Staleness-based Subgraph Sampling for Training GNNs on Large-Scale Graphs

Limei Wang[1,2], **Si Zhang**[1], **Hanqing Zeng**[1], **Hao Wu**[1], **Zhigang Hua**[1], **Kaveh Hassani**[1], **Andrey Malevich**[1], **Bo Long**[1], **Shuiwang Ji**[2]
[1]Meta AI, [2]Texas A&M University
{limeiwang, sizhang, zengh, haowu1, zhua, kavehhassani, amalevich, bolong}@meta.com
{limei, sji}@tamu.edu

Abstract

Training Graph Neural Networks (GNNs) on large-scale graphs is challenging. The main difficulty is to obtain accurate node embeddings while avoiding the neighbor explosion problem. One existing solution is using historical embeddings. Specifically, by using historical embeddings for the out-of-batch nodes, these methods can approximate full-batch training without dropping any input data while keeping constant GPU memory consumption. However, it still remains nascent to specifically design a subgraph sampling method that can benefit these historical embedding-based methods. In this paper, we first analyze the approximation error of node embeddings caused by using historical embeddings for out-of-batch neighbors and prove that this approximation error can be minimized by minimizing the staleness of historical embeddings of out-of-batch nodes. Based on the theoretical analysis, we design a simple yet effective Staleness score-based Subgraph Sampling method, called S3, to benefit these historical embedding-based methods. Experimental results show that our S3 sampling method can consistently improve historical embedding-based methods and set the new state-of-the-art without bringing additional computation overhead due to our efficient staleness score calculation, improved re-sampling strategy, and faster training converge.

1 Introduction

Graph neural networks (GNNs) [Kipf and Welling, 2017, Velickovic et al., 2018, Xu et al., 2018, Gasteiger et al., 2018, Corso et al., 2020, Chen et al., 2020] are powerful methods to learn node, edge, and graph representations for various downstream tasks [Hu et al., 2020, 2021, Zhang and Chen, 2018, Gilmer et al., 2017, Wu et al., 2018, Yang et al., 2019]. However, the scalability of GNNs is often limited due to neighbor explosion. Specifically, the size of the node receptive field increases exponentially with respect to the number of layers/hops.

To improve the scalability of GNNs and the efficiency of GNN training, existing studies mainly focus on designing advanced sampling strategies, including node-wise sampling methods [Hamilton et al., 2017, Ying et al., 2018, Huang et al., 2023], layer-wise sampling methods [Chen et al., 2018, Huang et al., 2018, Zou et al., 2019, Balin and Çatalyürek, 2023], and subgraph sampling methods [Zeng et al., 2019, 2021, Chiang et al., 2019]. Given a sampled mini-batch $\mathcal{B}$, the embedding $h_u^{\ell+1}$ for a target node u is obtained by aggregating information from sampled neighbors, as

$$h_u^{\ell+1} = f_\theta^{\ell+1}(h_u^\ell, \{h_v^\ell\}_{v \in \mathcal{N}(u) \cap \mathcal{B}}) \tag{1}$$

where h_u^ℓ is the embedding of node u at layer l, and $\mathcal{N}(u)$ denotes 1-hop neighbors of node u. However, the information from unsampled neighbors is missed.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: New Perspectives in Graph Machine Learning.

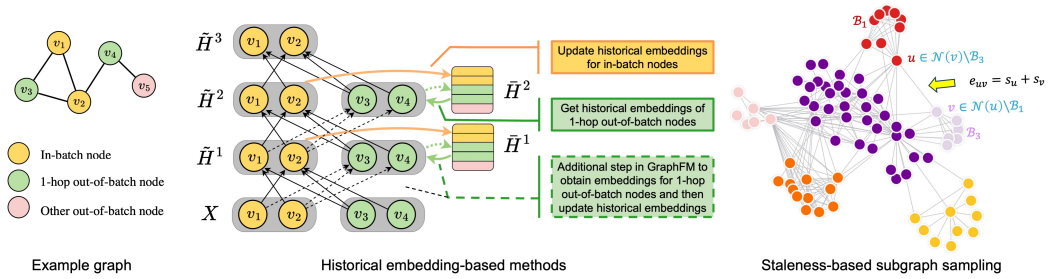

Figure 1: Left: An example graph with in-batch nodes, 1-hop out-of-batch nodes, and other out-of-batch nodes. Middle: Illustrations of historical embedding-based methods. Two main steps are updating historical embeddings for in-batch nodes, and getting historical embeddings for 1-hop out-of-batch nodes. In addition, GraphFM proposes another step to update historical embeddings for 1-hop out-of-batch nodes. Right: Our proposed S3 subgraph sampling. Different from previous methods [Chiang et al., 2019] that sample dense subgraphs by minimizing inter-partition edges, we further define edge weights based on staleness scores. In this way, we can minimize the approximation error of using historical embeddings compare to full-neighborhood propagation embeddings.

To mitigate this problem, several historical embedding-based methods [Chen et al., 2017, Cong et al., 2020, Fey et al., 2021, Yu et al., 2022, Shi et al., 2023] have been proposed to use historical embeddings of the unsampled neighbors as an affordable approximation. Especially, GAS [Fey et al., 2021] provides a reliable framework to use historical embeddings and sets the state-of-the-art in various benchmarks. As shown in GAS, the message passing with historical embeddings can be formulated as

$$\begin{aligned} h_u^{\ell+1} &= f_\theta^{\ell+1}(h_u^\ell, \{h_v^\ell\}_{v\in\mathcal{N}(u)}) \\ &= f_\theta^{\ell+1}(h_u^\ell, \{h_v^\ell\}_{\underbrace{v\in\mathcal{N}(u)\cap\mathcal{B}}_{\text{in-batch neighbors}}} \cup \{h_v^\ell\}_{\underbrace{v\in\mathcal{N}(u)\setminus\mathcal{B}}_{\text{out-of-batch neighbors}}}) \\ &\approx f_\theta^{\ell+1}(h_u^\ell, \{h_v^\ell\}_{v\in\mathcal{N}(u)\cap\mathcal{B}} \cup \underbrace{\{\bar{h}_v^\ell\}_{v\in\mathcal{N}(u)\setminus\mathcal{B}}}_{\text{historical embeddings}}) \end{aligned} \tag{2}$$

where $\bar{h}_v^\ell$ is the corresponding historical embedding of node v at layer ℓ. GAS shows that using historical embeddings can lead to constant GPU memory consumption with respect to input node size without dropping any data. GraphFM [Yu et al., 2022] and LMC [Shi et al., 2023] further enhance GAS by reducing the staleness of historical embeddings from the algorithmic perspective and improving the accuracy of gradient estimation, respectively. Detailed illustrations of GAS and GraphFM are in Fig. 1. However, However, the main challenge is that there is no sampling method specifically designed that can further benefit these historical embedding-based methods.

