SGS-GNN: A Supervised Graph Sparsifier for Graph Neural Networks

Siddhartha Shankar Das
siddhartha.das@pnnl.gov
Pacific Northwest National Laboratory
Richland, WA, USA

Naheed Anjum Arafat
Naheedanjum.Arafat@howard.edu
DoD Center of Excellence in AI/ML (CoE-AIML), Howard University
Washington, DC, USA

Muftiqur Rahman
muftiqurrahman@iut-dhaka.edu
Islamic University of Technology
Dhaka, Bangladesh

S M Ferdous
sm.ferdous@pnnl.gov
Pacific Northwest National Laboratory
Richland, WA, USA

Alex Pothen
apothen@purdue.edu
Purdue University
West Lafayette, IN, USA

Mahantesh M Halappanavar
hala@pnnl.gov
Pacific Northwest National Laboratory
Richland, WA, USA

Danda B. Rawat
danda.rawat@howard.edu
DoD Center of Excellence in AI/ML (CoE-AIML), Howard University
Washington, DC, USA

Abstract

We propose SGS-GNN, a supervised graph sparsifier for Graph Neural Networks (GNNs) to improve predictive performance and reduce the cost of message passing by removing task-irrelevant edges. Existing unsupervised sparsifiers are not task-aware, while existing supervised sparsifiers suffer from significant memory overhead, poor sparsity control, and a lack of homophily/heterophily awareness. SGS-GNN addresses these limitations by adopting a feature- and structure-aware edge-probability encoder, a sparse subgraph sampler that strictly adheres to a global sparsity constraint, and a homophily-aware regularizer to improve prediction accuracy across homophilic and heterophilic graphs. A key scalability-enhancing feature of SGS-GNN is that it ensures encoder updates are computed by backpropagating through the sampled subgraph, and avoids retaining edge-level computation graphs for all edges via gradient checkpointing. A key efficiency-enhancing feature of SGS-GNN is that the edge-probability encoder is updated only when it outperforms a degree-based edge sampler, ensuring performance no worse than a strong unsupervised baseline. Experiments on 33 homophilic and heterophilic graphs show that SGS-GNN improves F1-scores by 4% relative to full training and up to 30% on heterophilic graphs. Furthermore, SGS-GNN outperforms state-of-the-art methods by 4–7% at similar sparsity levels while reducing peak memory usage by up to 3.9×.

CCS Concepts

• **Computing methodologies** → **Supervised learning**; • **Theory of computation** → **Sparsification and spanners**.

Keywords

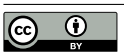
Graph Sparsification, Graph Neural Network

ACM Reference Format:
Siddhartha Shankar Das, Naheed Anjum Arafat, Muftiqur Rahman, S M Ferdous, Alex Pothen, Mahantesh M Halappanavar, and Danda B. Rawat. 2026. SGS-GNN: A Supervised Graph Sparsifier for Graph Neural Networks. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD 2026), August 9–13, 2026, Jeju Island, Republic of Korea.* ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/3770855.3818110

1 Introduction

Graph Neural Networks (GNNs) [44, 52] are tools for learning from graph-structured data and are widely used in social networks, biological networks, computational physics, recommendation systems, and many more. The two primary steps in a GNN are *aggregation* and *update*. For each node, the aggregation step involves accumulating embeddings from neighboring nodes using sparse matrix operations such as sparse-matrix dense-matrix multiplication (SpMM) and sampled dense-matrix dense-matrix multiplication (SDDMM) [50]. During the update step, a node updates its own embedding based on the aggregated information using dense matrix operations (MatMul) [50]. Approximately **70%** of the total cost is attributed to the SpMM operations [23]. Thus, by systematically removing nonessential edges, *graph sparsification* [5, 13] reduces memory and speeds up GNN training and inference while also improving accuracy.

A graph can be sparsified using either unsupervised or supervised methods, and each approach has its own advantages and limitations. Unsupervised sparsification methods can be fast to compute, often supported by strong theoretical guarantees, and



KDD 2026, Jeju Island, Republic of Korea.

ACM ISBN 979-8-4007-2259-2/2026/08
https://doi.org/10.1145/3770855.3818110

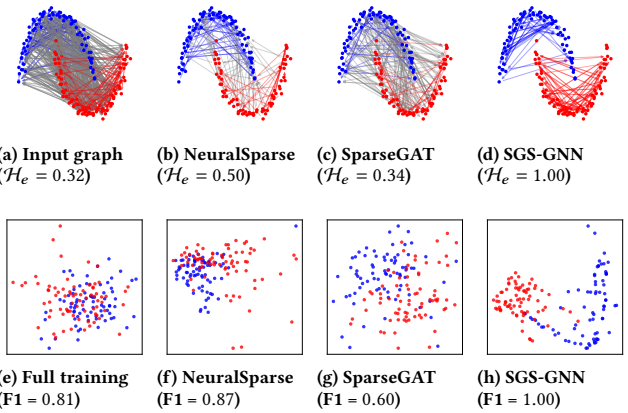


Figure 1: Comparing full graph training, and existing supervised sparsifiers (NeuralSparse, SparseGAT) with SGS-GNN on a synthetic heterophilic graph ($|\mathcal{G}|$ = 150, $|\mathcal{E}|$ = 870, Train = 3%). (a-d) shows the input graph and the sparsified graphs along with edge homophily. (e-h) shows the corresponding node embeddings and F1 scores. SGS-GNN yields clearly seperable embeddings with better predictive performance while preserving task-relevant structure better.

effective at preserving global structural or spectral properties of graphs [2, 5, 9]. However, they are inherently task-agnostic; as a result, edges that are crucial for prediction tasks like node classification or link prediction may be removed, producing sparsified graphs with desired properties yet less effective for learning [51]. In contrast, Supervised sparsification methods learn edge importance directly from supervision and therefore often achieve stronger predictive performance by aligning sparsification with the downstream task [26, 48, 51]. Despite this advantage, existing supervised sparsifiers face several challenges that limit their scalability and performance.

(1) *Computational overhead and scalability:* Supervised sparsifiers frequently require training an additional edge-scoring module [51] alongside the downstream GNN module. The decision to keep or drop edges is discrete and non-differentiable, so most methods rely on differentiable trick such as Gumbel-Softmax [15], which often introduce significant memory overhead due to edge-score gradient retained for all edges during backpropagation [29].

(2) *Sparsity Control:* Methods such as SparseGAT [48] employs L_0-regularization of binary mask to control sparsity. However, such a control does not provide a predictable graph size for the downstream GNN. The mask might be too dense for a given downstream GNN to handle or too sparse to ensure good predictive performance.

(3) *Lack of homophily/heterophily awareness:* The homophily or heterophily regime of a graph provides a useful structural heuristic for identifying which edges are informative for message passing. However, most supervised sparsifiers rely solely on the downstream task loss and do not explicitly leverage this insight.

(4) *Combinatorial search space and optimization difficulty:* The space of possible sparse subgraphs is combinatorially large. Starting from an uninformative or random edge distribution makes it difficult for supervised sparsifiers to discover an effective sampling strategy within a limited number of training iterations.

In this work, we propose SGS-GNN (§5), a fast and memory-efficient supervised graph sparsifier that addresses key limitations of existing supervised sparsification while retaining task-aware edge selection. At a high level, SGS-GNN consists of three components: (i) *Edge probability encoding*, (ii) a *sparse subgraph sampler*, and (iii) a *downstream GNN*.

Our contributions:

(i) **Edge probability encoding (addressing issue #1).** EdgePE is a neighborhood-structure aware edge-scorer, which operates on an initial sparsified graph that is constructed in an unsupervised manner. Prior supervised sparsifiers only leverage endpoint node features, ignoring the effect of neighboring nodes on edge-scoring. This may lead to sub-optimal edge encoding and poor predictive performance. In contrast, EdgePE uses a structure-aware MLP encoder that transforms node features into edge probability. To reduce memory overhead, EdgePE is updated with gradients that flow only through the sampled edges. We couple this with *gradient checkpointing* [33] to avoid persistent retention of edge-level computation graphs, yielding up to a 44% reduction in peak memory without sacrificing accuracy (§6.1).

(ii) **Sparse subgraph sampler (addressing issue #2 & #4).** The sampler in SGS-GNN constructs an *explicit* sparse subgraph under a user-controlled global sparsity budget. This ensures that the downstream GNN performs message passing only on a reduced graph with a prescribed graph size constraint. To ease optimization over the combinatorial subgraph space, the sampler warm-starts sampling using a degree-normalized prior over edges that approximates effective resistance.

(iii) **Downstream learning (addressing issue #3).** The downstream module can be any message-passing GNN trained on the sampled sparse graph. SGS-GNN combines the task objective with a homophily-aware regularizer, and a consistency loss that encourages the learned edge scores between two nodes to remain aligned with the similarity score of their node embeddings. As shown in Fig. 1, SGS-GNN preserves task-relevant structure in a heterophilic graph and yields well-separated embeddings, whereas NeuralSparse and SparseGAT lose critical signal.

(iv) **End-to-end efficiency.** Finally, SGS-GNN employs *conditional updates* so that EdgePE is updated only when it improves over an unsupervised baseline that samples from a degree-based prior distribution over edges. This improves runtime efficiency while ensuring performance that is no worse than strong unsupervised sparsification baselines.

(v) **Memory efficiency.** With *gradient checkpointing*, SGS-GNN reduces peak GPU memory usage by up to 3.9× over existing supervised sparsifiers, and by up to 44% compared to SGS-GNN *without checkpointing* (§6.1).

(vi) **Predictive performance.** We demonstrate consistent empirical gains on 33 benchmarks (21 heterophilic, 12 homophilic), achieving up to 30% F1 improvement using only 20% of edges and outperforming sparsification-based GNN baselines (§6.2, §6.3).

2 Preliminaries and Problem Statement

Let $\mathcal{G} \triangleq (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ be an undirected graph with its set of nodes and edges denoted by $\mathcal{V}$ and $\mathcal{E}$ respectively. The node feature matrix $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}| \times F}$ contains node feature $\boldsymbol{x}_v \in \mathbb{R}^F$ as a row vector for every node $v \in \mathcal{V}$. The adjacency matrix $\boldsymbol{A}_{\mathcal{G}}$ of size $|\mathcal{V}| \times |\mathcal{V}|$ captures the neighborhood of each node in $\mathcal{G}$. In node classification, the goal is to predict a label $y_v \in C$ for each node $v \in \mathcal{V}$ among the $|C|$ possible class labels. The training uses labeled nodes $\mathcal{V}_L \subset \mathcal{V}$, while the unlabeled nodes $\mathcal{V}_U = \mathcal{V} \setminus \mathcal{V}_L$ are used for validation and testing. A single-layer Graph Convolutional Network (GCN) is defined as $\boldsymbol{H}^{(l+1)} = \sigma(\hat{\boldsymbol{A}}_{\mathcal{G}} \boldsymbol{H}^{(l)} \boldsymbol{W}^{(l)})$, where $\boldsymbol{H}^{(l)}$ represents the node embedding at layer l, with $\boldsymbol{H}^{(0)} = \boldsymbol{X}$, and $\boldsymbol{W}^{(l)}$ is the learnable weight matrix [17]. $\hat{\boldsymbol{A}}_{\mathcal{G}}$ denotes the normalized adjacency matrix and σ is an activation function such as ReLU. The predicted probabilities can be expressed as $\tilde{Y} = \texttt{Softmax}(\texttt{GNN}_\theta(\mathcal{G}))$ where $\texttt{GNN}_\theta$ is a GNN with L layers and learnable parameters θ. The dimension of $\tilde{Y}$ is $|\mathcal{V}| \times |C|$. The training objective is to find parameters θ that minimize the cross-entropy loss,

$$\mathcal{L}_{\text{CE}} = -\frac{1}{|\mathcal{V}_L|} \sum_{v \in \mathcal{V}_L} \sum_{c=1}^{|C|} Y_{vc} \log \tilde{Y}_{vc}, \tag{1}$$

where Y_{vc} is the true probability of node v belonging to class $c \in C$.

Problem Statement. Given $q > 0$, the goal is to construct a sparse subgraph $\tilde{\mathcal{G}} \triangleq (\mathcal{V}, \tilde{\mathcal{E}}, \boldsymbol{X})$ with $q\%$ of original edges in $\mathcal{E}$. Let $\mathcal{G}_q$ be the space of all distinct sparse subgraphs of $\mathcal{G}$ where every subgraph contains exactly $k = \lfloor \frac{q|\mathcal{E}|}{100} \rfloor$ edges, $\mathcal{G}_q = \{\tilde{\mathcal{E}} \subset \mathcal{E} : |\tilde{\mathcal{E}}| = k\}$. The objective is to find the parameters θ along with a sparse subgraph $\tilde{\mathcal{G}} \in \mathcal{G}_q$ that minimizes $\mathcal{L}_{\text{CE}}$.

3 Related Work

Unsupervised Graph Sparsification. Effective resistance (ER) based sampling generates sparse subgraphs whose eigenvalues are multiplicative approximations of those of the full graphs. Since exact computation of ER is infeasible for large graphs [38], various approximate approaches are used in GNN [23, 39]. In contrast to spectral sampling, DropEdge [34] uniformly drops edges, GraphSAGE [12] samples fixed-size node neighborhoods, and GraphSAINT [49] samples subgraphs with degree-based normalization to reduce sampling bias and variance. Beyond sampling-based heuristics, topology-based methods include t-spanners [9] (bounding distances for multi-hop aggregation), spanning trees/forests (preserving connectivity), and degree-based sparsifiers [23, 40]. Alternatively, similarity-based methods score edges by structural similarity (e.g., Jaccard in SCAN [47]) or feature similarity (AGS-GNN [7]). Building on these concepts, Mixture of Graphs (MoG) [50] combines multiple priors (e.g., Jaccard similarity, gradient magnitude, and effective resistance) to select sparse subgraphs, and recent works continue to advance unsupervised graph sparsification for GNNs [1, 8, 19, 20, 43, 45].

Supervised Graph Sparsification. Supervised methods leverage downstream task-specific learning cues to retain informative edges [26, 51]. For instance, SparseGAT [48] learns edge attention by L_0-regularized task loss. Similarly, PTDNet [26] iteratively sparsifies the adjacency matrix using its nuclear norm as a regularizer. SGCN [18] alternates between sparsifier and GNN optimization to improve runtime. However, these methods lack control over the size of the generated subgraph. Methods such as NeuralSparse [51] employ local k-neighbor sampling to provide some degree of control over sparsification but also face limitations, such as awareness of homophily/heterophily.

4 Implicit vs Explicit Sparsification: a Characterization.

Supervised sparsifiers can be characterized by their degree of control over sparsification. Implicit sparsifiers do not control sparsity to satisfy a prescribed sparsity budget constraint. Depending on the learning task, sparsification can be too aggressive or too passive. In contrast, explicit sparsifiers must minimize the empirical risk subject to a prescribed budget. In the following, we position SGS-GNN among existing sparsifiers under this characterization.

