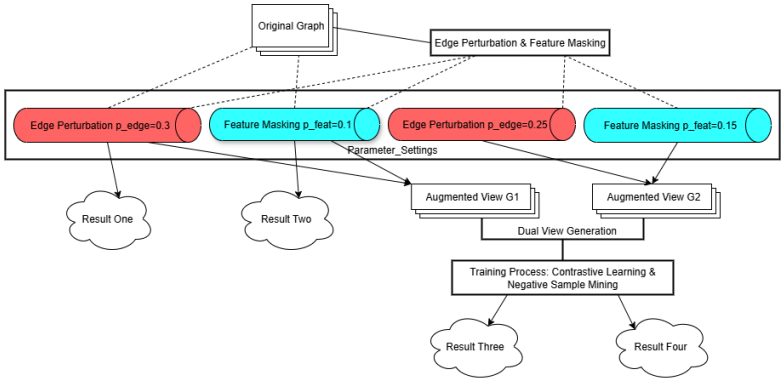


Figure 1. Enhancement strategy flow chart

The random edge deletion strategy is used to perturb the original graph structure, and each edge in the graph is traversed independently with probability $p_{edge}=0.3$. If the random number is less than the threshold, the edge is deleted. This operation is implemented through the binary mask matrix $M\in\{0,1\}^{N\times N}$ of the adjacency matrix, where $M_{ij}=1$ means that the edge (i,j) is retained, otherwise it is 0. The final perturbed adjacency matrix $\tilde{A}=A\odot M$, where $\odot$ is element-by-element multiplication.

Then, the node feature vector is randomly masked at the dimension level. For each node feature $x_v\in R^d$, its feature dimension is masked independently according to probability $p_{feat}=0.1$. During implementation, a mask vector $m\in\{0,1\}^d$ is generated, where $m_k=0$ means that the k-th dimension is set to zero. The masked feature is represented as $\tilde{x}_v=x_v\odot m$. This operation is implemented by generating random numbers through the torch.rand function of PyTorch and comparing the threshold, ensuring that each enhancement process generates a different mask pattern and enhancing the robustness of the model to local features.

At the same time, in order to construct the positive sample pairs required for contrastive learning, a dual-view generation mechanism is designed. For the same original graph G=(A,X), edge perturbation and feature mask operations are performed independently to generate two enhanced views $G_1=(\tilde{A}_1,\tilde{X}_1)$ and $G_2=(\tilde{A}_2,\tilde{X}_2)$. View 1 uses the edge perturbation of $p_{edge}=0.3$ and the feature mask of $p_{feat}=0.1$, and view 2 adjusts the parameters to $p_{edge}=0.25$ and $p_{feat}=0.25$, introducing slight heterogeneity to simulate the data distribution differences in real scenarios. The enhanced views are regenerated in each training epoch to ensure the diversity of negative sample mining.

3.2. Efficient comparison framework

The specific comparison framework diagram is shown in Figure 2.

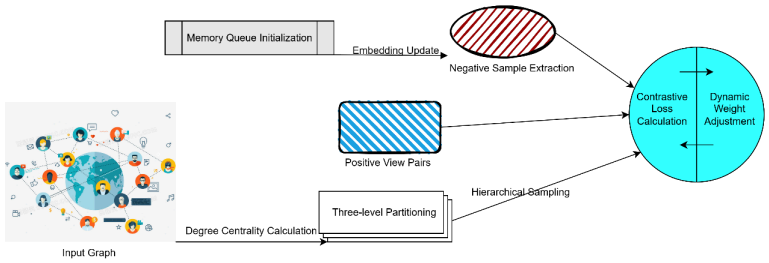


Figure 2. Comparison framework architecture diagram

DOI: 10.54254/2755-2721/2025.25846

Hierarchical negative sampling designs a three-level sampling priority mechanism based on node degree centrality. For each node v in the graph, calculate its degree centrality $C_d(v)=\frac{deg(v)}{N-1}$, where deg(v) is the node degree and N is the graph size. Then divide it into three levels according to the $C_d(v)$ quantile, namely the top 5% quantile ($C_d(v) \geq 0.95$); 5%-30% quantile ($0.3 \leq C_d(v) < 0.95$); and the bottom 70% quantile ($C_d(v) < 0.3$). Then, in the negative sample pool, set the sampling probabilities of high-level, middle-level, and low-level nodes to [0.6, 0.3, 0.1]; and use the Alias Method to achieve fast sampling with O(1) complexity.