In this paper, we provide our solution to systematically overcome the challenges and further improve historical embedding-based methods. We first theoretically analyze the impact of the quality of historical embeddings on the output node embeddings and prove that the approximation error of the learned node embeddings can be upper bounded with the staleness of the historical embeddings. Based on the theoretically analysis, we then design a novel subgraph sampling method, called S3, based on staleness scores to explicitly minimize the approximation error of learned node embeddings. To deal with the dynamic changes of staleness scores during training and improve the efficiency of graph partitioning, we design a fast algorithm to avoid re-partitioning the graph from scratch. Our refinement algorithm is 3x faster than graph partitioning from scratch on the large-scale ogbn-products dataset with 2M nodes.

Our work is orthogonal to the existing design of historical embedding-based methods that we focus on subgraph sampling for mini-batch training. Therefore, our method can be applied to all historical embedding-based methods, and preserve their advantages that approximate full-batch training without dropping any data while keeping constant GPU memory consumption. Our S3 shows consistent performance improvement over three historical embedding-based methods (GAS, GraphFM, and LMC). In addition, we show that staleness-based sampling does not bring additional computation

overhead due to efficient staleness score calculation, improved re-sampling strategy, and faster training converge.

2 Method

Historical embeddings has the unique advantage of enabling approximation of full-batch training without dropping any data while keeping constant GPU memory consumption [Fey et al., 2021]. However, the main challenge is that there is no sampling method specifically designed for these historical embedding-based methods. For example, GAS, GraphFM, and LMC [Fey et al., 2021, Yu et al., 2022, Shi et al., 2023] simply apply Cluster-GCN [Chiang et al., 2019] to generate mini-batches practically. In Cluster-GCN, the unweighted graph partitioning algorithm (*e.g.* METIS) [Karypis and Kumar, 1998a, Dhillon et al., 2007] is directly used to convert the input graph into several subgraphs such that the number of inter-partition edges is minimized. However, there is lack of justification on whether Cluster-GCN can generate informative mini-batches to harness historical embeddings. Indeed, in this section, we show that it is not optimal. Specifically, we first analyze the approximation error of the learned node embeddings caused by using historical embeddings for out-of-batch nodes. We prove that the approximation error can be minimized by minimizing the staleness of historical embeddings. We then present our staleness score-based sampling (S3).

2.1 Approximation error analysis

As shown in Equation 2, the main idea of GNNs with historical embeddings is to approximate full-batch embedding $h_u^{\ell+1}$ for each node u by aggregating embeddings $\{h_v^\ell\}_{v\in\mathcal{N}(u)\cap\mathcal{B}}$ for in-batch neighbors and historical embeddings $\{\bar{h}_v^\ell\}_{v\in\mathcal{N}(u)\setminus\mathcal{B}}$ for out-of-batch neighbors. However, there exists an approximation error if $\bar{h}_v^\ell \neq h_v^\ell$ for $v \in \mathcal{N}(u)\setminus\mathcal{B}$. In this section, we analyze how historical embeddings affect the approximation error of the final node embedding $\tilde{h}_u^L$, which motivates the design of our new sampling method.

Theorem 1. *Given a GNN with a linear, graph-dependent aggregation and ReLU activations, the embedding approximation error, i.e., the error between the full-neighborhood propagation embedding h_u^L and the actual aggregated embedding $\tilde{h}_u^L$ by using historical embeddings,*

$$\|h_u^L - \tilde{h}_u^L\|$$

can be minimized by minimizing

$$\sum_{v\in\mathcal{N}(u)\setminus\mathcal{B}} \sum_{\ell=1}^{L-1} C_v^\ell \|h_v^\ell - \bar{h}_v^\ell\|.$$

Here $\sum_{\ell=1}^{L-1} C_v^\ell \|h_v^\ell - \bar{h}_v^\ell\|$ represents the overall quality of the historical embeddings at all $L-1$ layers where C_v^ℓ is a weight that depends on both graph structure and model parameters, and we want to minimize the sum of these terms of all out-of-batch neighbors. The proof for Theorem 1 is provided in Appendix A. Intuitively, based on Equation 2, we can see that the approximation error depends on both the number of out-of-batch neighbors and the quality of the historical embeddings of these out-of-batch neighbors. However, previous methods such as GAS, GraphFM, and LMC only consider the number of out-of-batch nodes and use unweighted graph partitioning algorithms to minimize it. Unfortunately, they do not directly consider the quality of historical embedding which limits their performance.

2.2 Staleness score-based subgraph sampling

Staleness scores of historical embeddings. As discussed in Section 2.1 and Theorem 1, the embedding approximation error can be minimized by minimizing the weighted sum of $\|h_v^\ell - \bar{h}_v^\ell\|$ for all out-of-batch neighbors at all $\ell = 1, \cdots, L-1$ layers. Here $\|h_v^\ell - \bar{h}_v^\ell\|$ is defined as the *staleness score* of the historical embedding of node v at layer ℓ [Fey et al., 2021, Yu et al., 2022], which measures the Euclidean distance between full-neighborhood propagation embedding h_v^ℓ and historical embedding $\bar{h}_v^\ell$. Formally, for each node v, the staleness score s_v^ℓ at layer ℓ is

$$s_v^\ell = \|h_v^\ell - \bar{h}_v^\ell\|. \tag{3}$$

Optimization objective. Based on Theorem 1, to sample a mini-batch $\mathcal{B}$, our optimization objective is

$$\arg\min_{\mathcal{B}} \sum_{u\in\mathcal{B}} \sum_{v\in\mathcal{N}(u)\setminus\mathcal{B}} \sum_{\ell} C_v^\ell s_v^\ell. \tag{4}$$

And we want to minimize the weighted sum of the staleness scores for all out-of-batch neighbors.