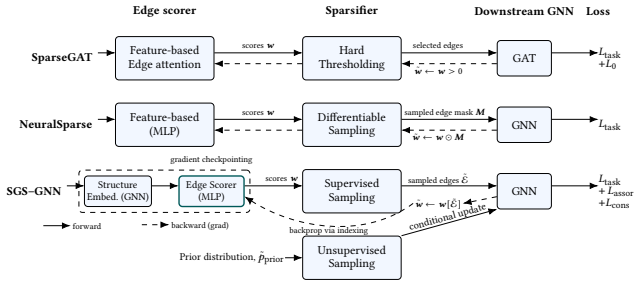


Figure 2: Component comparison of supervised sparsifiers..

Positioning SGS-GNN. In order to harness task-specific learning cues, supervised sparsifiers typically employ some learnable edge-scoring module that encodes edge importance during training. For instance, NeuralSparse [51] learns edge probability using an MLP, whereas SparseGAT [48] learns edge attention scores. SGS-GNN employs a neighborhood structure-aware MLP.

Implicit sparsifiers such as SparseGAT use L_0-norm of the learned mask to penalize the empirical loss, while the downstream model and edge-scorer are trained jointly. Lacking sparsity control, the resulting sparsified graph may remain too dense for downstream message-passing. Explicit sparsifiers, such as NeuralSparse, exert control using a differentiable sampler to explicitly reduce the edges that are passed to the downstream GNN. Despite being explicit in reducing the input graph, NeuralSparse has several limitations: (1) it still keeps autograd tensors associated to the edges that were not sampled, incurring memory overhead, (2) it only controls how many local neighbors of each node can be pruned, lacking any mechanism to enforce strict global sparsity control, and (3) it lacks awareness of homophily/heterophily. Our goal is to design an explicit sparsifier that does not suffer from these limitations.

The core characteristic of SGS-GNN is that *it backpropagates training cues (from training loss, through the edge scores, back to the sparsifier model) only through the sampled edges* . Fig. 2 highlights this aspect. This design choice is to avoid keeping gradients on the pruned edge scores. However, doing only that is insufficient, as SGS-GNN might miss important training cues from neighboring unsampled nodes. To address this issue, it uses a neighborhood-aware edge-scorer. To ensure global sparsity control, it explicitly

samples from the input graph without using any mask. Finally, the homophily aware loss function embeds homophily/heterophily awareness into the edge-scorer. In addition, conditional update reduces peak memory and runtime by updating the edge-scorer only when the learned subgraph outperforms a degree prior-based sparsifier. If the condition is not met, we skip gradients through the edge-scorer to avoid checkpoint-induced recomputation while guaranteeing performance no worse than a degree prior-based sparsifier. These features characterize SGS-GNN while eliminating limitations of existing explicit sparsifiers.

5 Methodology: SGS-GNN

Algorithm 1 SGS-GNN Training outline

1: **Input:** $\mathcal{G}(\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, sample % q, #layers L
2: $p_{\text{prior}}(u,v) \leftarrow \frac{1/d_u+1/d_v}{\sum_{i,j\in\mathcal{E}}(1/d_i+1/d_j)}$
3: **for** each epoch **do**
Module I: EdgePE
4: $\mathcal{E}_{\text{sp}} \leftarrow \text{Sample}(\mathcal{E}, p_{\text{prior}}, \lfloor \frac{q|\mathcal{E}|}{100} \rfloor)$
5: $\boldsymbol{H} \leftarrow \text{GNN}_{\boldsymbol{W}}(\mathcal{E}_{\text{sp}}, \boldsymbol{X}, L)$
6: $\forall_{(u,v)\in\mathcal{E}}\ \boldsymbol{w}(u,v) \leftarrow \sigma\Big(\text{MLP}_\phi\big((\boldsymbol{h}_u^{(i)} - \boldsymbol{h}_v^{(i)}) \oplus (\boldsymbol{h}_u^{(i)} \odot \boldsymbol{h}_v^{(i)})\big)\Big)$ // *checkpoint*
7: $\tilde{p} \leftarrow \text{Normalize}(\boldsymbol{w})$
Module II: Sampling
8: $\tilde{p}_a \leftarrow \lambda\tilde{p} + (1-\lambda)p_{\text{prior}}$ {**Augmenting $\tilde{p}$ with prior**}
9: $\tilde{\mathcal{E}} \leftarrow \text{Sample}(\mathcal{E}, \tilde{p}_a, \lfloor \frac{q|\mathcal{E}|}{100} \rfloor)$
10: $\tilde{\boldsymbol{w}} \leftarrow \boldsymbol{w}[\tilde{\mathcal{E}}]$ {i.e., $\tilde{w}(e_{uv}) = w(e_{uv})\ \forall(u,v) \in \tilde{\mathcal{E}}$.}
Module III: Downstream GNN + Loss
11: $\tilde{\boldsymbol{Y}}, \tilde{\boldsymbol{H}} \leftarrow \text{GNN}_\theta(\tilde{\mathcal{E}}, \boldsymbol{X}, \tilde{\boldsymbol{w}})$
12: Compute $\mathcal{L}_{CE}$, $\mathcal{L}_{\text{assor}}$, and $\mathcal{L}_{\text{cons}}$ using $\tilde{\boldsymbol{Y}}, \tilde{\boldsymbol{H}}$
13: $\mathcal{L} \leftarrow \alpha_1 \cdot \mathcal{L}_{\text{CE}} + \alpha_2 \cdot \mathcal{L}_{\text{assor}} + \alpha_3 \cdot \mathcal{L}_{\text{cons}}$
14: Backpropagate through $\mathcal{L}$ to update GNN_θ, MLP_ϕ, and, $\text{GNN}_{\boldsymbol{W}}$
15: **end for**

SGS-GNN comprises three core components: (i) an edge probability encoding module, (ii) a sparse subgraph sampling module, and (iii) a downstream GNN empowered by a combination of task loss, homophily/heterophily aware loss, and *consistency loss*. Algorithm 1 outlines the pseudocode. It starts from computing a degree-based prior distribution $p_{\text{prior}}(u,v) \propto \left(\frac{1}{d_u} + \frac{1}{d_v}\right)$ over edges.

Role of prior p_{prior}. p_{prior} serves two purposes. First, it provides a stable, task-agnostic starting point for subgraph construction before any learned probabilities are available. Second, it enables a cheap, memory-efficient proxy subgraph for computing structure-aware embeddings without materializing the full graph, which is prohibitively expensive at scale.

One reason to choose this prior is that it biases sampling toward edges incident to low-degree nodes, reducing the probability of isolated nodes appearing in the sampled subgraph. We are also motivated by Corollary 3.3 in Lovász [25], which shows that p_{prior} is a $\max\{2, 1/\alpha\}$-approximation to effective resistance for $\alpha \in (0, 2]$.

5.1 Module I: Edge Probability Encoding

In the EdgePE module using p_{prior}, we sample a subset $\mathcal{E}_{\text{sp}} \subseteq \mathcal{E}$ of size $\left\lfloor \frac{q|\mathcal{E}|}{100} \right\rfloor$ (Alg. 1, line 4) and compute node embeddings on the induced subgraph via $\boldsymbol{H} \leftarrow \text{GNN}_{\boldsymbol{W}}(\mathcal{E}_{\text{sp}}, \boldsymbol{X}, L)$ (Alg. 1, line 5). Using a GNN at this step is important because it injects neighborhood structure into $\boldsymbol{H}$, whereas an MLP-only mapping $\boldsymbol{H} = \text{MLP}(\boldsymbol{X})$ ignores graph connectivity and is typically less effective (see §6.7).

Given embeddings $\boldsymbol{h}_u, \boldsymbol{h}_v$ from $\boldsymbol{H}$, EdgePE assigns each edge $(u,v) \in \mathcal{E}$ an unnormalized confidence score in $[0,1]$ (Alg. 1, line 6):

$$w(e_{uv}) = \sigma\Big(\text{MLP}_\phi\big((\boldsymbol{h}_u - \boldsymbol{h}_v) \oplus (\boldsymbol{h}_u \odot \boldsymbol{h}_v)\big)\Big), \tag{2}$$

where σ is the sigmoid activation, $\oplus$ denotes concatenation, $\odot$ is element-wise multiplication, and MLP_ϕ has learnable parameters ϕ. We *checkpoint* [33] this computation (Alg. 1, line 6) because it is evaluated across *all* edges in $\mathcal{E}$; storing intermediate activations/gradients for every edge can be prohibitively memory-intensive (see the highlighted block in Fig. 3). Checkpointing avoids storing these gradients in the computation graph during the forward pass.

Finally, we normalize the learned weights to obtain a valid sampling distribution $\tilde{p}$ (Alg. 1, line 7). In our implementation we use *sum-normalization*, $\tilde{p}(e_{uv}) = \frac{w(e_{uv})}{\sum_{(i,j)\in\mathcal{E}} w(e_{ij})}$, although other normalization choices (e.g., softmax variants) are also possible (see §9).

5.2 Module II: Sparse Subgraph Sampling

Given the learned distribution $\tilde{p}$, sparse subgraph sampling aims to construct a sparse graph satisfying the user-controlled global budget $q > 0$. Instead of sampling directly from $\tilde{p}$, we *augment* the probability density $\tilde{p}$ with the prior p_{prior} to obtain a mixture density p_a over edges (Alg. 1, line 8):

$$\tilde{p}_a(u,v) = \lambda\,\tilde{p}(u,v) + (1-\lambda)\,p_{\text{prior}}(u,v), \qquad \lambda \in [0,1]. \tag{3}$$

This mixing helps prevent the sampler from over-committing early to a potentially noisy $\tilde{p}$ and reduces the probability of discarding structurally important bridge edges, especially around low-degree nodes. The following theorem, whose proof we defer to the Appendix, shows that the probability of a node becoming isolated is exponentially small.

Theorem 5.1. *If the simple undirected sparsified graph $\tilde{\mathcal{G}}$ is formed by successively sampling without replacement k edges from $\tilde{p}_a$, then for every node $u \in \mathcal{V}$, the probability of its degree being 0 in $\tilde{\mathcal{G}}$*

$$\Pr[d_{u,\tilde{\mathcal{G}}} = 0] \ \le\ \exp\left(-\frac{k(1-\lambda)}{2|\mathcal{V}|}\right).$$

Let $k = \left\lfloor \frac{q|\mathcal{E}|}{100} \right\rfloor$ denote the number of edges to retain. We sample a sparse edge set $\tilde{\mathcal{E}}$ according to $\tilde{p}_a$. A natural choice is to treat $\tilde{p}_a$ as the parameter of a *Multinomial* distribution and sample edges as

$$\tilde{\mathcal{E}} \sim \text{Multinomial}(\tilde{p}_a, k). \tag{4}$$

Crucially, sampling $\tilde{\mathcal{E}}$ also induces the corresponding *sampled edge weights* by indexing (Alg. 1, line 10) into the full edge-score vector $\boldsymbol{w}$ learned in Module I:

$$\tilde{\boldsymbol{w}} \triangleq \boldsymbol{w}[\tilde{\mathcal{E}}] \quad \text{(i.e., } \tilde{w}(e_{uv}) = w(e_{uv})\ \forall(u,v) \in \tilde{\mathcal{E}}\text{)}. \tag{5}$$

We therefore obtain the sparse weighted subgraph $\tilde{\mathcal{G}} = (\mathcal{V}, \tilde{\mathcal{E}}, \boldsymbol{X}, \tilde{\boldsymbol{w}})$, where $\tilde{\boldsymbol{w}}$ will be used by the downstream GNN in Module III.

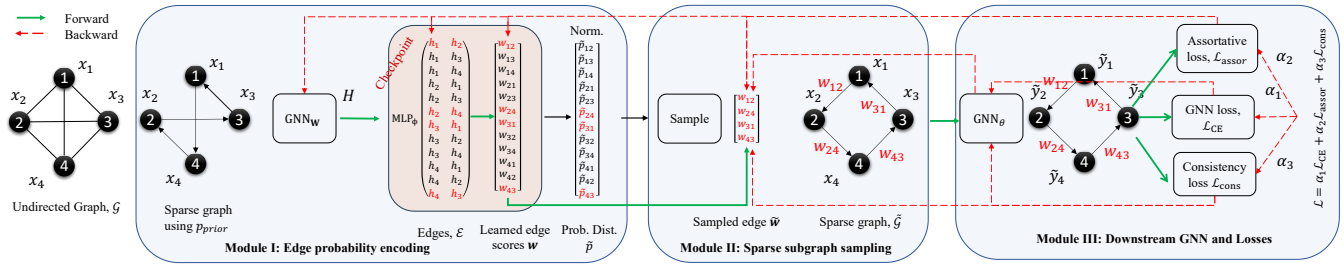


Figure 3: Overview of the three modules in SGS-GNN. EdgePE computes edge sampling probabilities, the sampler selects a sparse subgraph, and the downstream GNN operates on the sampled edges only. To reduce the high memory cost of edge-level MLPs, SGS-GNN applies gradient checkpointing.

5.3 Module III: Downstream GNN and Losses

Given the sampled sparse subgraph $\tilde{\mathcal{G}} = (\mathcal{V}, \tilde{\mathcal{E}}, X, \tilde{\boldsymbol{w}})$ from Module II, we perform the downstream prediction task by running a GNN on $\tilde{\mathcal{E}}$ while *retaining the sampled edge weights* $\tilde{\boldsymbol{w}}$ in the computation graph. This allows gradients from the task loss to flow through $\tilde{\boldsymbol{w}}$ and update EdgePE during backpropagation. The forward pass of the downstream GNN computes

$$\tilde{Y} = \texttt{Softmax}\Big(\text{GNN}_\theta(\mathcal{V}, \tilde{\mathcal{E}}, X, \tilde{\boldsymbol{w}})\Big), \tag{6}$$

where GNN_θ is any edge-weighted message-passing GNN.

Loss functions. We optimize the downstream objective together with two regularizers that bias the sparsifier toward task-relevant and structurally consistent edges (Fig. 4). The overall objective is

$$\mathcal{L} = \alpha_1\mathcal{L}_{\text{CE}} + \alpha_2\mathcal{L}_{\text{assor}} + \alpha_3\mathcal{L}_{\text{cons}}, \tag{7}$$

where $\mathcal{L}_{\text{CE}}$ is standard **task loss** computed on labeled nodes (Case II in Fig. 4) and $\alpha_1, \alpha_2, \alpha_3 \in [0, 1]$ are coefficients.

Assortativity regularizer. To discourage selecting heterophilic edges among labeled nodes, we increase the weight of edges that connect training nodes with the same label:

$$\mathcal{L}_{\text{assor}} \triangleq -\sum_{(u,v)\in\tilde{\mathcal{E}}:\, u\in\mathcal{V}_L \wedge v\in\mathcal{V}_L} \mathbb{I}(y_u = y_v)\cdot \log w(e_{uv}), \tag{8}$$

where $\mathcal{V}_L$ denotes the set of labeled nodes, $\mathbb{I}(\cdot)$ is the indicator function, and $w(e_{uv})$ is the edge score from Eq. (2). By penalizing small $w(e_{uv})$ on same-label pairs, this term induces lower probability mass on heterophilic edges (Case III in Fig. 4).