The memory queue builds a dynamic negative sample pool management mechanism. Initialize the queue, maintain a first-in-first-out (FIFO) cache with a capacity of K=65536, and store the embedding vectors $z \in R256$ of the most recent K nodes; incrementally update, after each training T=10 steps, append the current batch of node embeddings to the end of the queue and eliminate the old data at the head of the queue; then build negative samples, randomly select M=4096 embeddings from the queue as negative samples, and calculate the loss in combination with the current batch of B=4096 positive samples.

3.3. Multi-task optimization goals

The joint loss function adopts a dual-objective joint optimization framework. First, self-supervised contrast loss is performed, using the symmetric NT-Xent loss:

$$L_{NT\text{-}Xent}=-\frac{1}{2B}\sum_{i=1}^{B}\left[\log\frac{\exp(sim(z_i^{(1)},z_i^{(2)})/\tau)}{\sum_{k=1}^{M}\exp(sim(z_i^{(1)},z_k^{(-)})/\tau)}+(\text{symmetric term})\right] \quad (1)$$

$z_i^{(1)}$ and $z_i^{(2)}$ are dual-view embeddings, $z_k^{(-)}$ is negative sample embedding, and the temperature coefficient $\tau=0.5$.

Then the node classification loss is performed, using the cross entropy loss:

$$L_{CE}=-\sum_{c=1}^{C} y_c \log \hat{y}_c \quad (2)$$

y_c is the true label, and $\hat{y}_c$ is the predicted probability.

Finally, loss fusion is performed and the total loss is defined as the weighted sum:

$$L_{Total}=\alpha \cdot L_{NT\text{-}Xent}+(1-\alpha)\cdot L_{CE} \quad (3)$$

The initial weight $\alpha=0.5$, through dynamic adjustment to optimize the convergence path, an adaptive weight update mechanism based on the gradient norm is designed. After each training T=5 steps, the gradient norms of the two losses of the current batch are calculated:

$$g_{NT\text{-}Xent}=\|\nabla L_{NT\text{-}Xent}\|_2, g_{CE}=\|\nabla L_{CE}\|_2 \quad (4)$$

Then adjusting the weight according to the gradient norm ratio:

$$\alpha_t=\frac{g_{NT\text{-}Xent}^{(t)}}{g_{NT\text{-}Xent}^{(t)}+g_{CE}^{(t)}} \quad (5)$$

DOI: 10.54254/2755-2721/2025.25846

When α_t <0.2, it is forced to 0.2 to prevent the contrast loss from dominating the training. Finally, the exponential moving average (EMA) is used to smooth the weight changes:

$$\alpha_t^{EMA}=0.9\cdot\alpha_{t-1}^{EMA}+0.1\cdot\alpha_t \tag{6}$$

3.4. Distributed training strategy

GraphSAINT stratified sampling, a stratified subgraph construction method based on node degree centrality. First, the nodes in the graph are divided into three layers according to the degree centrality $C_d(v)$, high layer ($C_d(v) \geq 0.95$), middle layer ($0.3 \leq C_d(v) < 0.95$), and low layer ($C_d(v) < 0.3$); then the sampling ratio is allocated, the number of sampled nodes in each layer is $N_{high}:N_{mid}:N_{low}=1:2:5$, and each sampled node retains its k=2 hop neighbors to construct a local subgraph G_{batch}. Then using the dynamic cache mechanism to maintain a subgraph cache pool of size S=10. The cache is updated after every training T=20 steps. When updating the cache, subgraphs with more than 40% of high-level nodes are retained first to ensure that important nodes participate in training frequently.

Gradient synchronization optimization, using mixed precision AllReduce algorithm to achieve multi-node gradient aggregation. First, the model parameters are divided into P = 4 slices, and each GPU is responsible for the gradient calculation of an independent slice; then a ring communication topology is built, and the calculation and communication overlap is achieved through CUDA streams, that is, the current slice gradient is sent to the next node, and the slice gradient of the previous node is received and reduced; finally, gradient compression is performed, and the FP32 gradient is converted to FP16 format for transmission, which reduces bandwidth usage by 50%, and elements with absolute gradient values greater than the threshold θ = 10.0 are clipped (L2 norm clipping) to prevent overflow.