In subgraph sampling, we want to convert the input graph $G = (\mathcal{V}, \mathcal{E})$ to M subgraphs $G_1, G_2, ..., G_M$, and each subgraph G_i can be viewed as a mini-batch $\mathcal{B}_i$ during training. Note that the number of nodes in each mini-batch should be (roughly) the same. Considering all M mini-batches, the overall minimization objective becomes

$$\begin{aligned} &\arg\min_{\{\mathcal{B}_1,...,\mathcal{B}_M\}} \sum_{\mathcal{B}_i\in\{\mathcal{B}_1,...,\mathcal{B}_M\}} \sum_{u\in\mathcal{B}_i} \sum_{v\in\mathcal{N}(u)\setminus\mathcal{B}_i} \sum_{\ell} C_v^\ell s_v^\ell \\ &\text{subject to} \quad \mathcal{V} = \mathcal{B}_1 \cup \mathcal{B}_2 \cup ... \cup \mathcal{B}_M \\ &\qquad\qquad \mathcal{B}_i \cap \mathcal{B}_j = \varnothing \quad \text{for all} \quad i \neq j, 1 \leq i, j \leq M \end{aligned} \tag{5}$$

Equivalence to graph partitioning objective. Note that for $u \in \mathcal{B}_i$, $v \in \mathcal{N}(u)\setminus\mathcal{B}_i$ is equivalent to $v \in \mathcal{B}_j$ such that $i \neq j$ and $(u, v) \in \mathcal{E}$. Therefore, the objective in Equation 5 is equivalent to

$$\begin{aligned} &\arg\min_{\{\mathcal{B}_1,...,\mathcal{B}_M\}} \sum_{u\in\mathcal{B}_i, v\in\mathcal{B}_j, i\neq j, (u,v)\in\mathcal{E}} \sum_{\ell} C_u^\ell s_u^\ell + C_v^\ell s_v^\ell \\ &\text{subject to} \quad \mathcal{V} = \mathcal{B}_1 \cup \mathcal{B}_2 \cup ... \cup \mathcal{B}_M \\ &\qquad\qquad \mathcal{B}_i \cap \mathcal{B}_j = \varnothing \quad \text{for all} \quad i \neq j, 1 \leq i, j, \leq M \end{aligned} \tag{6}$$

Then our optimization objective becomes a graph partitioning problem where we want to minimize the edge weight $e_{uv} = \sum_\ell C_u^\ell s_u^\ell + C_v^\ell s_v^\ell$ for all inter-partition edges. Therefore, we can use graph partitioning algorithms to minimize our objective and generate subgraphs (mini-batches) while explicitly minimizing the approximation error of learned node embeddings.

Staleness score-based Subgraph Sampling (S3). Our objective aligns with the Kernighan-Lin objective [Kernighan and Lin, 1970] for graph partitioning problem, where we aim to minimize the total edge weight $e_{uv} = \sum_\ell C_u^\ell s_u^\ell + C_v^\ell s_v^\ell$ for all inter-partition edges. Nevertheless, it is often impractical to use the exact $\sum_\ell C_u^\ell s_u^\ell + C_v^\ell s_v^\ell$ as the edge weight since the computations of C_u^ℓ, C_v^ℓ involve a lot of path-dependent factors as shown in Appendix A. Alternatively, we drop them to simplify the computations, i.e., $e_{uv} = \sum_\ell s_u^\ell + s_v^\ell$. Meanwhile, multi-level approaches proposed by Karypis and Kumar [1997], Dhillon et al. [2007] have been widely employed to efficiently solve the graph partitioning tasks based on similar objectives. In this way, our S3 sampling works as follows. We first define the weight of each edge (u, v) as the sum of the staleness scores of the source and target nodes. Then we apply multi-level graph partitioning to generate mini-batches. In this way, we can explicitly reduce the approximation error of learned node embeddings.

2.3 Efficient staleness score calculation

Calculating staleness score requires both historical embeddings and full-neighborhood propagation embedding. To reduce the cost and avoid out-of-memory issue for full-neighborhood propagation, we employ a layer-wise mini-batch inference following GAS. Specifically, we iterate over the mini-batch loader in a layer-wise fashion. For each individual layer and mini-batch, we do a forward pass to compute the aggregated node embeddings. In order to re-use some intermediate representations, we maintain a dictionary for each individual mini-batch. In this way, we can safely obtain full-neighborhood propagation during inference, even for very large-scale graphs. Note that this technique can only be use during inference to reduce the cost and avoid out-of-memory issue, because during training, we have to calculate gradient.

2.4 Re-sampling via fast refinement

To reduce cost, we further propose an improved re-sampling strategy to reduce the frequency of staleness score calculation and graph partitioning time. Specifically, in our S3 sampling method, the edge weight $e_{uv} = \sum_\ell s_u^\ell + s_v^\ell$ evolves dynamically throughout the training process because of the continual updates applied to both historical embeddings and learnable parameters during training. To

Table 1: Comparison between our sampling method and other baseline methods. We apply our S3 sampling method to the three popular and powerful historical embedding-based methods, GAS [Fey et al., 2021], GraphFM [Yu et al., 2022], and LMC [Shi et al., 2023]. We report the mean and standard deviation over five random runs. The three different background colors, gray, pink, and yellow, correspond to the three baseline methods. – indicates there is no reported value. Note that for Yelp, GCN is much worse than other models, therefore, we do not report this. Red indicates that our sampling method can improve the corresponding baselines with their default sampling.