Consistency regularizer. We further align edge scores with representation similarity by encouraging sampled edges to have weights consistent with cosine similarity of node embeddings:

$$\mathcal{L}_{\text{cons}} \triangleq \frac{1}{|\tilde{\mathcal{E}}|}\sum_{(u,v)\in\tilde{\mathcal{E}}} \Big\| w(e_{uv}) - \cos(\boldsymbol{h}_u^l, \boldsymbol{h}_v^l)\Big\|_2^2, \tag{9}$$

where $\cos(\boldsymbol{h}_u^l, \boldsymbol{h}_v^l) = \frac{\boldsymbol{h}_u^l \cdot \boldsymbol{h}_v^l}{\|\boldsymbol{h}_u^l\|\|\boldsymbol{h}_v^l\|}$ is the cosine similarity between embeddings from layer l. This term reduces the selection probability of edges whose endpoints are far in representation space (Case IV in Fig. 4).

The following theorem upper-bounds the expected error in the downstream SGS-GNN node encodings (proof deferred to appendix).

Theorem 5.2. *(Error in GCN encodings) Let $\tilde{\boldsymbol{H}}$ and $\boldsymbol{H}^*$ be the corresponding learned node encodings from an L-layer GCN trained on the sparse graph $\tilde{\mathcal{G}}$ and true optimal sparse graph $\mathcal{G}^*$, respectively. Let the samplers sample k edges successively, without replacement following their respective distributions $\tilde{p}$ and p^*. Let us assume the approximation error $\|p^* - \tilde{p}\|_\infty \le \epsilon$, such that in the diffuse (non-concentrated) regime: there exists $\rho \in (0, 1/4]$ such that $p^*_{\max} \le \frac{\rho}{k}$, $\tilde{p}_{\max} \le \frac{\rho}{k}$. If for all layer ℓ, GCN weights $\|\boldsymbol{W}^{(\ell)}\|_2 \le \alpha$, $\alpha \in [0, 1]$ and hidden states $\|\tilde{\boldsymbol{H}}^{(\ell)}\|_2, \|\boldsymbol{H}^{*(\ell)}\|_2 \le \beta$, $\beta > 0$, then for sufficiently large L,*

$$\mathbb{E}[\lim_{L\to\infty} \|\tilde{\boldsymbol{H}}^{(L)} - \boldsymbol{H}^{*(L)}\|_2]$$
$$\le \frac{\beta\alpha}{1-\alpha}\min\{2, \frac{2}{d_{\min}}\Delta_{\text{sam}}(p^*, \tilde{p}) + \frac{4(d^*_{\max}+1)}{d_{\min}^{3/2}}\Delta_{\text{sam}}(p^*, \tilde{p})^2\},$$

where $d^*_{\max}$ is the maximum degree in $\mathcal{G}^*$, $\Delta_{\text{sam}}(p^*, \tilde{p})$ is the sampling disagreement radius

$$\Delta_{\text{sam}}(p^*, \tilde{p}) \triangleq \sqrt{k\left(1 - k(1-12\rho)\sum_{j=1}^{m}\frac{(p_j^* + \tilde{p}_j - \epsilon)^2}{4}\right)}.$$

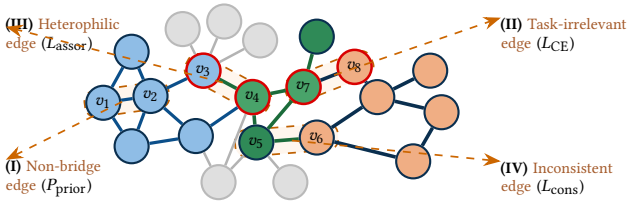


Figure 4: Four inductive-bias cases guiding edge removal (training nodes have red borders): (I) p_{prior} down-weights non-bridge edges; (II) $\mathcal{L}_{\text{CE}}$ down-weights task-irrelevant edges; (III) $\mathcal{L}_{\text{assor}}$ discourages heterophilic labeled edges; (IV) $\mathcal{L}_{\text{cons}}$ down-weights embedding-inconsistent edges.

5.4 End-to-end Training Efficiency

Algorithm 2 summarizes the end-to-end training pipeline of SGS-GNN for large-scale graphs with two efficiency mechanisms: (i) *locality-aware graph partitioning* for batch processing, and (ii) *conditional updates* of the edge-probability module to avoid redundant gradient computations.

Locality-aware batch processing via graph partitioning. Computing edge scores over hundreds of millions to billions of edges can be time- and memory-intensive, since we need to run a forward pass through MLP_ϕ. Partitioning the graph into local subgraphs bounds

Algorithm 2 Training epoch with conditional update

1: **Input:** graph partitions $\mathcal{G}_{\text{parts}} = \{\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_n\}$
2: **for** each graph partition $\mathcal{G}_i(\mathcal{E}_i, \mathcal{V}_i, \boldsymbol{X}_i, p^i_{\text{prior}}) \in \mathcal{G}_{\text{parts}}$ **do**
3: $\tilde{p}_i, \boldsymbol{w}_i \leftarrow \texttt{EdgePE}_\phi(\mathcal{G}_i, p^i_{\text{prior}}, L)$
4: $\tilde{\mathcal{E}}_i, \tilde{\boldsymbol{w}}_i \leftarrow$ SAMPLE a subgraph using $\tilde{p}_i$
5: $\mathcal{E}^i_{\text{prior}} \leftarrow$ SAMPLE a subgraph using p^i_{prior}
6: **if** Evaluate$(\texttt{GNN}_\theta(\tilde{\mathcal{E}}_i, \tilde{\boldsymbol{w}}_i)) \geq$ Evaluate$(\texttt{GNN}_\theta(\mathcal{E}^i_{\text{prior}}))$ **then**
7: Backpropagate through $\mathcal{L}$ to update $\texttt{EdgePE}_\phi$ and $\texttt{GNN}_\theta$.
8: **else**
9: Backpropagate only through $\mathcal{L}_{\text{CE}}$ to update $\texttt{GNN}_\theta$.
10: **end if**
11: **end for**

memory by limiting computation to a subset of nodes/edges and increases locality, enabling message passing within local regions. It also supports mini-batching, which introduces beneficial stochasticity during optimization and can improve accuracy on large graphs (§10). We use METIS [16] since it produces approximately balanced partitions while minimizing edge cuts, yielding batches with high intra-partition connectivity and limited cross-partition communication, which is a practical fit for GPU training [6] on large sparse graphs. Partitioning alone does not remove task-irrelevant edges. By learning a sparse subgraph within each partition, SGS-GNN removes such edges, thereby reducing computation and memory usage. As shown later in experiments (§6.2), this leads to higher accuracy than using the full partitioned graph (e.g., ClusterGCN; see the *Org. Graph.* column in Table 4) on heterophilic datasets, even when using identical partitions.

Conditional updates to reduce compute and memory. One of the major computational bottlenecks is when edge scores are computed for all edges, and gradients must flow back to update EdgePE. Although we apply checkpointing to the edge-scoring computation (Alg. 1, line 6), backpropagating through EdgePE still incurs recomputation overhead. To avoid paying this cost at every iteration, we use a *conditional update* policy (Alg. 2) that updates EdgePE only when it is beneficial.

For each partition, we compare the downstream of the learned sampler against a sample from the fixed prior. If the learned sampler improves over baseline, we backpropagate through the full loss to update both EdgePE and the downstream GNN. Otherwise, we update only the downstream GNN by backpropagating $\mathcal{L}_{\text{CE}}$ (Alg. 2, lines 4–8) for the prior. This way, the conditional update encourages EdgePE 's performance to be at least as good as that of the unsupervised prior-based sparsifier.

5.5 Computational Complexity

Let hidden dimensions $H \approx F$, where F is the dimension of the node features. An L-layer GCN costs $\mathcal{O}\big(L(|\mathcal{E}|H + |\mathcal{V}|H^2)\big)$ [6]. In SGS-GNN, the edge scorer computes node embeddings on $\mathcal{E}_{\text{sp}}$ at cost $\mathcal{O}\big(L(|\mathcal{E}_{\text{sp}}|H + |\mathcal{V}|H^2)\big)$ and then scores all edges with an r-layer MLP at $\mathcal{O}(|\mathcal{E}|\, rH^2)$. The downstream L-layer GCN on $\tilde{\mathcal{E}}$ costs $\mathcal{O}\big(L(|\tilde{\mathcal{E}}|H + |\mathcal{V}|H^2)\big)$. Thus, $T^{\text{train}}_{\texttt{SGS-GNN}} = \mathcal{O}\Big(L\big((|\mathcal{E}_{\text{sp}}| + |\tilde{\mathcal{E}}|)H + 2|\mathcal{V}|H^2\big) + |\mathcal{E}|\, rH^2\Big)$. With checkpointing applied only to the edge-scoring MLP block, $T^{\text{train}}_{\texttt{SGS-GNN,ckpt}} = T^{\text{train}}_{\texttt{SGS-GNN}} + \mathcal{O}(|\mathcal{E}|\, rH^2)$.

Space Complexity (Alg. 1). Without checkpointing,

$$S_{\texttt{SGS-GNN}} = \mathcal{O}\big(|\mathcal{E}| + |\mathcal{V}|F + L|\mathcal{V}|H + |\mathcal{E}| \cdot rH + (L + r)H^2\big),$$

where $L|\mathcal{V}|H$ accounts for stored GNN activations for backpropagation, and $|\mathcal{E}| \cdot rH$ accounts for stored intermediate MLP activations over all edges. With checkpointing on $\texttt{MLP}_\phi$, GNN activations are unchanged, while MLP activation storage is reduced by recomputing intermediate activations during backpropagation instead of storing them in the forward pass.

$$S_{\texttt{SGS-GNN}} = \mathcal{O}\big(|\mathcal{E}| + |\mathcal{V}|F + L|\mathcal{V}|H + |\mathcal{E}| \cdot H + (L + r)H^2\big).$$

Note that, checkpointing is most beneficial when gradients of $\texttt{MLP}_\phi$ dominates, i.e., $|\mathcal{E}| \cdot rH \gg L|\mathcal{V}|H$; otherwise, the memory savings are limited relative to the added recomputation overhead.

6 Experiments

We experimented with 33 benchmark datasets of varying sizes and homophily on node classification tasks (Table 1). All experiments are run 10 times on a 24GB NVIDIA A10 GPU with 500 GB memory, using a 20%/40%/40% train/val/test split unless the benchmark specifies otherwise. For baseline models, we follow the settings set by the respective authors. We use 2 message passing layers for SGS-GNN and a hidden layer dimension of $H = 256$ for both EdgePE and $\texttt{GNN}_\theta$. We set a dropout rate of 0.2 and use the Adam optimizer with a learning rate of 0.001. We trained all models for a maximum epoch of 500 with early stopping. The edge batch size is $500K$, and the number of partitions for METIS is $n = \lceil |\mathcal{E}|/500K \rceil$. The percentage of edge sample is set to $q = 20\%$ following DropEdge [34]. We take 10 samples of sparse subgraphs during inference. The model with the best validation F1-score is selected for testing. Our source codes are anonymously provided on https://github.com/anonymousauthors001/SGS-GNN/.

Baselines. We use ClusterGCN [6] to evaluate performance on the original large graphs. DropEdge [34], and GraphSAINT (GSAINT-E) [49] are used as fixed distribution samplers. For Mixture of Graph (MoG) [50], we use 3 experts and their recommended settings. We adjust the neighborhood sample size of NeuralSparse [51] to match the sparsity of ours for a fair comparison. We included SparseGAT [48] as another supervised method for generating sparse graphs. We also compare with heterophilic GNNs (H2GCN [53], ASGC [3]), sparsifiers based on the lottery ticket hypothesis (UGT-GCN [4]), and others (DSpar [24], CGP-GCN [22], UNIFEWS [19], SGNN-LS [8], SGFormer [45]).

6.1 Memory efficiency

Table 2 shows that SGS-GNN consistently attains lower peak GPU memory than three related sparsification baselines across eight datasets. SGS-GNN has up to 3.9× less peak memory usage than related supervised sparsifier. A key reason is that SGS-GNN maintains gradient flow only through the sampled subgraph, whereas several supervised sparsifiers retain gradients for all edges in the forward pass which does not reduce peak memory and often triggers out of memory (OOM) error on large graphs.

We also report an ablation of SGS-GNN without gradient checkpointing. Even without checkpointing, SGS-GNN uses less peak memory than the related methods. Checkpointing is not beneficial in

Table 1: Dataset statistics. heterophilic-red, homophilic-blue. d-average degree, C-#classes, and F-feature dimension.

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	d	$\mathcal{H}_{\text{adj}}$	C	F
Cornell	183	557	3.04	-0.42	5	1.70K
Texas	183	574	3.14	-0.26	5	1.70K
Wisconsin	251	916	3.65	-0.20	5	1.70K
reed98	962	37.62K	39.11	-0.10	3	1.00K
amherst41	2.24K	181.91K	81.39	-0.07	3	1.19K
penn94	41.55K	2.72M	65.56	-0.06	2	4.81K
Roman-empire	22.66K	65.85K	2.91	-0.05	18	300
cornell5	18.66K	1.58M	84.76	-0.04	3	4.74K
Squirrel	5.20K	396.85K	76.30	-0.01	5	2.35K
johnshopkins55	5.18K	373.17K	72.04	0.00	3	2.41K
Actor	7.60K	53.41K	7.03	0.01	5	932
Minesweeper	10.00K	78.80K	7.88	0.01	2	7
Questions	48.92K	307.08K	6.28	0.02	2	301
Chameleon	2.28K	62.79K	27.58	0.03	5	2.58K
Tolokers	11.76K	1.04M	88.28	0.09	2	10
Amazon-ratings	24.49K	186.10K	7.60	0.14	5	556
genius	421.96K	1.85M	4.37	0.17	2	12
pokec	1.63M	44.60M	27.32	0.42	3	65
arxiv-year	169.34K	2.32M	13.67	0.26	5	128
snap-patents	2.92M	27.95M	9.56	0.21	5	269
ogbn-proteins	132.53K	79.12M	597.00	0.05	94	8
Cora	19.79K	126.84K	6.41	0.56	70	8.71K
DBLP	17.72K	105.73K	5.97	0.68	4	1.64K
Computers	13.75K	491.72K	35.76	0.68	10	767
PubMed	19.72K	88.65K	4.50	0.69	3	500
Cora_ML	2.99K	16.32K	5.45	0.75	7	2.88K
SmallCora	2.71K	10.56K	3.90	0.77	7	1.43K
CS	18.33K	163.79K	8.93	0.78	15	6.80K
Photo	7.65K	238.16K	31.13	0.79	8	745
Physics	34.49K	495.92K	14.38	0.87	5	8.42K
CiteSeer	4.23K	10.67K	2.52	0.94	6	602
wiki	11.70K	431.73K	36.90	0.58	10	300
Reddit	232.97K	114.62M	491.99	0.74	41	602

small graphs and we can see it can reduces peak memory use by 30% in large graph like `Reddit` and up to 44% in `cornell5`.