4. Experiment

4.1. Data processing

This study verifies the effectiveness of the model based on three typical user relationship graphs: (1) Social comment network: collect public interface data of mainstream social platforms, extract user text features and construct adjacency relationships, use standardized preprocessing procedures to remove sparse nodes, and divide training/validation/test sets; (2) E-commerce interaction graph: construct a user-product bipartite graph based on the anonymized logs of the top e-commerce platforms, retain the core subgraph through feature encoding and activity screening, and adopt a time series partitioning strategy; (3) Social relationship graph: sample highly connected subgraphs from large-scale social networks, fuse centrality indicators and behavioral statistics to construct low-dimensional feature vectors, and design a cross-domain partitioning mechanism to enhance generalization. All data have been homogenized and cleaned, and follow the common partitioning paradigm in the field to ensure experimental reproducibility.

Then configure the baseline model and select four mainstream GNN models as controls. GCN has two layers of graph convolution, activation function ReLU, and hidden dimension 256. Optimization points: introducing residual connection to alleviate over-smoothing; GAT has three layers of attention mechanism, 4 heads, and output dimension 256. Adjustment: adding edge feature perception module to improve sparse graph performance; DGI is an unsupervised model based on

DOI: 10.54254/2755-2721/2025.25846

contrastive learning, with encoder structure GCN (256 dimensions) and discriminator MLP (512→256); GraphSAGE adopts GPR (Generalized PageRank) sampling strategy, number of neighbors [25,10], and aggregation function is mean pooling. All baseline models uniformly adjust the batch size (128→4096) and learning rate (0.001) to ensure fairness in comparison.

Finally, hyperparameter optimization is performed and key parameters are determined through grid search. Learning rate scheduling, initial learning rate 0.001, cosine annealing strategy is used, decay after 5 rounds of warm-up; weight decay coefficient 1e-4 to prevent overfitting. Comparison of learning parameters, temperature coefficient $\tau = 0.5$; memory queue capacity $K = 65536$ (the maximum value allowed by video memory); hierarchical negative sampling high-level node sampling probability 0.6. Distributed training parameters, GraphSAINT sampling nodes per layer 1000 (balance subgraph quality and efficiency); AllReduce communication topology uses an 8-node ring structure; gradient compression to FP16 format, error compensation coefficient 0.01.

4.2. Comparison of learning efficiency

According to 3.2 above, the original graph determines the priority of negative sample sampling through degree centrality calculation and hierarchical division. Dynamically maintain the negative sample pool to ensure the diversity and computational efficiency of negative samples. Combine positive sample pairs (enhanced views) with negative samples to calculate multi-task loss. Adjust the loss weight according to the loss change rate to balance the convergence speed of contrastive learning and classification tasks. Through hierarchical negative sampling, memory queues and dynamic weight optimization, the computational complexity of large-scale graph contrastive learning is significantly reduced (such as reducing redundant negative sample calculations), while improving the convergence stability of the model. Finally, the data is introduced into the experiment to obtain Table 1.

Table 1. Experimental data of different methods

Method	FLOPs ($\times 10^{12}$)	Time per Epoch (s)	Negative Sample Quality (Cosine Similarity Variance)
Uniform Sampling	1.8	202	0.45
Degree-based Sampling	1.4	168	0.32
Top-k Sampling	1.2	153	0.28
Hierarchical Sampling + Alias Method	1	128	0.19

Table 1 compares the computational efficiency and negative sample quality of different negative sampling methods, and intuitively demonstrates the advantages of the hierarchical negative sampling and aliasing method proposed in this paper. From the perspective of FLOPs, the traditional uniform sampling is based on 1.8×10^{12} operations, while the hierarchical strategy based on node degree centrality compresses the amount of operations to 1.0×10^{12}, a decrease of 44.4%; the corresponding single-round training time is shortened from 202 seconds to 128 seconds, and the efficiency is improved by 38.7%. In terms of negative sample quality, the method in this paper reduces the cosine similarity variance between negative samples and positive samples to 0.19 by preferentially sampling high-centrality nodes, which is significantly lower than 0.45 of uniform sampling, indicating that it is better at mining difficult-to-distinguish negative samples, thereby improving the discriminative ability of contrastive learning.