# Nodes		89K	230K	717K	169K	2.4M
# Edges		450K	11.6M	7.9M	1.2M	61.9M
Method		Flickr	Reddit	Yelp	ogbn-arxiv	ogbn-products
VR-GCN		0.482 ± 0.003	0.964 ± 0.001	0.640 ± 0.002	–	–
FastGCN		0.504 ± 0.001	0.924 ± 0.001	0.265 ± 0.053	–	–
GraphSAINT		0.511 ± 0.001	0.966 ± 0.001	0.653 ± 0.003	–	0.791 ± 0.002
Cluster-GCN		0.481 ± 0.005	0.954 ± 0.001	0.609 ± 0.005	–	0.790 ± 0.003
SIGN		0.514 ± 0.001	0.968 ± 0.000	0.631 ± 0.003	0.720 ± 0.001	0.776 ± 0.001
GraphSAGE		0.501 ± 0.013	0.953 ± 0.001	0.634 ± 0.006	0.715 ± 0.003	0.783 ± 0.002
GCN	GAS	0.534 ± 0.001	0.954 ± 0.000	–	0.715 ± 0.002	0.767 ± 0.002
	S3 + GAS	0.545 ± 0.001	0.955 ± 0.000	–	0.724 ± 0.002	0.771 ± 0.002
GCNII	GAS	0.554 ± 0.003	0.967 ± 0.000	0.639 ± 0.003	0.725 ± 0.003	0.770 ± 0.002
	S3 + GAS	0.567 ± 0.002	0.969 ± 0.001	0.652 ± 0.003	0.735 ± 0.002	0.778 ± 0.002
GCN	FM	0.535 ± 0.002	0.953 ± 0.000	–	0.715 ± 0.003	0.767 ± 0.001
	S3 + FM	0.549 ± 0.001	0.952 ± 0.000	–	0.722 ± 0.002	0.770 ± 0.002
GCNII	FM	0.547 ± 0.003	0.965 ± 0.001	0.641 ± 0.003	0.725 ± 0.003	0.771 ± 0.002
	S3 + FM	0.566 ± 0.003	0.969 ± 0.000	0.652 ± 0.002	0.733 ± 0.003	0.776 ± 0.001
GCN	LMC	0.538 ± 0.001	0.954 ± 0.000	–	0.714 ± 0.002	0.765 ± 0.002
	S3 + LMC	0.541 ± 0.001	0.955 ± 0.000	–	0.721 ± 0.002	0.770 ± 0.002
GCNII	LMC	0.554 ± 0.005	0.969 ± 0.000	0.647 ± 0.003	0.728 ± 0.002	0.769 ± 0.002
	S3 + LMC	0.562 ± 0.002	0.969 ± 0.000	0.650 ± 0.003	0.731 ± 0.001	0.773 ± 0.002

deal with the dynamic changes of staleness scores and improve the efficiency of graph partitioning, we further design a fast algorithm to avoid re-partitioning the graph from scratch. In addition, we carefully design the re-sampling scheduler (frequency) based on empirical observations.

Re-sampling scheduler. To deal with the graph partitioning for dynamic graphs, one of the key factors is the partitioning frequency (scheduler). This hyperparameter plays a pivotal role in dictating when re-partitioning should be initiated, holding significant implications for the overall time complexity of the partitioning process. Practically, we find that conducting re-partitioning after a fixed number of epochs (*e.g.* 20 epochs) consistently yields favorable results without imposing significant time overhead. Detailed empirical analysis on the frequency is included in Section 3.2.

Efficient refinement. In addition, instead of re-partitioning the graph from scratch, we use k-way Kernighan–Lin refinement algorithm [Karypis and Kumar, 1998b] to do refinement, which is much more efficient. Specifically, at the t-th training epoch, instead of directly partitioning the graph G^t into $G_1^t, G_2^t, ..., G_M^t$, we perform refinement on the partitioning result $G_1^{t-1}, G_2^{t-1}, ..., G_M^{t-1}$ at the previous epoch. The refinement is based on the gain, i.e., the reduction in the edge cuts, by moving nodes to other mini-batches. Formally, for a node $u \in \mathcal{B}_i$, the potential gain of moving it from subgraph G_i to G_j is

$$\begin{aligned} g(u)_j &= EW(u)_j - IW(u) \\ &= \sum_{v \in \mathcal{N}(u) \cap \mathcal{B}_j} e_{uv} - \sum_{v \in \mathcal{N}(u) \cap \mathcal{B}_i} e_{uv}. \end{aligned} \tag{7}$$

Node u is moved to G_k such that $k = \arg\max_j g(u)_j$.

3 Experiments

3.1 Main empirical results

Since our sampling method is specially designed for historical embedding-based methods, we select three most recent and powerful historical embedding-based backbone methods (GAS [Fey et al., 2021], GraphFM [Yu et al., 2022], and LMC [Shi et al., 2023]) to show the improvement of our S3 sampling. We rerun some baseline method multiple times to get means and standard deviation if not

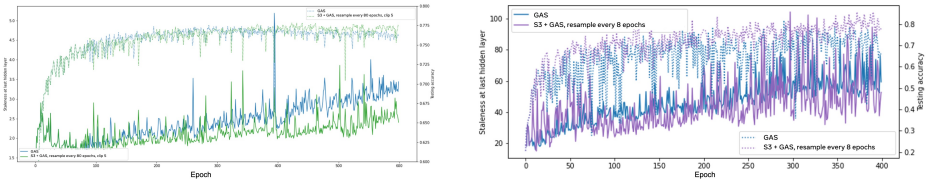

Figure 2: Comparison between GAS (blue) and S3 + GAS (green and purple) on ogbn-products and Reddit datasets in terms of staleness scores of historical embeddings (solid line) and testing accuracy (dashed line). Results show that our S3 sampling leads to improved quality of historical embeddings (lower staleness score) and better accuracy.

provided in original paper. The comparison with baseline methods is shown in Table 1. Results show that our new sampling method can improve the performance of all three historical embedding-based methods (GAS, GraphFM, and LMC) on almost all datasets. The consistent improvements indicate the great effectiveness and generalizability of our S3 sampling method. In detail, we achieve more than 1% improvement on most datasets.

Specifically, as shown in Fig. 2, our S3 + GAS leads to improved quality of historical embeddings (lower staleness score) and better accuracy, especially for the largest dataset we used, ogbn-products with more than 2M nodes.

3.2 Efficiency analysis

Compared to baseline methods, our S3 requires additional steps to compute staleness score and do re-sampling. Here we show that we do not bring additional computational overhead due to our efficient staleness score calculation and improved re-sampling strategy, as in Table 2. Specifically, we show that the time for staleness score calculation is much smaller than the training time per epoch. And our fast refinement can significantly reduce the time for graph partitioning, especially for the large-scale graphs, e.g. ogbn-products with more than 2M nodes. Note that in the next subsection, we also show that doing re-sampling every fixed number of epochs (*e.g.* 20 epochs) already consistently yields favorable results, which further support that our method does not bring additional overhead. In addition, with out designed subgraph sampling, the training converge faster, as in Table 3.