Additionally, locality-aware partitioning further improves scalability by restricting computation to subgraphs with high intra-partition locality. Table 3 shows that, even when we apply the same METIS partitioning used by `SGS-GNN` to NeuralSparse and MoG on `Reddit`, `SGS-GNN` remains more memory efficient than both baselines. This shows that the memory advantage of `SGS-GNN` does not come solely from METIS partitioning.

Table 2: Peak GPU memory (GB). SGS*/SGS+ denote runs without/with gradient checkpointing. For each dataset, the minimum memory is normalized to 1.0 (bold); parentheses show relative factors. Imp.(%) reports the memory reduction from checkpointing. #-METIS partitions of `SGS-GNN`; *OOM* denotes out-of-memory. NS: NeuralSparse; SG: SparseGAT.

Data	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	NS	SG	MoG	SGS*	SGS+	Imp.(%)	#
amhe.	2.24K	182K	2.42 (1.8×)	3.12 (2.3×)	7.44 (5.6×)	**1.33** (1.0×)	1.33 (1.0×)	–	1
amaz.	24.5K	186K	3.85 (2.5×)	2.68 (1.8×)	4.50 (3.0×)	1.57 (1.0×)	**1.51** (1.0×)	3.2	1
Tolo.	11.8K	1.04M	5.60 (1.5×)	8.48 (2.2×)	4.24 (1.1×)	6.35 (1.7×)	**3.83** (1.0×)	39.6	2
John.	5.18K	373K	8.12 (3.0×)	6.09 (2.2×)	OOM	**2.72** (1.0×)	2.72 (1.0×)	–	1
corn.	18.7K	1.58M	OOM	18.79 (3.9×)	OOM	8.58 (1.8×)	**4.80** (1.0×)	44.0	2
arxi.(3)	169K	2.32M	22.06 (2.7×)	22.01 (2.7×)	16.34 (2.0×)	11.62 (1.4×)	**8.08** (1.0×)	30.5	3
arxi.(5)	169K	2.32M	22.06 (3.9×)	22.01 (3.9×)	16.34 (2.9×)	7.69 (1.4×)	**5.62** (1.0×)	26.9	5
Wiki	11.7K	432K	3.62 (1.2×)	3.96 (1.3×)	5.51 (1.8×)	**3.11** (1.0×)	3.12 (1.0×)	–	1
Redd.(115)	233K	114.6M	OOM	OOM	OOM	12.96 (1.4×)	**9.03** (1.0×)	30.4	115
Redd.(230)	233K	114.6M	OOM	OOM	OOM	6.58 (1.3×)	**4.92** (1.0×)	25.2	230

Table 3: Peak GPU memory (GB) on `Reddit` when all methods use the same METIS partitioning.

Data	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	#	NS	MoG	SGS*	SGS+
Redd.	233K	114.6M	115	14.15 (1.57×)	19.81 (2.19×)	12.96 (1.44×)	**9.03** (1.0×)
Redd.	233K	114.6M	230	7.70 (1.57×)	14.13 (2.87×)	6.58 (1.34×)	**4.92** (1.0×)

Table 4: Mean F1 (%) ± std. over fixed samplers using 20% edges. Bold: best sampler excluding *Org. graph*. GM: geometric mean.

Data.	Org. Graph	Random	Edge	ER	SGS-GNN
Corn.	43.78 ± 4.32	49.19 ± 4.65	46.49 ± 2.65	43.78 ± 3.97	**74.59 ± 1.32**
Texa.	61.62 ± 1.08	55.14 ± 5.01	69.19 ± 2.76	61.08 ± 2.76	**76.22 ± 2.02**
Wisc.	51.76 ± 5.49	61.96 ± 4.40	66.27 ± 2.29	58.82 ± 2.15	**76.08 ± 3.14**
reed.	61.35 ± 0.84	54.92 ± 2.64	54.51 ± 1.98	59.69 ± 2.03	**64.15 ± 2.28**
amhe.	61.83 ± 0.30	57.49 ± 0.81	57.40 ± 0.58	50.60 ± 1.14	**72.75 ± 0.59**
penn.	73.23 ± 0.05	68.52 ± 0.34	68.35 ± 0.36	70.74 ± 0.39	**75.65 ± 0.41**
Roma.	44.25 ± 0.16	43.08 ± 0.61	42.91 ± 0.63	58.18 ± 1.25	**64.69 ± 0.12**
corn.	65.13 ± 0.31	63.36 ± 0.50	63.47 ± 0.46	63.89 ± 0.35	**69.15 ± 0.33**
Squi.	48.38 ± 0.65	42.40 ± 0.96	42.44 ± 1.10	43.86 ± 0.42	**52.35 ± 0.35**
John.	68.71 ± 0.28	63.40 ± 0.53	62.80 ± 0.74	62.14 ± 2.51	**73.80 ± 0.33**
Acto.	28.42 ± 0.23	32.37 ± 0.78	30.53 ± 0.59	32.03 ± 0.27	**33.88 ± 0.42**
Mine.	79.56 ± 0.03	79.73 ± 0.12	79.84 ± 0.06	**80.02 ± 0.03**	80.00 ± 0.00
Ques.	97.05 ± 0.01	97.07 ± 0.03	**97.08 ± 0.01**	97.02 ± 0.00	97.05 ± 0.01
Cham.	64.43 ± 0.43	57.98 ± 1.39	57.11 ± 1.22	59.78 ± 0.85	**62.37 ± 0.98**
Tolo.	79.03 ± 0.15	78.59 ± 0.16	78.59 ± 0.21	78.10 ± 0.06	**79.98 ± 0.17**
Amaz.	46.72 ± 0.20	45.70 ± 0.21	45.75 ± 0.35	44.39 ± 0.12	**50.15 ± 0.34**
geni.	80.80 ± 0.02	81.99 ± 0.09	81.60 ± 0.03	82.25 ± 0.86	**82.59 ± 0.00**
poke.	62.05 ± 0.37	60.30 ± 0.27	60.17 ± 0.17	58.76 ± 0.59	**60.49 ± 0.10**
arxi.	39.05 ± 0.08	36.96 ± 0.01	37.06 ± 0.04	36.62 ± 0.33	**38.42 ± 0.10**
snap.	35.38 ± 0.15	34.57 ± 0.08	34.48 ± 0.16	33.13 ± 0.37	**35.41 ± 0.10**
ogbn.	93.15 ± 0.00	93.15 ± 0.00	93.15 ± 0.00	93.15 ± 0.00	93.15 ± 0.00
GM.	58.42	57.25	57.63	57.68	64.79
Cora.	67.29 ± 0.51	61.20 ± 7.76	57.63 ± 14.73	**66.90 ± 0.17**	65.58 ± 0.69
DBLP.	83.92 ± 0.04	81.00 ± 0.27	81.19 ± 0.33	**81.81 ± 0.10**	80.37 ± 0.16
Comp.	90.19 ± 0.18	90.34 ± 0.29	90.37 ± 0.25	89.87 ± 0.96	**90.97 ± 0.31**
PubM.	86.73 ± 0.07	87.58 ± 0.22	87.62 ± 0.14	**87.70 ± 0.11**	87.52 ± 0.15
Cora.	86.29 ± 0.51	85.39 ± 0.35	85.29 ± 0.60	**85.63 ± 0.51**	83.99 ± 0.53
Smal.	80.28 ± 0.37	75.82 ± 0.54	76.44 ± 1.21	75.90 ± 1.25	**76.94 ± 0.76**
CS.	92.79 ± 0.10	94.07 ± 0.14	94.09 ± 0.09	93.77 ± 0.17	**94.25 ± 0.15**
Phot.	92.41 ± 2.01	93.54 ± 0.14	93.63 ± 0.25	93.42 ± 0.10	**93.99 ± 0.25**
Phys.	96.08 ± 0.03	96.20 ± 0.07	96.23 ± 0.12	96.22 ± 0.07	**96.27 ± 0.09**
Cite.	91.44 ± 0.29	86.38 ± 0.26	86.75 ± 0.22	86.24 ± 0.26	**86.78 ± 0.28**
wiki.	80.07 ± 0.21	80.10 ± 0.13	80.19 ± 0.16	80.32 ± 0.13	**81.49 ± 0.31**
Redd.	91.43 ± 0.07	91.39 ± 0.06	91.35 ± 0.08	91.00 ± 0.06	**91.45 ± 0.06**
GM.	86.22	84.68	84.37	85.33	85.36

6.2 `SGS-GNN` vs. fixed distr. sparsifiers

We compare `SGS-GNN` with fixed edge distribution sparsifiers like *Random* from DropEdge, *Edge* sampler from GraphSAINT, and *Effective resistance (ER)*. Table 4 presents F1-scores, with the last row summarizing overall performance. The *Org. Graph* represents the original dense graph's performance computed using ClusterGCN. Our learnable sampler `EdgePE` significantly outperforms fixed distribution samplers and original dense graphs on heterophilic datasets. On homophilic graphs, we observe a smaller margin of improvement, but `SGS-GNN` still outperforms other baselines.

6.3 `SGS-GNN` vs. sparsification baselines

Table 5 compares `SGS-GNN` with representative sparsification-based GNN baselines under a common setup. For `SGS-GNN`, we instantiate the downstream predictor with a GCN. Overall, `SGS-GNN` achieves the best average performance, with a geometric-mean improvement of approximately 4–5% while retaining only 20% of the edges. The gains are most pronounced on heterophilic graphs, where eliminating task-irrelevant or misleading edges is particularly important, and `SGS-GNN` remains competitive on homophilic graphs under the same budget. Additional comparisons with methods beyond sparsifiers, including heterophilic GNNs and lottery-ticket-style approaches, are reported in Appendix G.

6.4 Runtime

Table 6 compares the training times per epoch of `SGS-GNN` with other GNN-based sparsifier baselines. Under similar conditions, `SGS-GNN` is more efficient than NeuralSparse, MoG, and competitive

Table 5: Mean F1-score (%) ± std. dev. for baseline sparsifiers with 20% edges. Bold: best method. OOM: out of memory. GM: geometric mean, computed over datasets where all methods report results.

Data.	GSAINT-E	DropEdge	MoG	SparseGAT	NeuralSp.	SGS-GNN
Corn.	46.49 ± 2.65	43.24 ± 2.20	42.16 ± 3.08	51.35 ± 0.10	72.43 ± 6.48	**74.59 ± 1.32**
Texa.	69.19 ± 2.76	54.95 ± 3.37	57.30 ± 4.83	66.66 ± 1.27	**84.44 ± 1.53**	76.22 ± 2.02
Wisc.	66.27 ± 2.29	48.36 ± 0.92	53.33 ± 1.64	56.86 ± 0.00	52.83 ± 47.00	**76.08 ± 3.14**
reed.	54.51 ± 1.98	60.03 ± 0.05	55.75 ± 2.61	55.69 ± 0.25	58.54 ± 1.60	**64.15 ± 2.28**
amhe.	57.40 ± 0.58	59.00 ± 18.80	56.78 ± 2.05	49.00 ± 0.20	56.85 ± 75.00	**72.75 ± 0.59**
penn.	68.35 ± 0.36	65.87 ± 0.30	OOM	65.00 ± 0.10	OOM	**75.65 ± 0.41**
Roma.	42.91 ± 0.63	46.00 ± 1.20	39.27 ± 0.60	41.18 ± 0.07	44.91 ± 5.79	**64.69 ± 0.12**
corn.	63.47 ± 0.46	61.40 ± 1.45	OOM	53.77 ± 0.39	OOM	**69.15 ± 0.33**
Squi.	42.44 ± 1.10	48.60 ± 0.00	27.67 ± 0.51	29.50 ± 0.00	38.24 ± 0.00	**52.35 ± 0.35**
john.	62.80 ± 0.74	64.14 ± 1.75	OOM	57.57 ± 0.29	57.56 ± 1.06	**73.80 ± 0.33**
Acto.	30.53 ± 0.59	34.64 ± 1.40	27.74 ± 0.96	25.05 ± 0.60	27.85 ± 0.19	**33.88 ± 0.42**
Mine.	79.84 ± 0.06	**80.00 ± 0.00**	**80.00 ± 0.00**	**80.00 ± 0.00**	**80.00 ± 0.10**	**80.00 ± 0.00**
Ques.	**97.08 ± 0.01**	97.00 ± 0.01	97.04 ± 0.01	**97.08 ± 0.01**	97.02 ± 0.01	97.05 ± 0.01
Cham.	57.11 ± 1.22	50.60 ± 0.04	53.25 ± 0.63	60.60 ± 0.15	60.52 ± 0.78	**62.37 ± 0.98**
Tolo.	78.59 ± 0.21	78.40 ± 0.20	78.49 ± 0.28	78.20 ± 0.72	78.16 ± 0.00	**79.98 ± 0.17**
Amaz.	45.75 ± 0.35	43.87 ± 0.67	41.18 ± 0.49	44.23 ± 0.05	47.05 ± 0.47	**50.15 ± 0.34**
GM.	56.44	55.04	51.15	53.04	58.25	65.99
Cora	57.63 ± 14.73	65.09 ± 0.44	**67.26 ± 1.11**	61.07 ± 0.39	56.68 ± 0.32	65.58 ± 0.69
DBLP	81.19 ± 0.33	87.20 ± 0.15	72.37 ± 0.63	**84.68 ± 1.00**	73.39 ± 0.67	80.37 ± 0.16
Comp.	90.37 ± 0.25	60.65 ± 4.66	OOM	88.91 ± 0.00	75.32 ± 4.11	**90.97 ± 0.31**
PubM.	**87.62 ± 0.14**	86.00 ± 1.21	83.84 ± 0.58	75.30 ± 0.35	73.97 ± 0.40	87.52 ± 0.15
Cora.	**85.29 ± 0.60**	84.80 ± 0.20	OOM	80.60 ± 0.40	79.30 ± 0.87	83.99 ± 0.53
Smal.	76.44 ± 1.21	76.47 ± 0.31	78.43 ± 0.73	75.70 ± 0.43	75.79 ± 9.00	**76.94 ± 0.76**
CS	94.09 ± 0.09	93.30 ± 0.32	72.88 ± 0.32	92.35 ± 0.06	94.36 ± 0.40	**94.25 ± 0.15**
Phot.	93.63 ± 0.25	80.47 ± 59.10	83.84 ± 0.58	**95.30 ± 0.02**	94.16 ± 0.25	93.99 ± 0.25
Phys.	96.23 ± 0.12	97.28 ± 0.70	OOM	95.96 ± 0.06	96.38 ± 0.04	**96.27 ± 0.09**
Cite.	86.75 ± 0.22	77.40 ± 0.10	78.43 ± 0.73	58.75 ± 0.10	65.40 ± 1.20	**86.78 ± 0.28**
wiki	80.19 ± 0.16	80.19 ± 0.16	72.88 ± 0.32	79.26 ± 0.11	73.03 ± 1.10	**81.49 ± 0.31**
GM.	81.36	80.36	76.04	76.78	74.89	82.87

with SparseGAT. Since SparseGAT is an implicit sparsifier, unlike SGS-GNN, it is not memory efficient and cannot handle large graphs. Unsupervised sparsifiers such as DropEdge and GraphSAINT are faster but not always as accurate as SGS-GNN.