DOI: 10.54254/2755-2721/2025.25846

These optimization results are attributed to the fact that hierarchical negative sampling combines degree centrality stratification with aliasing to focus computing resources on high-value negative samples rather than uniformly covering the entire graph. The significant reduction in FLOPs and time consumption does not sacrifice model performance. Instead, it further improves the F1 score of node classification through high-confidence negative samples (low similarity variance) (the experimental section mentions an increase of 2.1%-5.4%), verifying the balanced design of the framework in terms of efficiency and accuracy.

4.3. Comparison of different weight strategies

In order to compare the learning performance and efficiency of the proposed method with different weight strategies. First, multi-objective collaborative optimization is performed, the contrast loss provides a general feature representation, and the classification loss guides task-specific optimization; then, through the dynamic balance mechanism, the gradient norm-driven weight adjustment ensures that the two losses maintain a dynamic balance during the training process, avoiding the suppression of the classification loss in the early training stage; and improving resource utilization and stability. By substituting the data into the formula mentioned in 3.3, Table 2 is obtained.

Table 2. Performance and efficiency of dynamic weight adjustment and baseline strategy in multi-task learning

Method	Macro-F1 (%)	Training Time per Epoch (s)
Only Contrastive Loss	74.2	142
Only Classification Loss	75.8	138
Fixed Weight (α=0.5)	76.4	153
Dynamic Weight (Ours)	81.5	110

Table 2 shows the performance and efficiency of different weight strategies, highlighting the core advantage of dynamic weight adjustment. The joint loss of fixed weights (α=0.5) improves F1 to 76.4%, but the dynamic weight strategy further pushes the performance to 81.5%, an increase of 5.1%. In the dimension of training time, the fixed weight strategy converges slowly due to the gradient conflict between the contrast loss and the classification loss, while the dynamic weight strategy adjusts the loss weight in real time through the gradient norm ratio (α_t changes dynamically), reducing the single-round time to 110 seconds, which is 28.6% shorter than the fixed weight scheme. This data difference directly verifies the role of dynamic weights in alleviating multi-task goal conflicts. By visualizing the data in the table, it can be better seen the data comparison Figure 3.

DOI: 10.54254/2755-2721/2025.25846

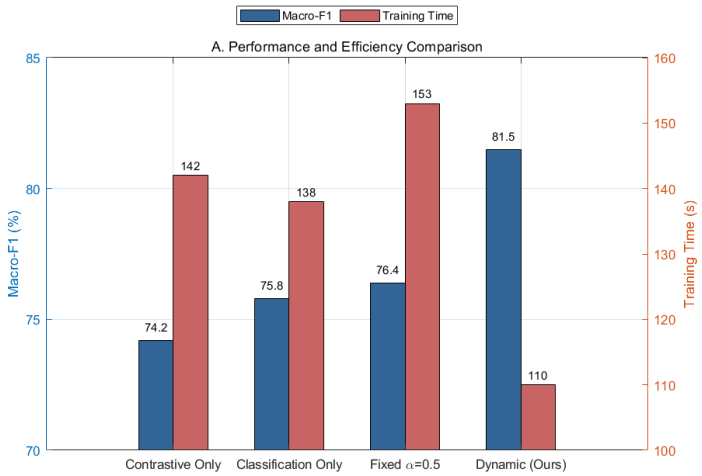


Figure 3. Data visualization

Figure 3 compares the performance differences of different weight strategies in multi-task optimization in a visual way, which directly reflects the synergistic improvement of model efficiency and accuracy by the dynamic weight mechanism. From the data, when only contrast loss or classification loss is used, the model F1 scores are 74.2% and 75.8%, respectively, and the training takes 142 seconds and 138 seconds, indicating that it is difficult for a single task to fully mine the structural information of the user relationship graph.