Table 2: Training, staleness score calculation, and re-sampling time.

Running time (s)	ogbn-arxiv	ogbn-products
Training per epoch	1	37
Staleness score calculation	0	3
Re-sampling from scratch	3	120
Re-sampling with fast refinement	2	48

Table 3: Number of epochs and total running time. Here we only list the number with LMC, because among the three historical embedding-based methods, LMC has the shortest running time. This is because LMC retrieves the discarded messages in backward passes, leading to accurate mini-batch gradients and accelerating convergence.

		Flickr & GCNII	ogbn-arxiv & GCNII
Epochs	LMC	356	197.4
	S3 + LMC	**211.4**	**180**
Runtime (s)	LMC	475	178
	S3 + LMC	**304**	**175**

4 Conclusion

We focus on the task of training GNNs on large-scale graphs, where the main challenge is the neighbor explosion problem. Many of the existing methods use historical embeddings to solve this problem. In this paper, we provide a simple yet effective sampling method called S3 to further benefit historical embedding-based methods. Our S3 is build based on the theoretical analysis of approximation error caused by using historical embeddings. Experimental results show that our sampling method can

further improve existing historical embedding-based methods and set new state-of-the-art on various datasets without bring additional computation overhead.

References

Muhammed Fatih Balin and Ümit Çatalyürek. Layer-neighbor sampling — defusing neighborhood explosion in gnns. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Jianfei Chen, Jun Zhu, and Le Song. Stochastic training of graph convolutional networks with variance reduction. *arXiv preprint arXiv:1710.10568*, 2017.

Jie Chen, Tengfei Ma, and Cao Xiao. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations*, 2018.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1725–1735. PMLR, 2020.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui 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 & data mining*, pages 257–266, 2019.

Anna Choromanska, Yann LeCun, and Gérard Ben Arous. Open problem: The landscape of the loss surfaces of multilayer networks. In *Conference on Learning Theory*, pages 1756–1760. PMLR, 2015.

Weilin Cong, Rana Forsati, Mahmut Kandemir, and Mehrdad Mahdavi. Minimal variance sampling with provable guarantees for fast training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1393–1403, 2020.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33:13260–13271, 2020.

Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE transactions on pattern analysis and machine intelligence*, 29(11): 1944–1957, 2007.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Matthias Fey, Jan E Lenssen, Frank Weichert, and Jure Leskovec. GNNAutoScale: Scalable and expressive graph neural networks via historical embeddings. In *International Conference on Machine Learning*, pages 3294–3304. PMLR, 2021.

Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. SIGN: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198*, 2020.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

Johannes Gasteiger, Chendi Qian, and Stephan Günnemann. Influence-based mini-batching for graph neural networks. In *Learning on Graphs Conference*, pages 9–1. PMLR, 2022.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. OGB-LSC: A large-scale challenge for machine learning on graphs. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Kezhao Huang, Haitian Jiang, Minjie Wang, Guangxuan Xiao, David Wipf, Xiang Song, Quan Gan, Zengfeng Huang, Jidong Zhai, and Zheng Zhang. Refresh: Reducing memory access from exploiting stable historical embeddings for graph neural network training. *arXiv preprint arXiv:2301.07482*, 2023.

Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. Adaptive sampling towards fast graph representation learning. *Advances in neural information processing systems*, 31, 2018.

George Karypis and Vipin Kumar. Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. 1997.

George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998a.

George Karypis and Vipin Kumar. Multilevelk-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed computing*, 48(1):96–129, 1998b.

Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307, 1970.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

Zhihao Shi, Xize Liang, and Jie Wang. LMC: Fast training of GNNs via subgraph sampling with provable convergence. In *The Eleventh International Conference on Learning Representations*, 2023.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, et al. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8): 3370–3388, 2019.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.

Haiyang Yu, Limei Wang, Bokun Wang, Meng Liu, Tianbao Yang, and Shuiwang Ji. Graphfm: Improving large-scale gnn training via feature momentum. In *International Conference on Machine Learning*, pages 25684–25701. PMLR, 2022.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graph-SAINT: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

Hanqing Zeng, Muhan Zhang, Yinglong Xia, Ajitesh Srivastava, Andrey Malevich, Rajgopal Kannan, Viktor Prasanna, Long Jin, and Ren Chen. Decoupling the depth and scope of graph neural networks. *Advances in Neural Information Processing Systems*, 34:19665–19679, 2021.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. Layer-dependent importance sampling for training deep and large graph convolutional networks. *Advances in neural information processing systems*, 32, 2019.

A Proof of Theorem 1

Path-based view of GNNs. We can view a graph neural network with ReLUs as a directed acyclic computational graph and express the i-th output logit of node u via paths through this graph [Gasteiger et al., 2022] as

$$h_{u,i}^{L} = C \sum_{v \in \mathcal{N}_{all}^{L}(u)} \sum_{p=1}^{\psi} \sum_{q=1}^{\phi} z_{v,p,i,q} x_{v,p,i,q} \prod_{\ell=1}^{L} a_{v,p}^{\ell} w_{i,q}^{\ell},$$

where C is a constant related to the size of the network [Choromanska et al., 2015], $\mathcal{N}_{all}^{L}(u)$ includes all nodes within L-hop of node u, ψ is the total number of graph-based paths, ϕ is the total number of paths in learnable weights, $z_{v,p,i,q} \in \{0,1\}$ denotes whether the path is active or inactive when any ReLU is deactivated, $x_{v,p,i,q}$ is the input feature used in the path, $a_{v,p}^{\ell}$ denotes the graph-dependent but feature-independent aggregation weight, and $w_{i,q}^{\ell}$ represents the used entry of the weight matrix W_ℓ at layer ℓ.