Table 6: Mean training time per epoch (in milliseconds). Red - Unsupervised sparsifiers, Blue - Supervised spasrifiers.

Data.	Clu.GCN	GSAINT	D.Edge	MOG	S.GAT	NeuralSp.	SGS-GNN
Amaz.	6.8	6.2	17.0	110.0	15.0	50.0	18.0
john.	7.1	6.1	21.0	OOM	10.0	120.0	24.0
amhe.	6.2	5.8	10.0	OOM	5.3	37.0	16.0
CS	9.5	8.9	15.0	OOM	100.0	150.0	22.0
Ques.	8.2	7.2	29.0	130.0	24.0	120.0	26.0

Table 7 compares SGS-GNN with and without conditional updates on large-scale graphs ($|\mathcal{E}| \geq 1$M). Conditional updates consistently reduce per-epoch runtime (up to 1.135× speedup) while maintaining comparable F1-scores under identical settings. Due to conditional updates, EdgePE is updated less frequently (about 40% of times), and coupled with gradient checkpointing this reduces overall training time.

6.5 Convergence

To compare the sparsifiers in terms of convergence, we terminate training when the std. dev of loss in five consecutive epochs is $\leq 10^{-3}$. Fig. 5 shows the bar plot of the number of epochs required for the methods to converge. SGS-GNN requires fewer iterations than other fixed distribution sparsifiers highlighting the benefit of learning the distribution.

Table 7: Runtime and F1 of SGS-GNN with/without conditional updates on large graphs ($|\mathcal{E}| \geq 1$M). SpeedUp (SU) = $t_{w/o}/t_{w/}$ denotes the relative runtime. #EdgePE/#GNN$_\theta$ is the EdgePE update frequency.

Data	Time (s)/epoch		SU	F1-Scores		#EdgePE/#GNN
	w/o	w/	$t_{w/o}/t_{w/}$	w/o	w/	
Tolo.	0.1965	**0.1731**	1.135	78.12 ± 0.13	78.13 ± 0.17	0.42
geni.	0.4424	**0.4334**	1.021	79.92 ± 0.08	80.07 ± 0.11	0.43
arxi.	0.4847	**0.4450**	1.089	36.99 ± 0.11	36.98 ± 0.13	0.23
Redd.	8.7202	**7.8851**	1.106	91.45 ± 0.07	91.43 ± 0.02	0.44

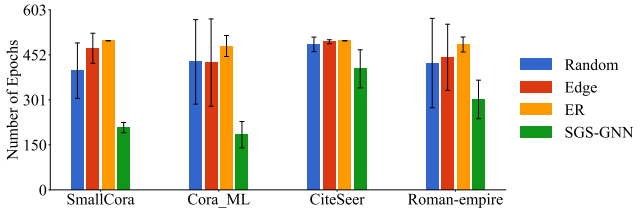


Figure 5: Number of epochs required by SGS-GNN to converge compared to other samplers under the same settings.

6.6 Impact of Sparsity and Homophily

We analyzed the performance of SGS-GNN across varying homophily levels using synthetic and benchmark graphs in Fig. 6. Synthetic graphs were generated with nodes of homophily $\mathcal{H}_n$ and degree d by connecting $\lceil d\mathcal{H}_n \rceil$ edges neighbors of the same class, and $d - \lceil d\mathcal{H}_n \rceil$ edges randomly. Results in Fig. 6a show that high homophily yields good performance regardless of sparsity, while high sparsity benefits heterophily, with optimal performance seen at $30\% - 40\%$ edge retention for heterophily levels $0.3 - 0.4$. Furthermore, Fig. 6b indicates that while more edges improve accuracy on homophilic graphs, high sparsity can be advantageous on heterophilic graphs such as reed98 and amherst41, where we observe best performance at $\sim 20\%$ sparsity. Fig. 7 shows that the sampled

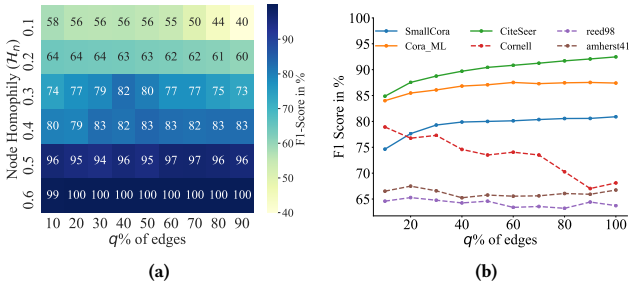


Figure 6: (Left) Heatmap of F1 scores (in %) at different homophily and sparsity levels (q) in Cora synthetic graphs. (Right) F1-scores of SGS-GNN at different sparsity for homophilic (solid line) and heterophilic (dashed line) graphs.

subgraphs of SGS-GNN have higher *edge homophily* than those from the fixed distribution unsupervised sparsifiers.

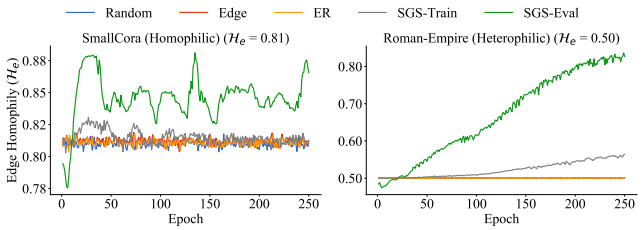


Figure 7: Edge homophily of selected subgraphs from different fixed distribution samplers vs. subgraphs from training and evaluation phase of SGS-GNN.

6.7 Ablation Studies

Impact of components (I). Table 8 shows performance of SGS-GNN with various combinations of $\mathcal{L}_{\text{assor}}$, $\mathcal{L}_{cons}$, EdgePE, GNN, and Conditional Updates (complete list at §16). **(I)** $\mathcal{L}_{\text{assor}}$**:** Cases 1, 2 show improvement in results when $\mathcal{L}_{\text{assor}}$ is used. **(II)** $\mathcal{L}_{\text{cons}}$**:** Cases 3, 4, 5 show$L_{\text{cons}}$ improves results when GCN module is used in the GNN. **(III) Conditional updates:** Case 3 shows that conditional updates can benefit some graphs. **(IV)** When **EdgePE** uses an MLP for edge-scoring embeddings, it relies solely on node features. In contrast, structure-aware alternatives (GCN/GAT) consistently outperform MLP-based EdgePE, demonstrating the importance of structure-aware edge-scoring. (cases 4, 8). **(V) GNN:** Both GCN and GAT modules yielded overall the best results (Cases 4, 8).

Table 8: EdgePE, GNN, Conditional update and regularizers.

C.	$L_{as.}$	$L_{cn.}$	EdgePE	GNN	Cond.	SmallCora	CoraFull	johns.
1	N	N	MLP	GCN	N	73.80 ± 0.67	61.78 ± 0.20	66.12 ± 1.38
2	Y	N	MLP	GCN	N	74.88 ± 0.15	63.99 ± 0.24	66.18 ± 1.05
3	Y	N	GCN	GCN	Y	75.80 ± 0.77	65.66 ± 0.14	71.06 ± 0.32
4	Y	Y	GCN	GCN	Y	77.50 ± 0.62	**66.56 ± 0.22**	70.79 ± 0.18
5	Y	Y	GCN	GCN	N	75.88 ± 0.71	65.87 ± 0.17	69.83 ± 0.67
6	Y	N	GSAGE	GCN	Y	75.82 ± 0.44	63.70 ± 0.09	67.53 ± 0.80
7	Y	Y	GSAGE	GCN	Y	77.48 ± 0.61	65.12 ± 0.11	68.63 ± 0.66
8	Y	N	GCN	GAT	Y	**78.18 ± 0.74**	66.33 ± 0.20	**71.97 ± 0.59**

Impact of components (II). Table 9 highlights the impact of p_{prior}, Normalization & Sampling schemes, and Ensembling on SGS-GNN. **(I) Prior** Cases 2-3 show that augmenting the learned probability distribution $\tilde{p}$ with prior p_{prior} can benefit some datasets. **(II) Normalization and Sampling:** Cases 3-5 show that each of the sampling techniques can improve results in certain datasets, and it isn't easy to nominate a single one as best. However, in our experiments, we opted for multinomial sampling with softmax temperature annealing for training to encourage exploration in early iterations. **(III) Ensemble subgraphs during inference:** Case 2 demonstrates that using multiple subgraphs for ensemble prediction yields better results than a single subgraph (Case 1).

SGS-GNN ***with and without Graph Partitioning.*** Partitioning large graphs is critical to solve large problems, avoiding the memory bottleneck of supervised classification methods. However, partitioning increases the training time. Additionally, the batch processing of subgraphs in the partitions prevents the method from being stuck in local minima, thereby improving accuracy (see Table 10). On small graphs, we found negligible effects of partitioning on SGS-GNN's prediction quality.

Table 9: Prior, Normalization, Sampling & Inference.

C.	*P.*	Norm.	Sampl.	Ens.	SmallCora	johns.	Roman.
1	N	Sum	Mult	N	69.30 ± 1.20	63.86 ± 0.58	63.27 ± 0.31
2	N	Sum	Mult	Y	72.84 ± 0.91	65.14 ± 1.14	**64.31 ± 0.13**
3	Y	Sum	Mult	Y	75.54 ± 0.41	**72.68 ± 0.51**	62.88 ± 0.19
4	Y	Soft	Mult	Y	75.44 ± 0.51	**72.97 ± 0.20**	62.98 ± 0.16
5	Y	Gumbel	TopK	Y	**76.24 ± 0.43**	71.83 ± 1.00	63.00 ± 0.11

P: Prior, **Sum:** Sum-Normalization, **Soft.:** *SoftMax* with temperature annealing.
Mult: *Multinomial*, **Gumbel:** *Gumbel-Softmax* with *TopK*, **Ens.** Ensemble inference.

Table 10: Training time (T) with and without METIS (#-Parts).

Dataset	without METIS		with METIS		#
	T/itr. (#itr.)	F1	T/itr. (#itr.)	F1	
SmallCora	0.01 (192)	**76.94±0.76**	0.08 (120)	76.27±0.29	5
arxiv-year	0.16 (52)	35.42±0.04	1.34 (109)	**37.17±0.51**	5

7 Conclusion

We proposed SGS-GNN, a supervised graph sparsifier that produces a sparse subgraph with user-prescribed sparsity, facilitating GNN computation on large-scale graphs. We provided a theoretical analysis of SGS-GNN in terms of the quality of embedding it produces compared to an idealized oracle sparsifier. Finally, we empirically validated the effectiveness, efficiency, and convergence of SGS-GNN over several baselines across homophilic and heterophilic graphs of various sizes. In the future, we plan to explore SGS-GNN on other graph learning tasks, as well as its robustness in the presence of noise.

Acknowledgments

This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research's Advanced Computing Technologies Competitive Portfolios program at Pacific Northwest National Laboratory (PNNL). PNNL is a multi-program national laboratory operated for the U.S. Department of Energy (DOE) by Battelle Memorial Institute under Contract No. DE-AC05-76RL01830. Alex Pothen's research was supported by grant SC-0022260 from the Advanced Scientific Computing Research program of the Office of Science, U.S. Department of Energy. The work of Danda B. Rawat was supported by the DoW Center of Excellence in AI/ML (CoE-AIML) at Howard University under Contract W911NF-20-2-0277 with the U.S. Army Research Laboratory. However, any opinions, findings, conclusions, or recommendations expressed in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the funding agency.

References

[1] Selahattin Akkas and Ariful Azad. 2025. Shapley-Value-Based Graph Sparsification for GNN Inference. *arXiv preprint arXiv:2507.20460* (2025).

[2] Joshua Batson, Daniel A Spielman, Nikhil Srivastava, and Shang-Hua Teng. 2013. Spectral sparsification of graphs: theory and algorithms. *Commun. ACM* 56, 8 (2013), 87–94.

[3] Sudhanshu Chanpuriya and Cameron Musco. 2022. Simplified graph convolution with heterophily. *Advances in neural information processing systems* 35 (2022), 27184–27197.

[4] Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang. 2021. A unified lottery ticket hypothesis for Graph Neural Networks. In *International conference on machine learning*. PMLR, 1695–1706.

[5] Yuhan Chen, Haojie Ye, Sanketh Vedula, Alex Bronstein, Ronald Dreslinski, Trevor Mudge, and Nishil Talati. 2023. Demystifying graph sparsification algorithms in graph properties preservation. *Proceedings of the VLDB Endowment* 17, 3 (2023), 427–440.

[6] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. 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*. 257–266.

[7] Siddhartha Shankar Das, SM Ferdous, Mahantesh M Halappanavar, Edoardo Serra, and Alex Pothen. 2024. AGS-GNN: Attribute-guided sampling for graph neural networks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 538–549.

[8] Haipeng Ding, Zhewei Wei, and Yuhang Ye. 2025. Large-Scale Spectral Graph Neural Networks via Laplacian Sparsification. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*. 213–223.

[9] Feodor F Dragan, Fedor V Fomin, and Petr A Golovach. 2011. Spanners in sparse graphs. *J. Comput. System Sci.* 77, 6 (2011), 1108–1119.

[10] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of The Web Conference 2020*. 2331–2341.

[11] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. 1998. CiteSeer: An automatic citation indexing system. In *Proceedings of the third ACM Conference on Digital Libraries*. 89–98.

[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.

[13] Mohammad Hashemi, Shengbo Gong, Juntong Ni, Wenqi Fan, B Aditya Prakash, and Wei Jin. 2024. A comprehensive survey on graph reduction: sparsification, coarsening, and condensation. *arXiv:2402.03358* (2024).

[14] Mingguo He, Zhewei Wei, and Ji-Rong Wen. 2022. Convolutional neural networks on graphs with chebyshev approximation, revisited. *Advances in Neural Information Processing Systems* 35 (2022), 7264–7276.

[15] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with Gumbel-Softmax. *arXiv:1611.01144* (2016).

[16] George Karypis. 1997. METIS: Unstructured graph partitioning and sparse matrix ordering system. *Technical report* (1997).

[17] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv:1609.02907* (2016).

[18] Jiayu Li, Tianyun Zhang, Hao Tian, Shengmin Jin, Makan Fardad, and Reza Zafarani. 2020. SGCN: A graph sparsifier based on graph convolutional networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer.

[19] Ningyi Liao, Zihao Yu, and Siqiang Luo. 2024. Unifews: Unified Entry-Wise Sparsification for Efficient Graph Neural Network. *arXiv preprint arXiv:2403.13268* (2024).

[20] Angelica Liguori, Ettore Ritacco, Pietro Sabatino, and Annalisa Socievole. 2025. Spectral Neural Graph Sparsification. *arXiv preprint arXiv:2510.27474* (2025).

[21] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. 2021. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. *Advances in Neural Information Processing Systems* 34 (2021), 20887–20902.