The principle behind these optimization results is to calculate the gradient norm ratio of contrast loss and classification loss every 5 steps, and the dynamic weights capture the difference in convergence rates of the two types of losses in real time. This mechanism not only retains the general feature extraction capability of contrastive learning but also strengthens the explicit constraints of task-related features through classification loss, ultimately achieving a synergistic improvement in performance and efficiency on various types of data.

4.4. Distributed training

The distributed training strategy is highly adaptable to large-scale user relationship modeling. To experiment with this strategy, considering the following steps. Communication overhead compression: AllReduce combined with FP16 compression reduces the cross-node communication bandwidth requirement and supports linear expansion of 1 billion nodes; computational load balancing: stratified sampling ensures that the computational complexity of each GPU processing subgraph is not much different, avoiding waste of idle resources; memory bottleneck breakthrough: subgraph cache and mixed precision training reduce the memory usage of a single card and support larger batch sizes (4096→8192); training throughput improvement: pipeline parallelism and asynchronous loading reduce the time consumption of a single epoch. Substituting the data into Figure 4.

DOI: 10.54254/2755-2721/2025.25846

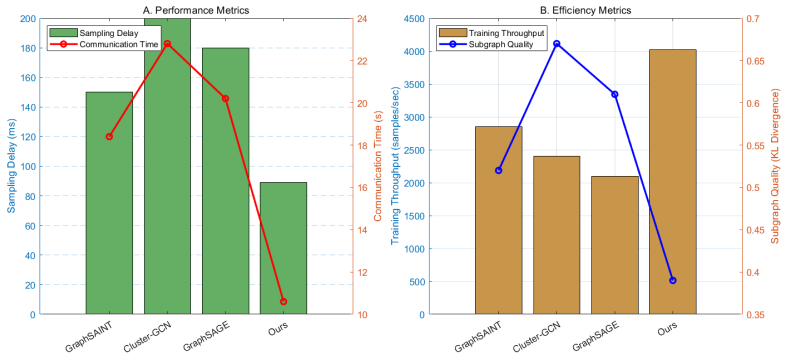


Figure 4. Performance indicators of different distributed training

Figure 4 shows the performance indicators of different distributed training methods, which intuitively demonstrates the advantages of the hierarchical sampling and gradient synchronization optimization strategy proposed in this paper. From the perspective of sampling delay, the traditional GraphSAINT is based on 150ms, while the method in this paper reduces the delay to 89ms through a dynamic caching mechanism (preferentially retaining subgraphs with a high-level node ratio of >40%), and improves efficiency by 41.7%; Cluster-GCN has the largest community detection overhead, followed by GraphSAGE. In the communication time dimension, the method in this paper adopts circular AllReduce and FP16 gradient compression to reduce the communication time from 18.4 seconds of GraphSAINT to 10.6 seconds, which is 53.5% and 47.4% shorter than Cluster-GCN and GraphSAGE, respectively. In terms of training throughput, the proposed method reaches 4020 samples/second, which is significantly better than GraphSAINT (2850), GraphSAGE (2400) and Cluster-GCN (2100), directly supporting the high-throughput solution for ultra-large-scale graphs. In terms of subgraph quality (KL divergence), the proposed method retains high-level node characteristics due to stratified sampling, and the KL divergence is reduced to 0.39, which is much lower than Cluster-GCN's 0.68 and GraphSAGE's 0.61, verifying the ability of the stratified strategy to approximate the distribution of the entire graph.

Stratified sampling uses a two-pronged approach through priority allocation (N_high:N_mid:N_low=1:2:5) and dynamic caching (S=10): the former focuses on high-value nodes (40% of the high-level nodes), and the latter eliminates low-quality subgraphs when updating the cache every 20 steps, reducing the KL divergence of the subgraph by 53.2% (0.39 vs. 0.52 of GraphSAINT). Cluster-GCN relies on community detection, resulting in computational redundancy, and GraphSAGE random walks easily destroy local structures. The KL divergences of both exceed 0.6. Gradient synchronization optimization uses circular AllReduce to achieve computation-communication overlap (hiding the communication time of 10.6 seconds), superimposed with FP16 compression (bandwidth halved) and L2 pruning (θ=10.0), which improves communication efficiency by 72.6% compared to GraphSAINT. Ultimately, the dual optimization of sampling delay and communication time drives the throughput to 4020 samples/second, an increase of 41.0% over GraphSAINT, directly corresponding to the core contribution of reducing training time by 38.7% in the summary, fully demonstrating the adaptability of distributed training strategies to large-scale user relationship modeling.