Aggregated embedding $\tilde{h}_u^L$ by using historical embeddings. In the historical embedding-based methods [Fey et al., 2021], the aggregated feature $\tilde{h}_u^L$ of node u is based on the input features x_v of in-batch nodes $\mathcal{N}_{in}^{L}(u)$ within L-hop of node u, input features x_v of 1-hop out-of-batch nodes $\mathcal{N}_{out}^{1}(u)$, and the historical embeddings $\bar{h}_v$ of 1-hop out-of-batch nodes $\mathcal{N}_{out}^{1}(u)$, denoted as

$$\begin{aligned}
\tilde{h}_{u,i}^{L} = \quad & C \sum_{v \in \mathcal{N}_{in}^{L}(u)} \sum_{p=1}^{\psi_0^{in}} \sum_{q=1}^{\phi_0^{in}} z_{v,p,i,q} x_{v,p,i,q} \prod_{\ell=1}^{L} a_{v,p}^{\ell} w_{i,q}^{\ell} \\
& + C \sum_{v \in \mathcal{N}_{out}^{1}(u)} \sum_{p=1}^{\psi_{L-1}^{out}} \sum_{q=1}^{\phi_{L-1}^{out}} z_{v,p,i,q} \bar{h}_{v,p,i,q}^{L-1} a_{v,p}^{L} w_{i,q}^{L} \\
& + C \sum_{v \in \mathcal{N}_{out}^{1}(u)} \sum_{p=1}^{\psi_{L-2}^{out}} \sum_{q=1}^{\phi_{L-2}^{out}} z_{v,p,i,q} \bar{h}_{v,p,i,q}^{L-2} \\
& \prod_{\ell=L-1}^{L} a_{v,p}^{\ell} w_{i,q}^{\ell} \\
& + \dots \\
& + C \sum_{v \in \mathcal{N}_{out}^{1}(u)} \sum_{p=1}^{\psi_1^{out}} \sum_{q=1}^{\phi_1^{out}} z_{v,p,i,q} \bar{h}_{v,p,i,q}^{1} \prod_{\ell=2}^{L} a_{v,p}^{\ell} w_{i,q}^{\ell} \\
& + C \sum_{v \in \mathcal{N}_{out}^{1}(u)} \sum_{p=1}^{\psi_0^{out}} \sum_{q=1}^{\phi_0^{out}} z_{v,p,i,q} x_{v,p,i,q} \prod_{\ell=1}^{L} a_{v,p}^{\ell} w_{i,q}^{\ell}.
\end{aligned}$$

Note that $\mathcal{N}^1_{out}(u)$ is equivalent to $\mathcal{N}(u)\backslash\mathcal{B}$ in the main text. In the appendix, we use $\mathcal{N}^1_{out}(u)$ to simplify the notation.

Full-neighborhood propagation embedding h^L_u. Based on the path-based view of GNNs, the full-neighborhood propagation embedding h^L_u can be formulated as

$$
\begin{aligned}
h^L_{u,i} = \quad & C \sum_{v\in\mathcal{N}^L_{all}(u)} \sum_{p=1}^{\psi}\sum_{q=1}^{\phi} z_{v,p,i,q} x_{v,p,i,q} \prod_{\ell=1}^{L} a^{\ell}_{v,p} w^{\ell}_{i,q} \\
= \quad & C \sum_{v\in\mathcal{N}^1_{in}(u)} \sum_{p=1}^{\psi^{in}_{L-1}}\sum_{q=1}^{\phi^{in}_{L-1}} z_{v,p,i,q} h^{L-1}_{v,p,i,q} a^{L}_{v,p} w^{L}_{i,q} \\
& + C \sum_{v\in\mathcal{N}^1_{out}(u)} \sum_{p=1}^{\psi^{out}_{L-1}}\sum_{q=1}^{\phi^{out}_{L-1}} z_{v,p,i,q} h^{L-1}_{v,p,i,q} a^{L}_{v,p} w^{L}_{i,q} \\
= \quad & C \sum_{v\in\mathcal{N}^2_{in}(u)} \sum_{p=1}^{\psi^{in}_{L-2}}\sum_{q=1}^{\phi^{in}_{L-2}} z_{v,p,i,q} h^{L-2}_{v,p,i,q} \prod_{\ell=L-1}^{L} a^{\ell}_{v,p} w^{\ell}_{i,q} \\
& + C \sum_{v\in\mathcal{N}^1_{out}(u)} \sum_{p=1}^{\psi^{out}_{L-2}}\sum_{q=1}^{\phi^{out}_{L-2}} z_{v,p,i,q} h^{L-2}_{v,p,i,q} \prod_{\ell=L-1}^{L} a^{\ell}_{v,p} w^{\ell}_{i,q} \\
& + C \sum_{v\in\mathcal{N}^1_{out}(u)} \sum_{p=1}^{\psi^{out}_{L-1}}\sum_{q=1}^{\phi^{out}_{L-1}} z_{v,p,i,q} h^{L-1}_{v,p,i,q} a^{L}_{v,p} w^{L}_{i,q} \\
= \quad & C \sum_{v\in\mathcal{N}^L_{in}(u)} \sum_{p=1}^{\psi^{in}_{0}}\sum_{q=1}^{\phi^{in}_{0}} z_{v,p,i,q} x_{v,p,i,q} \prod_{\ell=1}^{L} a^{\ell}_{v,p} w^{\ell}_{i,q} \\
& + C \sum_{v\in\mathcal{N}^1_{out}(u)} \sum_{p=1}^{\psi^{out}_{0}}\sum_{q=1}^{\phi^{out}_{0}} z_{v,p,i,q} x_{v,p,i,q} \prod_{\ell=1}^{L} a^{\ell}_{v,p} w^{\ell}_{i,q} \\
& + C \sum_{v\in\mathcal{N}^1_{out}(u)} \sum_{p=1}^{\psi^{out}_{1}}\sum_{q=1}^{\phi^{out}_{1}} z_{v,p,i,q} h^{1}_{v,p,i,q} \prod_{\ell=2}^{L} a^{\ell}_{v,p} w^{\ell}_{i,q} \\
& + \dots \\
& + C \sum_{v\in\mathcal{N}^1_{out}(u)} \sum_{p=1}^{\psi^{out}_{L-2}}\sum_{q=1}^{\phi^{out}_{L-2}} z_{v,p,i,q} h^{L-2}_{v,p,i,q} \prod_{\ell=L-1}^{L} a^{\ell}_{v,p} w^{\ell}_{i,q} \\
& + C \sum_{v\in\mathcal{N}^1_{out}(u)} \sum_{p=1}^{\psi^{out}_{L-1}}\sum_{q=1}^{\phi^{out}_{L-1}} z_{v,p,i,q} h^{L-1}_{v,p,i,q} a^{L}_{v,p} w^{L}_{i,q}.
\end{aligned}
$$