[22] Chuang Liu, Xueqi Ma, Yibing Zhan, Liang Ding, Dapeng Tao, Bo Du, Wenbin Hu, and Danilo P Mandic. 2023. Comprehensive graph gradual pruning for sparse training in graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[23] Zirui Liu, Kaixiong Zhou, Zhimeng Jiang, Li Li, Rui Chen, Soo-Hyun Choi, and Xia Hu. 2023. DSpar: An embarrassingly simple strategy for efficient GNN training and inference via degree-based sparsification. *Transactions on Machine Learning Research* (2023).

[24] Zirui Liu, Kaixiong Zhou, Zhimeng Jiang, Li Li, Rui Chen, Soo-Hyun Choi, and Xia Hu. 2023. DSpar: An embarrassingly simple strategy for efficient GNN training and inference via degree-based sparsification. *Transactions on Machine Learning Research* (2023).

[25] László Lovász. 1993. Random walks on graphs. *Combinatorics, Paul erdos is eighty* 2, 1-46 (1993), 4.

[26] Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. 2021. Learning to drop: Robust Graph Neural Network via topological denoising. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 779–787.

[27] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. 2021. Is homophily a necessity for graph neural networks? *arXiv preprint arXiv:2106.06134* (2021).

[28] Galileo Namata, Ben London, Lise Getoor, Bert Huang, and U Edu. 2012. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs*, Vol. 8. 1.

[29] Max Paulus, Dami Choi, Daniel Tarlow, Andreas Krause, and Chris J Maddison. 2020. Gradient estimation with stochastic softmax tricks. *Advances in Neural Information Processing Systems* 33 (2020), 5691–5704.

[30] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-GCN: Geometric graph convolutional networks. *arXiv:2002.05287* (2020).

[31] Oleg Platonov, Denis Kuznedelev, Artem Babenko, and Liudmila Prokhorenkova. 2022. Characterizing graph datasets for node classification: Beyond homophily-heterophily dichotomy. *arXiv:2209.06177* (2022).

[32] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. 2023. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? *arXiv:2302.11640* (2023).

[33] PyTorch Contributors. 2025. *torch.utils.checkpoint — PyTorch 2.10 documentation*. https://docs.pytorch.org/docs/stable/checkpoint.html Last updated Oct. 29, 2025.

[34] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2019. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv:1907.10903* (2019).

[35] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2021. Multi-scale attributed node embedding. *Journal of Complex Networks* 9, 2 (2021), cnab014.

[36] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[37] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv:1811.05868* (2018).

[38] Daniel A Spielman and Nikhil Srivastava. 2011. Graph sparsification by effective resistances. *SIAM J. Comput.* 40, 6 (2011), 1913–1926.

[39] Rakshith S Srinivasa, Cao Xiao, Lucas Glass, Justin Romberg, and Jimeng Sun. 2020. Fast graph attention networks using effective resistance based graph sparsification. *arXiv:2006.08796* (2020).

[40] Zhen Su, Yang Liu, Jürgen Kurths, and Henning Meyerhenke. 2024. Generic network sparsification via degree-and subgraph-based edge sampling. *Information Sciences* 679 (2024), 121096.

[41] Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. 2021. Adversarial graph augmentation to improve graph contrastive learning. *Advances in Neural Information Processing Systems* 34 (2021), 15920–15933.

[42] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv:1710.10903* (2017).

[43] Siyao Wang and Miles E Lopes. 2025. Empirical Error Estimates for Graph Sparsification. *arXiv preprint arXiv:2503.08031* (2025).

[44] Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao. 2022. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer Singapore, Singapore. 725 pages.

[45] Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. 2023. Sgformer: Simplifying and empowering transformers for large-graph representations. *Advances in neural information processing systems* 36 (2023), 64753–64773.

[46] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv:1810.00826* (2018).

[47] Xiaowei Xu, Nurcan Yuruk, Zhidan Feng, and Thomas AJ Schweiger. 2007. SCAN: A structural clustering algorithm for networks. In *Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining*. 824–833.

[48] Yang Ye and Shihao Ji. 2021. Sparse graph attention networks. *IEEE Transactions on Knowledge and Data Engineering* 35, 1 (2021), 905–916.

[49] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2019. GraphSAINT: Graph Sampling Based Inductive Learning Method. In *International Conference on Learning Representations*.

[50] Guibin Zhang, Xiangguo Sun, Yanwei Yue, Chonghe Jiang, Kun Wang, Tianlong Chen, and Shirui Pan. 2024. Graph Sparsification via Mixture of Graphs. *arXiv:2405.14260* (2024).

[51] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. 2020. Robust graph representation learning via neural sparsification. In *International Conference on Machine Learning*. PMLR, 11458–11468.

[52] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph Neural Networks: A review of methods and applications. *AI open* 1 (2020), 57–81.

[53] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems* 33 (2020), 7793–7804.

A Appendix

A.1 Notations

We dedicate Table 11 to index the notations used in this paper. Note that every notation is also defined when it is introduced.

Table 11: Notations.

$\mathcal{G}$	$\triangleq$	Input graph with a vertex set $\mathcal{V}$, an edge set $\mathcal{E}$, and features $\boldsymbol{X}$
$\boldsymbol{A}$	$\triangleq$	Adjacency matrix of $\mathcal{G}$
$\mathcal{E}$	$\triangleq$	Edges of $\mathcal{G}$
$\mathcal{V}$	$\triangleq$	Nodes of $\mathcal{G}$
$\boldsymbol{X}$	$\triangleq$	Matrix containing node features of $\mathcal{G}$
$\boldsymbol{y}$	$\triangleq$	Vector of node labels of $\mathcal{G}$
C	$\triangleq$	An ordered set containing all possible node labels of $\mathcal{G}$
F	$\triangleq$	Dimension of node features in $\mathcal{G}$
L	$\triangleq$	Number of GNN layers
H	$\triangleq$	Node embedding dimension
$\boldsymbol{H}$	$\triangleq$	Node embedding matrix
$\boldsymbol{h}_u$	$\triangleq$	Embedding of node u
$\boldsymbol{w}$	$\triangleq$	Vector of edge weights in $\mathcal{G}$
q	$\triangleq$	Ratio of # edges in sparse graph and # edges in input graph in %
k	$\triangleq$	# edges in the sparse graph, $k \triangleq \lfloor \frac{q\lvert\mathcal{E}\rvert}{100} \rfloor$
$\tilde{p}$	$\triangleq$	Learned probability distribution by SGS-GNN
$\tilde{\mathcal{E}}$	$\triangleq$	Set of edges sampled from $\mathcal{E}$ by SGS-GNN following $\tilde{p}$
$\tilde{\mathcal{G}}$	$\triangleq$	Sparse subgraph $(\mathcal{V}, \tilde{\mathcal{E}}, \boldsymbol{X})$ constructed by SGS-GNN
$\boldsymbol{A}_{\tilde{\mathcal{G}}}$ or $\tilde{\boldsymbol{A}}$	$\triangleq$	Adjacency matrix of $\tilde{\mathcal{G}}$
$\tilde{\boldsymbol{w}}$	$\triangleq$	Edge weight of sparse graph learned by SGS-GNN
p_{prior}	$\triangleq$	Probability distribution of a fixed prior on $\mathcal{G}$
$\tilde{p}_a$	$\triangleq$	Augmented learned probability distribution
p^*	$\triangleq$	True probability distribution known by the idealized learning ORACLE
$\mathcal{E}^*$	$\triangleq$	Set of edges sampled from $\mathcal{E}$ by the learning ORACLE following distribution p^*
$\mathcal{G}^*$	$\triangleq$	True sparse subgraph $(\mathcal{V}, \mathcal{E}^*, \boldsymbol{X})$ constructed by the learning ORACLE
$\boldsymbol{A}_{\mathcal{G}^*}$ or $\boldsymbol{A}^*$	$\triangleq$	Adjacency matrix of $\mathcal{G}^*$
$\mathcal{L}_{\text{CE}}$	$\triangleq$	Cross entropy loss
$\mathcal{L}_{\text{assor}}$	$\triangleq$	Assortative loss
$\mathcal{L}_{\text{cons}}$	$\triangleq$	Consistency loss
$\mathcal{L}$	$\triangleq$	Total loss

A.2 Homophily Measures

The *homophily* of a graph characterizes the likelihood that nodes with the same labels are neighbors. Some well-known definitions include *Node homophily* $\mathcal{H}_n$ [30], *Edge homophily* $\mathcal{H}_e$ [53], and *Adjusted Homophily* $\mathcal{H}_{\text{adj}}$ [31]:

$$\mathcal{H}_n = \frac{1}{|\mathcal{V}|} \sum_{u \in \mathcal{V}} \frac{|\{v \in \mathcal{N}(u) : y_v = y_u\}|}{|\mathcal{N}(u)|} \tag{10}$$

$$\mathcal{H}_e = \frac{(u,v) \in \mathcal{E} : y_u = y_v}{|\mathcal{E}|} \tag{11}$$

$$\mathcal{H}_{\text{adj}} = \frac{\mathcal{H}_e - \sum_{k=1}^{c} D_k^2/(2|\mathcal{E}|^2)}{1 - \sum_{k=1}^{c} D_k^2/2|\mathcal{E}|^2}. \tag{12}$$

Here $D_k = \sum_{v:y_v=k} d_v$ denotes the sum of degrees of the nodes belonging to class k. The values of the node homophily and the edge homophily range from 0 to 1, and the adjusted homophily ranges from $-\frac{1}{3}$ to +1. In this work, we classify graphs with $\mathcal{H}_{\text{adj}} \leq 0.50$ as heterophilic [7, 27].

B Theoretical Analysis

Assumptions. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ be the input graph with $m := |\mathcal{E}|$ edges. We assume there is an idealized learning ORACLE that knows the true distribution p^*, can sample edges from p^* given a budget k, and uses the constructed edge set $\mathcal{E}^* \subseteq \mathcal{E}$ as input to a downstream GNN. We assume the architecture of the ORACLE learners GNN to be the same as SGS-GNN.

We assume the learning ORACLE samples a size-k set $\mathcal{E}^* \subseteq \mathcal{E}$ without replacement from the true distribution $p^* = (p_1^*, \ldots, p_m^*)$ via successive sampling. Similarly, SGS-GNN samples a size-k set $\tilde{\mathcal{E}} \subseteq \mathcal{E}$ without replacement from distribution $\tilde{p} = (\tilde{p}_1, \ldots, \tilde{p}_m)$ via successive sampling. Sampling process of ORACLE and SGS-GNN are independent.

We assume that their probability distributions are ℓ_∞ bounded, meaning, there exists $\epsilon \in [0, 1]$ such that

$$\|p^* - \tilde{p}\|_\infty \leq \epsilon. \tag{13}$$

We work in the *diffuse/non-concentrated regime*: there exists $\rho \in (0, 1/4]$ such that

$$p^*_{\max} \triangleq \max_j p_j^* \leq \frac{\rho}{k}, \qquad \tilde{p}_{\max} \triangleq \max_j \tilde{p}_j \leq \frac{\rho}{k}. \tag{14}$$

This condition ensures no single edge has $\Omega(1/k)$ mass and thus renormalization effects from weighted sampling without replacement remain small.

B.1 #Overlapping Edges wrt. a Learning ORACLE

For each edge $e_j \in \mathcal{E}$, let us define inclusion probabilities

$$\pi_j^* \triangleq \Pr(e_j \in \mathcal{E}^*), \qquad \tilde{\pi}_j \triangleq \Pr(e_j \in \tilde{\mathcal{E}}).$$

B.1.1 Relating π to p. By independence between the ORACLE and SGS-GNN sampler,

$$\mathbb{E}\big[|\mathcal{E}^* \cap \tilde{\mathcal{E}}|\big] = \sum_{j=1}^{m} \pi_j^* \, \tilde{\pi}_j. \tag{15}$$

Thus, the key is to relate the inclusion probabilities π to the base weights p.

Lemma B.1 (Inclusion probability approximation with absolute error). *Let p be any distribution over $\mathcal{E}$ and let $\mathcal{E}^{(p)}$ be the size-k successive-sampling sample. Write $\pi_j(p) \triangleq \Pr(e_j \in \mathcal{E}^{(p)})$ and $p_{\max} \triangleq \max_j p_j$. If $(k-1)p_{\max} \leq 1/4$, then for every j,*

$$\Big|\pi_j(p) - k p_j\Big| \;\leq\; 4\,k(k-1)\,p_j\,p_{\max}. \tag{16}$$

Proof. Let E_t denote the edge selected at step t and $R_t \triangleq \mathcal{E} \setminus \{E_1, E_2, \cdots, E_{t-1}\}$ the random remaining set before step t. By successive sampling,

$$\Pr(E_t = e_j \mid R_t) = \mathbf{1}\{e_j \in R_t\} \frac{p_j}{\sum_{e_\ell \in R_t} p_\ell}. \tag{17}$$

Since at most $t-1$ edges have been removed before step t, and each removed edge has mass at most $p_{\max}$, the remaining mass satisfies

$$1 - (t-1)p_{\max} \leq \sum_{e_\ell \in R_t} p_\ell \leq 1.$$

Thus, combining this with equation (17), for every realization of R_t,

$$\mathbf{1}\{e_j \in R_t\}\, p_j \;\leq\; \Pr(E_t = e_j \mid R_t) \;\leq\; \mathbf{1}\{e_j \in R_t\}\, \frac{p_j}{1 - (t-1)p_{\max}}.$$

Taking expectation yields

$$p_j \,\Pr(e_j \in R_t) \;\leq\; \Pr(E_t = e_j) \;\leq\; \frac{p_j}{1 - (t-1)p_{\max}} \Pr(e_j \in R_t). \tag{18}$$

Lower-bounding $\Pr(e_j \in R_t)$**.** By disjointness of sampling without replacement, $\Pr(e_j \notin R_t) = \sum_{s=1}^{t-1} \Pr(E_s = e_j)$. Hence $\Pr(e_j \in R_t) =$

$1 - \sum_{s=1}^{t-1} \Pr(E_s = e_j) \le 1$. Since $(s-1)p_{\max} \le (k-1)p_{\max} \le 1/4$, Equation (18) applied to event $E_s = e_j$ and the fact $\Pr(e_j \in R_s) \le 1$ yields

$$\Pr(E_s = e_j) \le \frac{p_j}{1-(s-1)p_{\max}} \Pr(e_j \in R_s) \le \frac{p_j}{1-(k-1)p_{\max}} \cdot 1 \le \frac{4}{3} p_j.$$

Hence,

$$\Pr(e_j \in R_t) = 1 - \sum_{s=1}^{t-1} \Pr(E_s = e_j) \ge 1 - \sum_{s=1}^{t-1} \frac{4}{3} p_j = 1 - (t-1)\frac{4}{3} p_j.$$

We now quantify the deviation of $\Pr(E_t = e_j)$ from p_j. From Equation (18),

$$\Pr(E_t = e_j) \ge p_j \Pr(e_j \in R_t) \ge p_j \left(1 - (t-1)\frac{4}{3} p_j\right) = p_j - (t-1)\frac{4}{3} p_j^2. \tag{19}$$