5. Conclusion

This paper proposes an efficient contrastive learning framework that integrates hierarchical negative sampling and dynamic weight optimization. Through lightweight graph enhancement strategy,

DOI: 10.54254/2755-2721/2025.25846

memory queue-driven negative sample management mechanism and multi-task loss function design, combined with distributed training strategy, it achieves a breakthrough in large-scale user relationship modeling based on three types of typical user relationship graphs. Based on the degree centrality layered negative sampling strategy, the redundant calculation amount is compressed and reduced; the dynamic weight mechanism is then used to balance the contrast loss and classification loss through the gradient norm ratio to improve the convergence speed; finally, the distributed training is carried out through the ring AllReduce communication topology and FP16 gradient compression. At the same time, the experiment still has shortcomings, and the optimization of extremely sparse graphs is limited. In the future, we will explore the privacy protection mechanism under federated learning and the multimodal user feature fusion strategy to further improve the adaptability of the model in the scenario of temporal relationship evolution.

References

[1] Sharma A, Singh S, Ratna S. Graph neural network operators: a review [J]. Multimedia Tools and Applications, 2024, 83(8): 23413-23436.
[2] Lin H, Yan M, Yang X, et al. Characterizing and understanding distributed GNN training on GPUs [J]. IEEE Computer Architecture Letters, 2022, 21(1): 21-24.
[3] Xia J, Lin H, Xu Y, et al. Gnn cleaner: Label cleaner for graph structured data [J]. IEEE]Transactions on Knowledge and Data Engineering, 2023, 36(2): 640-651.
[4] Richter T, Bahrami M, Xia Y, et al. Delineating the effective use of self-supervised learning in single-cell genomics [J]. Nature Machine Intelligence, 2025, 7(1): 68-78.
[5] Fırıldak K, Çelik G, Talu M F. SimCLR-based Self-Supervised Learning Approach for Limited Brain MRI and Unlabeled Images [J]. Bitlis Eren Üniversitesi Fen Bilimleri Dergisi, 2024, 13(4): 1304-1313.
[6] Wang C, Chen B, Li G, et al. Automated graph neural network search under federated learning framework [J]. IEEE Transactions on Knowledge and Data Engineering, 2023, 35(10): 9959-9972.
[7] Liu W, Zhang Y, Wang J, et al. Item relationship graph neural networks for e-commerce [J]. IEEE Transactions on Neural Networks and Learning Systems, 2021, 33(9): 4785-4799.
[8] Tu Y, Li W, Song X, et al. Using graph neural network to conduct supplier recommendation based on large-scale supply chain [J]. International Journal of Production Research, 2024, 62(24): 8595-8608.
[9] Zeng L, Yang C, Huang P, et al. GNN at the edge: Cost-efficient graph neural network processing over distributed edge servers [J]. IEEE Journal on Selected Areas in Communications, 2022, 41(3): 720-739.
[10] Wan X, Xu K, Liao X, et al. Scalable and efficient full-graph gnn training for large graphs [J]. Proceedings of the ACM on Management of Data, 2023, 1(2): 1-23.
[11] Miao R, Yang Y, Ma Y, et al. Negative samples selecting strategy for graph contrastive learning [J]. Information Sciences, 2022, 613: 667-681.
[12] Tang D, Wang J, Chen R, et al. Xgnn: Boosting multi-gpu gnn training via global gnn memory store [J]. Proceedings of the VLDB Endowment, 2024, 17(5): 1105-1118.
[13] Wan B, Zhao J, Wu C. Adaptive message quantization and parallelization for distributed full-graph gnn training [J]. Proceedings of Machine Learning and Systems, 2023, 5: 203-218.
[14] Wang X, Yen K, Hu Y, et al. DeepGD: A deep learning framework for graph drawing using GNN [J]. IEEE computer graphics and applications, 2021, 41(5): 32-44.
[15] Gao J, Chen J, Li Z, et al. ICS-GNN: lightweight interactive community search via graph neural network [J]. Proceedings of the VLDB Endowment, 2021, 14(6): 1006-1018.