Approximation error. The difference $h^L_{u,i} - \tilde{h}^L_{u,i}$ between the full-neighborhood propagation embedding$h^L_{u,i}$ and the actual aggregated embedding $\tilde{h}^L_{u,i}$ is

$$
\begin{aligned}
& C \sum_{v\in\mathcal{N}^1_{out}(u)} \sum_{p=1}^{\psi^{out}_{L-1}}\sum_{q=1}^{\phi^{out}_{L-1}} z_{v,p,i,q} (h^{L-1}_{v,p,i,q} - \bar{h}^{L-1}_{v,p,i,q}) a^{L}_{v,p} w^{L}_{i,q} \\
& + C \sum_{v\in\mathcal{N}^1_{out}(u)} \sum_{p=1}^{\psi^{out}_{L-2}}\sum_{q=1}^{\phi^{out}_{L-2}} z_{v,p,i,q} (h^{L-2}_{v,p,i,q} - \bar{h}^{L-2}_{v,p,i,q}) \prod_{\ell=L-1}^{L} a^{\ell}_{v,p} w^{\ell}_{i,q} \\
& + \dots
\end{aligned}
$$

$$+ C \sum_{v \in \mathcal{N}^1_{out}(u)} \sum_{p=1}^{\psi_1^{out}} \sum_{q=1}^{\phi_1^{out}} z_{v,p,i,q} (h^1_{v,p,i,q} - \bar{h}^1_{v,p,i,q}) \prod_{\ell=2}^{L} a^{\ell}_{v,p} w^{\ell}_{i,q}.$$

Then squaring both sides of the equation, we have the following inequality since the sum of the squares is always less than or equal to the square of the sums.

$$\begin{aligned}
&(h^L_{u,i} - \tilde{h}^L_{u,i})^2 \leq \\
&C^2 \sum_{v \in \mathcal{N}^1_{out}(u)} \sum_{p=1}^{\psi_{L-1}^{out}} \sum_{q=1}^{\phi_{L-1}^{out}} z_{v,p,i,q} (h^{L-1}_{v,p,i,q} - \bar{h}^{L-1}_{v,p,i,q})^2 (a^L_{v,p} w^L_{i,q})^2 \\
&+ C^2 \sum_{v \in \mathcal{N}^1_{out}(u)} \sum_{p=1}^{\psi_{L-1}^{out}} \sum_{q=1}^{\phi_{L-2}^{out}} z_{v,p,i,q} (h^{L-2}_{v,p,i,q} - \bar{h}^{L-2}_{v,p,i,q})^2 \prod_{\ell=L-1}^{L} (a^{\ell}_{v,p} w^{\ell}_{i,q})^2 \\
&+ \dots \\
&+ C^2 \sum_{v \in \mathcal{N}^1_{out}(u)} \sum_{p=1}^{\psi_1^{out}} \sum_{q=1}^{\phi_1^{out}} z_{v,p,i,q} (h^1_{v,p,i,q} - \bar{h}^1_{v,p,i,q})^2 \prod_{\ell=2}^{L} (a^{\ell}_{v,p} w^{\ell}_{i,q})^2.
\end{aligned}$$

Therefore, the approximation error can be formulated as

$$\begin{aligned}
&\|h^L_u - \tilde{h}^L_u\|_2^2 \\
&= \sum_i (h^L_{u,i} - \tilde{h}^L_{u,i})^2 \\
&\leq C^2 \sum_{v \in \mathcal{N}^1_{out}(u)} \sum_{p=1}^{\psi_{L-1}^{out}} \sum_{q=1}^{\phi_{L-1}^{out}} \sum_i z_{v,p,i,q} (h^{L-1}_{v,p,i,q} - \bar{h}^{L-1}_{v,p,i,q})^2 (a^L_{v,p} w^L_{i,q})^2 \\
&+ C^2 \sum_{v \in \mathcal{N}^1_{out}(u)} \sum_{p=1}^{\psi_{L-1}^{out}} \sum_{q=1}^{\phi_{L-2}^{out}} \sum_i z_{v,p,i,q} (h^{L-2}_{v,p,i,q} - \bar{h}^{L-2}_{v,p,i,q})^2 \\
&\prod_{\ell=L-1}^{L} (a^{\ell}_{v,p} w^{\ell}_{i,q})^2 \\
&+ \dots \\
&+ C^2 \sum_{v \in \mathcal{N}^1_{out}(u)} \sum_{p=1}^{\psi_1^{out}} \sum_{q=1}^{\phi_1^{out}} \sum_i z_{v,p,i,q} (h^1_{v,p,i,q} - \bar{h}^1_{v,p,i,q})^2 \\
&\prod_{\ell=2}^{L} (a^{\ell}_{v,p} w^{\ell}_{i,q})^2 \\
&\leq \sum_{v \in \mathcal{N}^1_{out}(u)} C_v^{L-1} \|h^{L-1}_v - \bar{h}^{L-1}_v\|_2^2 \\
&+ \sum_{v \in \mathcal{N}^1_{out}(u)} C_v^{L-2} \|h^{L-2}_v - \bar{h}^{L-2}_v\|_2^2 \\
&+ \dots \\
&+ \sum_{v \in \mathcal{N}^1_{out}(u)} C_v^{1} \|h^{1}_v - \bar{h}^{1}_v\|_2^2 \\
&= \sum_{v \in \mathcal{N}^1_{out}(u)} \sum_{\ell=1}^{L-1} C_v^{\ell} \|h^{\ell}_v - \bar{h}^{\ell}_v\|_2^2.
\end{aligned}$$

Here C_v^ℓ is a weight that depends on both the graph structure and model parameters. Finally, we have

$$\|h_u^L - \tilde{h}_u^L\| \leq \sum_{\ell=1}^{L-1} C_v^\ell \|h_v^\ell - \bar{h}_v^\ell\|.$$

The approximation error, the error (Euclidean distance) between the full-neighborhood propagation embedding h_u^L and the actual aggregated embedding $\tilde{h}_u^L$, can be upper bounded by $\sum_{\ell=1}^{L-1} C_v^\ell \|h_v^\ell - \bar{h}_v^\ell\|$. Therefore, we can minimize the approximation error by minimizing $\sum_{\ell=1}^{L-1} C_v^\ell \|h_v^\ell - \bar{h}_v^\ell\|$.