Upper-bounding $\Pr(e_j \in R_t)$**.** For the upper bound, using $\Pr(e_j \in R_t) \le 1$ in Equation (18):

$$\Pr(E_t = e_j) \le \frac{p_j}{1-(t-1)p_{\max}}. \tag{20}$$

Using $\frac{1}{1-x} \le 1 + 2x$ for $x \in [0, 1/4]$ with $x = (t-1)p_{\max}$ gives

$$\frac{p_j}{1-(t-1)p_{\max}} \le p_j\big(1 + 2(t-1)p_{\max}\big) = p_j + 2(t-1)p_j p_{\max}.$$

Combining with (19), and by defining

$$\Delta_{j,t} \triangleq \Pr(E_t = e_j) - p_j,$$

the deviation $\Delta_{j,t}$ satisfies

$$-\frac{4}{3}(t-1)p_j^2 \;\le\; \Delta_{j,t} \;\le\; 2(t-1)p_j p_{\max}.$$

Upper-bounding $|\Delta_{j,t}|$. We apply to the above equation $-a \le x \le b \implies |x| \le (a+b)$ for real values $a, b > 0$:

$$|\Delta_{j,t}| \le 2(t-1)p_j p_{\max} + \frac{4}{3}(t-1)p_j^2. \tag{21}$$

Since $p_j \le p_{\max}$, we have $p_j^2 \le p_j p_{\max}$, and (21) becomes

$$|\Delta_{j,t}| \le \left(2 + \frac{4}{3}\right)(t-1)p_j p_{\max} = \frac{10}{3}(t-1)p_j p_{\max} \le 4(t-1)p_j p_{\max}.$$

Obtaining $|\pi_j(p) - kp_j|$. By definition of inclusion probability and disjointness of the events $\{E_t = e_j\}_{t=1}^k$,

$$\pi_j(p) = \Pr(e_j \in \mathcal{E}^{(p)}) = \sum_{t=1}^{k} \Pr(E_t = e_j) = \sum_{t=1}^{k} (p_j + \Delta_{j,t}) = kp_j + \sum_{t=1}^{k} \Delta_{j,t}.$$

Applying triangle inequality,

$$\left|\pi_j(p) - kp_j\right| = \left|\sum_{t=1}^{k} \Delta_{j,t}\right| \le \sum_{t=1}^{k} |\Delta_{j,t}| \le 4p_j p_{\max} \sum_{t=1}^{k} (t-1) = 4k(k-1)p_j p_{\max},$$

□

B.1.2 *Overlap lower bound between ORACLE and* SGS-GNN.

Lemma B.2 (Sampling overlap).

$$\mathbb{E}\big[|\mathcal{E}^* \cap \tilde{\mathcal{E}}|\big] \;\ge\; k^2 \sum_{j=1}^{m} p_j^* \tilde{p}_j \;-\; C k^2 \rho \left(\sum_{j=1}^{m} p_j^* \tilde{p}_j\right), \tag{22}$$

for a universal constant $C > 0$*. In particular, in the diffuse regime*

$$\mathbb{E}\big[|\mathcal{E}^* \cap \tilde{\mathcal{E}}|\big] \ge k^2 (1 - 12\rho) \left(\sum_{j=1}^{m} p_j^* \tilde{p}_j\right) = k^2 (1 - 12\rho) \sum_{j=1}^{m} \frac{(p_j^* + \tilde{p}_j - \epsilon)^2}{4}. \tag{23}$$

Proof. Lemma B.1 applied to p^* and $\tilde{p}$ yields

$$\pi_j^* = kp_j^* + a_j, \qquad |a_j| \le 4k(k-1)p_j^* p_{\max}^*,$$
$$\tilde{\pi}_j = k\tilde{p}_j + b_j, \qquad |b_j| \le 4k(k-1)\tilde{p}_j \tilde{p}_{\max}. \tag{24}$$

Then

$$\mathbb{E}\big[|\mathcal{E}^* \cap \tilde{\mathcal{E}}|\big] = \sum_{j=1}^{m} \pi_j^* \tilde{\pi}_j = k^2 \sum_{j=1}^{m} p_j^* \tilde{p}_j + k \sum_{j=1}^{m} p_j^* b_j + k \sum_{j=1}^{m} \tilde{p}_j a_j + \sum_{j=1}^{m} a_j b_j. \tag{25}$$

We bound the three remainder terms.

(i) Bounding $k \sum_{j=1}^{m} p_j^* b_j$*.* Since $x \ge -|x|,\ \forall x \in \mathbb{R}$ and $p_j^* \ge 0$,

$$p_j^* b_j \ge -p_j^* |b_j|.$$

Summing over j and multiplying by k yields

$$k \sum_{j=1}^{m} p_j^* b_j \;\ge\; -k \sum_{j=1}^{m} p_j^* |b_j|. \tag{26}$$

Substituting the bound on $|b_j|$ from Equation (24):

$$\sum_{j=1}^{m} p_j^* |b_j| \;\le\; \sum_{j=1}^{m} p_j^* \Big(4k(k-1)\,\tilde{p}_j\,\tilde{p}_{\max}\Big) = 4k(k-1)\,\tilde{p}_{\max} \sum_{j=1}^{m} p_j^* \tilde{p}_j.$$

Plugging this into (26) gives

$$k \sum_{j=1}^{m} p_j^* b_j \;\ge\; -4k^2(k-1)\,\tilde{p}_{\max} \sum_{j=1}^{m} p_j^* \tilde{p}_j. \tag{27}$$

(ii) Bounding $k \sum_{j=1}^{m} \tilde{p}_j a_j$*.* Since $x \ge -|x|,\ \forall x \in \mathbb{R}$, and $p_j^* \ge 0$,

$$\tilde{p}_j a_j \ge -\tilde{p}_j |a_j|,$$

so

$$k \sum_{j=1}^{m} \tilde{p}_j a_j \;\ge\; -k \sum_{j=1}^{m} \tilde{p}_j |a_j|. \tag{28}$$

Using $|a_j| \le 4k(k-1)p_j^* p_{\max}^*$ from (24),

$$\sum_{j=1}^{m} \tilde{p}_j |a_j| \;\le\; \sum_{j=1}^{m} \tilde{p}_j \Big(4k(k-1)\,p_j^*\,p_{\max}^*\Big) = 4k(k-1)\,p_{\max}^* \sum_{j=1}^{m} p_j^* \tilde{p}_j.$$

Plugging into (28) yields

$$k \sum_{j=1}^{m} \tilde{p}_j a_j \;\ge\; -4k^2(k-1)\,p_{\max}^* \sum_{j=1}^{m} p_j^* \tilde{p}_j. \tag{29}$$

(iii) Bounding $\sum_{j=1}^{m} a_j b_j$. We bound the magnitude of the product term and then sum:

$$\sum_{j=1}^{m} a_j b_j \;\geq\; -\sum_{j=1}^{m} |a_j b_j| \;\geq\; -\sum_{j=1}^{m} |a_j|\,|b_j|.$$

Using (24) for both factors,

$$\begin{aligned}|a_j|\,|b_j| &\leq \Big(4k(k-1)\,p_j^*\,p_{\max}^*\Big)\Big(4k(k-1)\,\tilde{p}_j\,\tilde{p}_{\max}\Big)\\ &= 16k^2(k-1)^2\,p_{\max}^*\tilde{p}_{\max}\,(p_j^*\tilde{p}_j).\end{aligned}$$

Summing over j gives

$$\sum_{j=1}^{m} |a_j b_j| \;\leq\; 16k^2(k-1)^2\,p_{\max}^*\tilde{p}_{\max}\sum_{j=1}^{m} p_j^*\tilde{p}_j, \tag{30}$$

and therefore

$$\sum_{j=1}^{m} a_j b_j \;\geq\; -16k^2(k-1)^2\,p_{\max}^*\tilde{p}_{\max}\sum_{j=1}^{m} p_j^*\tilde{p}_j. \tag{31}$$

Remainder term. Under (14), we have $p_{\max}^* \leq \rho/k$ and $\tilde{p}_{\max} \leq \rho/k$, therefore each of the remainder term's coefficients:

$$\begin{aligned}4k^2(k-1)\,\tilde{p}_{\max} &\;\leq\; 4k^2(k-1)\,\frac{\rho}{k} = 4\rho\,k(k-1) \leq 4\rho\,k^2,\\ 4k^2(k-1)\,p_{\max}^* &\;\leq\; 4\rho\,k(k-1) \leq 4\rho\,k^2,\end{aligned}$$

$$16k^2(k-1)^2 p_{\max}^*\tilde{p}_{\max} \;\leq\; 16k^2(k-1)^2\frac{\rho^2}{k^2} \leq 16\rho^2k^2.$$

Since $\rho \leq 1/4$, we have $\rho^2 \leq \rho/4$, hence $16\rho^2 \leq 4\rho$, and therefore

$$16k^2(k-1)^2\,p_{\max}^*\tilde{p}_{\max} \leq 4\rho\,k^2.$$

Final term. Plugging the simplified remainder term lower-bounds into Equation (25) yields

$$\begin{aligned}\mathbb{E}\big[|\mathcal{E}^* \cap \tilde{\mathcal{E}}|\big] &= \sum_{j=1}^{m} \pi_j^*\,\tilde{\pi}_j\\ &\geq k^2\sum_{j=1}^{m} p_j^*\tilde{p}_j - (4+4+4)\rho\,k^2\sum_{j=1}^{m} p_j^*\tilde{p}_j\\ &= k^2(1-12\,\rho)\sum_{j=1}^{m} p_j^*\tilde{p}_j\\ &\geq k^2(1-12\,\rho)\sum_{j=1}^{m}\frac{(p_j^*+\tilde{p}_j-|p_j^*-\tilde{p}_j|)^2}{4}\\ &\geq k^2(1-12\,\rho)\sum_{j=1}^{m}\frac{(p_j^*+\tilde{p}_j-\epsilon)^2}{4}.\end{aligned}$$

□

B.2 Discrepency in the Learned Adjacency Matrix

With the bound proven earlier on the #common edges by the sparse subgraph of SGS-GNN with that by a learning ORACLE, in this section, we want to obtain an upper-bound on the error in terms of the norm of the Adjacency matrices. As adjacency matrices are used by GNNs to compute node embeddings, this result is important for establishing an error bound on the embeddings later.

Let $\boldsymbol{A}_{\tilde{\mathcal{G}}}$ and $\boldsymbol{A}_{\mathcal{G}^*}$ be the corresponding adjacency matrices of the learned sparse graph $\tilde{\mathcal{G}}$ and true optimal sparse graph $\mathcal{G}^*$. The dimension of these matrices is the same as the input adjacency matrix $\boldsymbol{A}_{\mathcal{G}}$ except that $\boldsymbol{A}_{\mathcal{G}}$ is denser. Let us also denote the Frobenius norm of a matrix $\boldsymbol{A}$ as $\|\boldsymbol{A}\|_F$ and the spectral norm of $\boldsymbol{A}$ as $\|\boldsymbol{A}\|_2$. The Frobenius norm of $\boldsymbol{A}$ is defined as $\sqrt{\sum_{ij}\boldsymbol{A}_{ij}^2}$, whereas the spectral norm of $\boldsymbol{A}$ is the largest singular value $\sigma_{max}(\boldsymbol{A})$ of $\boldsymbol{A}$.

Since SGS-GNN do not know the true probability distribution p^*, error is introduced in the learned adjacency matrix $\boldsymbol{A}_{\tilde{\mathcal{G}}}$ of the downstream sparse subgraph. We are interested in analyzing the expected error introduced in $\boldsymbol{A}_{\tilde{\mathcal{G}}}$ in terms of the spectral norm, to be precise, $\mathbb{E}[\|\boldsymbol{A}_{\tilde{\mathcal{G}}} - \boldsymbol{A}_{\mathcal{G}^*}\|_2]$. To this end, we will exploit the lower bound derived in Theorem 1 and the fact that $\|\boldsymbol{A}\|_2 \leq \|\boldsymbol{A}\|_F$.

Lemma B.3 (Error in Adjacency matrix approximation). *Let $\boldsymbol{A}_{\tilde{\mathcal{G}}}$ and $\boldsymbol{A}_{\mathcal{G}^*}$ be the corresponding adjacency matrices of the learned sparse graph $\tilde{\mathcal{G}}$ and true optimal sparse graph $\mathcal{G}^*$. Let the samplers sample k edges successively, without replacement following their respective distributions $\tilde{p}$ and p^*. Let us assume the approximation error $\|p^* - \tilde{p}\|_\infty \leq \epsilon$, then in the diffuse regime: there exists $\rho \in (0, 1/4]$ such that*

$$p_{\max}^* \leq \frac{\rho}{k}, \qquad \tilde{p}_{\max} \leq \frac{\rho}{k}.$$

Then

$$\mathbb{E}[\|\boldsymbol{A}_{\tilde{\mathcal{G}}} - \boldsymbol{A}_{\mathcal{G}^*}\|_2] \leq 2\Delta_{\text{sam}}(p^*, \tilde{p}),$$

where $k = \lfloor q|\mathcal{E}|/100 \rfloor$ with $0 \leq q \leq 100$ is a user-specified parameter and $\Delta_{\text{sam}}(p^*, \tilde{p}) \triangleq \sqrt{k\left(1 - k(1-12\rho)\sum_{j=1}^{m}\frac{(p_j^*+\tilde{p}_j-\epsilon)^2}{4}\right)}$. We term $\Delta_{\text{sam}}(p^*, \tilde{p})$ as sampling disagreement radius. .