B Training details and ablation study

Table 4: Statistics of the datasets. Here "m" indicates the multi-label classification task, and "s" indicates the single-label classification task.

Dataset	# of nodes	# of edges	Avg. degree	# of features	# of classes	Train/Val/Test
Flickr	89,250	899,756	10.0813	500	7(s)	0.500/0.250/0.250
Reddit	232,965	11,606,919	49.8226	602	41(s)	0.660/0.100/0.240
Yelp	716,847	6,997,410	9.7614	300	50(m)	0.750/0.150/0.100
ogbn-arxiv	169,343	1,166,243	6.8869	128	40(s)	0.537/0.176/0.287
ogbn-products	2,449,029	61,859,140	25.2586	100	47(s)	0.100/0.020/0.880

Datasets and baselines. Following previous studies, we evaluate our S3 subgraph sampling on 5 large-scale datasets, including Flickr [Zeng et al., 2019], Reddit [Hamilton et al., 2017], Yelp [Zeng et al., 2019], ogbn-arxiv [Hu et al., 2021] and ogbn-products Hu et al. [2021]. Detailed descriptions of the datasets are provided in Table 4. We consider node-wise, layer-wise, and subgraph sampling methods, historical embedding-based methods, and pre-computing methods, as baselines, including VR-GCN [Chen et al., 2017], FastGCN [Chen et al., 2018], GraphSAINT [Zeng et al., 2019], Cluster-GCN [Chiang et al., 2019], SIGN [Frasca et al., 2020], GraphSAGE [Hamilton et al., 2017], GAS [Fey et al., 2021], GraphFM [Yu et al., 2022], and LMC [Shi et al., 2023]. Results for baseline methods are directly taken from their original papers.

Software, hardware, and license. The implementation of our sampling method is based on PyTorch [Paszke et al., 2019] (license: https://github.com/pytorch/pytorch/blob/main/LICENSE), PyGAS [Fey et al., 2021] (MIT License), PyTorch Sparse [Fey and Lenssen, 2019] (MIT License), PyG (PyTorch Geometric) [Fey and Lenssen, 2019] (MIT License), and METIS [Karypis and Kumar, 1997] (Apache License). For a fair comparison with GAS (MIT License), GraphFM (GPL-3.0 license), and LMC, we follow their official code and only replace their sampling method with ours. All experiments are conducted on one Nvidia GeForce RTX 2080 GPU.

B.1 Ablation Study

In our sampling method, we first define the edge weight as the sum of the staleness scores of the two nodes. Then we partition this weighted graph into M subgraphs, corresponding to M mini-batches, by minimizing the total edge weights of inter-partition edges. This definition of edge weight and the associated minimization objective draw inspiration from our theoretical analysis, which aims to minimize the approximation error arising from the use of historical embeddings. In this section, we underscore the significance of our defined edge weights and minimization objective by comparing it with different baseline approaches. The comparison results are detailed in Table 5.

Specifically, the "random node sampler" randomly partitions the input graph, whereas the other three methods partition the input graph by minimizing inter-partition edges. "No edge weight" implies that we disregard the staleness scores during sampling, employing a graph partitioning algorithm solely to generate mini-batches. This is exactly the sampling method used in Cluster-GCN, GAS, GraphFM, and LMC. "Random edge weight" assigns random values to the edge weights. Meanwhile, "staleness score-based edge weight" is the approach introduced in this paper. Table 5 demonstrates that our solution surpasses all the variants, highlighting the effectiveness of our method with its optimal optimization objective.

In addition, as discussed in Section 2.2, the edge weight $e_{uv} = \sum_{\ell=1}^{L-1} s_u^\ell + s_v^\ell$ is defined as the sum of the staleness scores of both source and target nodes over all layers. But practically, we find that it is sufficient to only use the staleness scores at layer $L-1$ and set $e_{uv} = s_u^{L-1} + s_v^{L-1}$ to achieve

Table 5: Ablation study on the optimization objective in our S3. Specifically, in S3, we consider staleness scores as edge weight and aim to minimize the total edge weights of inter-partition edges. All results are based on the GAS baseline.

Method	Flickr & PNA	ogbn-arxiv & PNA
Random node sampling	0.5521	0.7104
No edge weight	0.5667	0.7250
Random edge weight	0.5671	0.7228
Staleness score-based edge weight (ours)	**0.5729**	**0.7303**

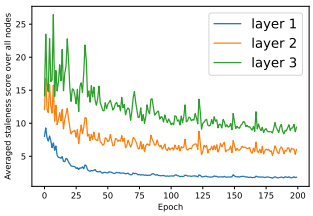

Figure 3: Staleness scores at different layers.

similar results. Therefore, in our final experiments, we only use the staleness score of layer $L-1$ in our S3 sampling method. We believe this is because the staleness score increases monotonically across the layers due to error accumulation, as shown in Fig. 3 (ogbn-arxiv& GCNII). Therefore, only using the staleness scores at layer $L-1$ can also minimize our optimization objective.

During training, we observe some extreme staleness scores which can lead to poor performance. Therefore, we normalize the staleness scores before performing our S3 sampling. Specifically, for staleness scores larger than a fixed value x, we set them to x. We choose x as the value that is greater than 90% of the staleness scores for all nodes.

C Limitations and discussions

As discussed in the paper, our sampling method is designed for and built on top of historical embedding-based methods, which has it own limitation. Specifically, the advantage of historical embedding-based methods is approximating full-batch training without dropping any data while keeping constant GPU memory consumption. However, they require additional (CPU) memory to store the historical embeddings for each node in each layer. Hash-based solutions can be applied to reduce the memory cost.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The introduction and introduction summarize the main contribution of this paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Included in the Appendix C.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Included in Appendix A.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Included in Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We only used public dataset and include training details in Appendix B.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: In Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report mean and standard deviation over five random run.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Included in Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We follow the rule and ensure to preserve anonymity.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This work is designing a subgraph sampling method for GNNs training, there is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Included in Appendix B.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.