Proof. Since the entries in adjacency matrices are either 0 or 1, the difference $\boldsymbol{A}_{\tilde{\mathcal{G}}}(i,j) - \boldsymbol{A}_{\mathcal{G}^*}(i,j)$ are in $\{-1, 0, 1\}$ for all i, j. The following holds by definition of Frobenus norm,

$$\|\boldsymbol{A}_{\tilde{G}} - \boldsymbol{A}_{G^*}\|_F^2 = \sum_{ij}(\boldsymbol{A}_{\tilde{\mathcal{G}}}(i,j) - \boldsymbol{A}_{\mathcal{G}^*}(i,j))^2.$$

As a result, only the non-zero entries in $\boldsymbol{A}_{\tilde{\mathcal{G}}} - \boldsymbol{A}_{\mathcal{G}^*}$ contribute to the square of Frobenius norm $\|\boldsymbol{A}_{\tilde{G}} - \boldsymbol{A}_{G^*}\|_F^2$. Due to symmetry the number of non-zero entries in $\|\boldsymbol{A}_{\tilde{\mathcal{G}}} - \boldsymbol{A}_{\mathcal{G}^*}\|_F^2$ corresponds to $2 \times |(\tilde{\mathcal{E}} \setminus \mathcal{E}^*) \cup (\mathcal{E}^* \setminus \tilde{\mathcal{E}})|$. Thus

$$\begin{aligned}\mathbb{E}[\|\boldsymbol{A}_{\tilde{\mathcal{G}}} - \boldsymbol{A}_{\mathcal{G}^*}\|_F^2] &= 2\mathbb{E}[|(\tilde{\mathcal{E}} \setminus \mathcal{E}^*) \cup (\mathcal{E}^* \setminus \tilde{\mathcal{E}})|]\\ &= 2\mathbb{E}[|\tilde{\mathcal{E}}| + |\mathcal{E}^*| - 2|\mathcal{E}^* \cap \tilde{\mathcal{E}}|]\\ &= 4k - 4\mathbb{E}[|\tilde{\mathcal{E}} \cap \mathcal{E}^*|]\end{aligned}$$

Applying Jensen's inequality $(\mathbb{E}[\mathrm{X}])^2 \leq \mathbb{E}[\mathrm{X}^2])$ yields,

$$\begin{aligned}(\mathbb{E}[\|\boldsymbol{A}_{\tilde{\mathcal{G}}} - \boldsymbol{A}_{\mathcal{G}^*}\|_F])^2 &\leq \mathbb{E}[\|\boldsymbol{A}_{\tilde{\mathcal{G}}} - \boldsymbol{A}_{\mathcal{G}^*}\|_F^2]\\ &= 4k - 4\mathbb{E}[|\tilde{\mathcal{E}} \cap \mathcal{E}^*|]\end{aligned}$$

Taking square-root on both sides yields,

$$\mathbb{E}[\|\boldsymbol{A}_{\tilde{\mathcal{G}}} - \boldsymbol{A}_{\mathcal{G}^*}\|_F] \leq 2\sqrt{k - \mathbb{E}[|\tilde{\mathcal{E}} \cap \mathcal{E}^*|]}$$

We obtain the theorem using the following relation between the Frobenius and spectral norms.

$$\|\boldsymbol{A}_{\tilde{\mathcal{G}}} - \boldsymbol{A}_{\mathcal{G}^*}\|_2 \leq \|\boldsymbol{A}_{\tilde{\mathcal{G}}} - \boldsymbol{A}_{\mathcal{G}^*}\|_F$$

$$\mathbb{E}[\|\boldsymbol{A}_{\tilde{\mathcal{G}}} - \boldsymbol{A}_{\mathcal{G}^*}\|_2] \le \mathbb{E}[\|\boldsymbol{A}_{\tilde{\mathcal{G}}} - \boldsymbol{A}_{\mathcal{G}^*}\|_F]$$
$$\le 2\sqrt{k - k^2\,(1-12\rho)\sum_{j=1}^{m}\frac{(p_j^* + \tilde{p}_j - \epsilon)^2}{4}}$$
$$= 2\sqrt{k\left(1 - k(1-12\rho)\sum_{j=1}^{m}\frac{(p_j^* + \tilde{p}_j - \epsilon)^2}{4}\right)}$$
$$= 2\Delta_{\text{sam}}(p^*, \tilde{p}).$$

Remarks. (I) For $\epsilon > 0$, the factor $1 - k(1-12\rho)\sum_{j=1}^{m}\frac{(p_j^*+\tilde{p}_j-\epsilon)^2}{4}$ is upper bounded by $\rho(1-12\rho)$, so the square root argument is nonnegative. (II) The sampling disagreement radius $\Delta_{\text{sam}}(p^*, \tilde{p})$ quantifies the intrinsic variability between two independently sampled sparse graphs induced by successive sampling, and acts as a radius in operator norm around the oracle adjacency within which the learned adjacency concentrates. □

B.3 Discrepancy in the Node Embeddings

We consider vanilla GCN as proof of concept to understand how the changes in the sparse subgraph affect the node embeddings produced by a GCN. Our goal is to analyze the respective encodings produced by an L-layer GCN when the input subgraphs are $\mathcal{G}^*$ (corresponding to $\boldsymbol{A}_{\mathcal{G}^*}$) and $\tilde{\mathcal{G}}$ (corresponding to $\boldsymbol{A}_{\tilde{\mathcal{G}}}$), respectively. For simplicity, we will shorten the matrices $\boldsymbol{A}_{\mathcal{G}^*}$ as $\boldsymbol{A}^*$ and $\boldsymbol{A}_{\tilde{\mathcal{G}}}$ as $\tilde{\boldsymbol{A}}$.

A single GCN layer is defined as,

$$\boldsymbol{H}^{(l+1)} = \sigma(\hat{\boldsymbol{A}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}),$$

where $\hat{\boldsymbol{A}} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ is the normalized adjacency matrix, $\boldsymbol{H}^{(l)}$ is the input to the l-th layer with $\boldsymbol{H}^{(0)} = \boldsymbol{X}$, $\boldsymbol{W}^{(l)}$ is the learnable weight matrix for l-th layer and σ is non-linear activation function. Let us suppose an L-layer GCN produces embeddings $\tilde{\boldsymbol{H}}^{(L)}$ and $\boldsymbol{H}^{*(L)}$ when it takes sparse matrices $\tilde{\boldsymbol{A}}$ and $\boldsymbol{A}^*$ as input. We want to upper-bound,

$$\mathbb{E}[\|\tilde{\boldsymbol{H}}^{(L)} - \boldsymbol{H}^{*(L)}\|_2],$$

in other words, the loss in the downstream node encodings is due to using our learned subgraph.

Assumptions. We assume that for all l, $\|\boldsymbol{W}\|_2 \le \alpha < 1$ where α is a constant no more than 1. This is reasonable since each $\boldsymbol{W}^{(l)}$ is typically controlled during training using regularization techniques, e.g., weight decay. Assuming that the input features in $\boldsymbol{X}$ are bounded, we can also assume that there exists a constant β such that $\forall l$, $\|\boldsymbol{H}\|_2^{(l)} \le \beta$. We also assume that σ is *Lipschitz continuous* with *Lipschitz constant* L_σ; for instance, activation functions such as ReLU, sigmoid, or tanH are Lipschitz continuous. In particular, we assume ReLU activation for our theoretical analysis because ReLU has *Lipschitz constant* $L_\sigma = 1$, which simplifies our analysis.

We proof the following lemma to be used later.

Lemma B.4 (Discrepancy in normalized adjacencies). *Let $\tilde{\mathcal{G}} = (\mathcal{V}, \tilde{\mathcal{E}}, \boldsymbol{X})$ and $\mathcal{G}^* = (\mathcal{V}, \mathcal{E}^*, \boldsymbol{X})$ be two simple undirected graphs on the same vertex set, with $|\tilde{\mathcal{E}}| = |\mathcal{E}^*| = k$, sampled* without replacement *via successive sampling from $\tilde{p}$ and p^*, respectively, and the two samplers are independent. Let $\hat{\tilde{\boldsymbol{A}}}$ and $\hat{\boldsymbol{A}}^*$ be the normalized adjacency matrices:*

$$\hat{\tilde{\boldsymbol{A}}} \triangleq \tilde{\boldsymbol{D}}^{-1/2}(\tilde{\boldsymbol{A}} + \boldsymbol{I})\tilde{\boldsymbol{D}}^{-1/2},$$
$$\hat{\boldsymbol{A}}^* \triangleq (\boldsymbol{D}^*)^{-1/2}(\boldsymbol{A}^* + \boldsymbol{I})(\boldsymbol{D}^*)^{-1/2},$$

with corresponding diagonal degree matrices $\tilde{\boldsymbol{D}} \triangleq \text{diag}\big((\tilde{\boldsymbol{A}} + \boldsymbol{I})\boldsymbol{1}\big)$ and $\boldsymbol{D}^* \triangleq \text{diag}\big((\boldsymbol{A}^* + \boldsymbol{I})\boldsymbol{1}\big)$. Assume the uniform approximation error $\|p^* - \tilde{p}\|_\infty \le \epsilon$ and the diffuse regime: there exists $\rho \in (0, 1/4]$ such that $p^*_{\max} \le \frac{\rho}{k}$ and $\tilde{p}_{\max} \le \frac{\rho}{k}$. Moreover, assume a uniform minimum-degree condition (after adding self-loops):

$$d_{\min} \triangleq \min_{v\in\mathcal{V}} \min\{d_v(\tilde{\mathcal{G}}) + 1,\ d_v(\mathcal{G}^*) + 1\} \ \ge\ 1. \tag{32}$$

Then

$$\mathbb{E}\Big[\|\hat{\tilde{\boldsymbol{A}}} - \hat{\boldsymbol{A}}^*\|_2\Big] \le \min\{2, \frac{2}{d_{\min}}\Delta_{\text{sam}}(p^*, \tilde{p}) + \frac{4(d^*_{\max}+1)}{d_{\min}^{3/2}}\Delta_{\text{sam}}(p^*, \tilde{p})^2\}, \tag{33}$$

where the sampling disagreement radius is

$$\Delta_{\text{sam}}(p^*, \tilde{p}) \triangleq \sqrt{k\left(1 - k(1-12\rho)\sum_{j=1}^{m}\frac{(p_j^* + \tilde{p}_j - \epsilon)^2}{4}\right)}.$$

Proof. For notational clarity, define the self-looped adjacencies

$$\tilde{\boldsymbol{A}}^+ \triangleq \tilde{\boldsymbol{A}} + \boldsymbol{I}, \qquad \boldsymbol{A}^{*+} \triangleq \boldsymbol{A}^* + \boldsymbol{I},$$

and the corresponding degree matrices

$$\tilde{\boldsymbol{D}} \triangleq \text{diag}(\tilde{\boldsymbol{A}}^+\boldsymbol{1}), \qquad \boldsymbol{D}^* \triangleq \text{diag}(\boldsymbol{A}^{*+}\boldsymbol{1}).$$
$$\tilde{\boldsymbol{S}} \triangleq \tilde{\boldsymbol{D}}^{-1/2}, \qquad \boldsymbol{S}^* \triangleq (\boldsymbol{D}^*)^{-1/2},$$

Hence,

$$\hat{\tilde{\boldsymbol{A}}} = \tilde{\boldsymbol{S}}\tilde{\boldsymbol{A}}^+\tilde{\boldsymbol{S}}, \qquad \hat{\boldsymbol{A}}^* = \boldsymbol{S}^*\boldsymbol{A}^{*+}\boldsymbol{S}^*.$$

We add and subtract $\tilde{\boldsymbol{S}}\boldsymbol{A}^{*+}\tilde{\boldsymbol{S}}$:

$$\hat{\tilde{\boldsymbol{A}}} - \hat{\boldsymbol{A}}^* = \tilde{\boldsymbol{S}}(\tilde{\boldsymbol{A}}^+ - \boldsymbol{A}^{*+})\tilde{\boldsymbol{S}} + (\tilde{\boldsymbol{S}} - \boldsymbol{S}^*)\boldsymbol{A}^{*+}\tilde{\boldsymbol{S}} + \boldsymbol{S}^*\boldsymbol{A}^{*+}(\tilde{\boldsymbol{S}} - \boldsymbol{S}^*)$$
$$= \tilde{\boldsymbol{S}}(\tilde{\boldsymbol{A}} - \boldsymbol{A}^*)\tilde{\boldsymbol{S}} + (\tilde{\boldsymbol{S}} - \boldsymbol{S}^*)\boldsymbol{A}^{*+}\tilde{\boldsymbol{S}} + \boldsymbol{S}^*\boldsymbol{A}^{*+}(\tilde{\boldsymbol{S}} - \boldsymbol{S}^*)$$

By Submultiplicity of induced operator norm,

$$\|\hat{\tilde{\boldsymbol{A}}} - \hat{\boldsymbol{A}}^*\|_2 \le \|\tilde{\boldsymbol{S}}\|_2^2\,\|\tilde{\boldsymbol{A}} - \boldsymbol{A}^*\|_2 + \|\tilde{\boldsymbol{S}} - \boldsymbol{S}^*\|_2\,\|\boldsymbol{A}^*\|_2\,\|\tilde{\boldsymbol{S}}\|_2 + \|\boldsymbol{S}^*\|_2\,\|\boldsymbol{A}^*\|_2\,\|\tilde{\boldsymbol{S}} - \boldsymbol{S}^*\|_2 \tag{34}$$

By (32), all diagonal entries of $\tilde{\boldsymbol{D}}$ and $\boldsymbol{D}^*$ are at least $d_{\min}$, hence

$$\|\tilde{\boldsymbol{S}}\|_2 = \|\tilde{\boldsymbol{D}}^{-1/2}\|_2 \le \frac{1}{\sqrt{d_{\min}}}, \qquad \|\boldsymbol{S}^*\|_2 \le \frac{1}{\sqrt{d_{\min}}}.$$

Since $\tilde{\boldsymbol{S}}$ and $\boldsymbol{S}^*$ are diagonal,

$$\|\tilde{\boldsymbol{S}} - \boldsymbol{S}^*\|_2 = \max_{v\in\mathcal{V}}\left|\frac{1}{\sqrt{\tilde{d}_v}} - \frac{1}{\sqrt{d_v^*}}\right|, \quad \tilde{d}_v \triangleq (\tilde{\boldsymbol{D}})_{vv},\ d_v^* \triangleq (\boldsymbol{D}^*)_{vv}.$$

The function $f(x) = x^{-1/2}$ satisfies $|f'(x)| = \frac{1}{2}x^{-3/2} \le \frac{1}{2}d_{\min}^{-3/2}$ for all $x \ge d_{\min}$. Therefore, by the mean value theorem,

$$\|\tilde{\boldsymbol{S}} - \boldsymbol{S}^*\| \le \frac{1}{2}d_{\min}^{-3/2}\,\|\tilde{\boldsymbol{D}} - \boldsymbol{D}^*\|_2. \tag{35}$$

Moreover, $\tilde{\boldsymbol{D}} - \boldsymbol{D}^*$ is diagonal, hence $\|\tilde{\boldsymbol{D}} - \boldsymbol{D}^*\|_2 = \max_v |\tilde{d}_v - d_v^*|$.

Each mismatched undirected edge in $\tilde{\mathcal{E}}\triangle\mathcal{E}^*$ changes degrees of its two endpoints by 1. Thus, for every v, $|\tilde{d}_v - d_v^*| \le 2|\tilde{\mathcal{E}}\triangle\mathcal{E}^*|$. In particular,

$$\|\tilde{\boldsymbol{D}} - \boldsymbol{D}^*\|_2 = \max_v |\tilde{d}_v - d_v^*| \le 2|\tilde{\mathcal{E}}\triangle\mathcal{E}^*| = 4\big(k - |\tilde{\mathcal{E}} \cap \mathcal{E}^*|\big).$$

Taking expectations gives

$$\mathbb{E}\left[\|\tilde{\boldsymbol{D}} - \boldsymbol{D}^*\|_2\right] \leq 4\left(k - \mathbb{E}[|\tilde{\mathcal{E}} \cap \mathcal{E}^*|]\right). \tag{36}$$

Substituting $\|\boldsymbol{S}^*\|_2$ into Equation (34) yields

$$\begin{aligned}
&\|\hat{\tilde{\boldsymbol{A}}} - \hat{\boldsymbol{A}}^*\|_2 \\
&\leq \|\tilde{\boldsymbol{S}}\|_2^2 \|\tilde{\boldsymbol{A}} - \boldsymbol{A}^*\|_2 + \|\tilde{\boldsymbol{S}} - \boldsymbol{S}^*\|_2 \|\boldsymbol{A}^*\|_2 \|\tilde{\boldsymbol{S}}\|_2 + \|\boldsymbol{S}^*\|_2 \|\boldsymbol{A}^*\|_2 \|\tilde{\boldsymbol{S}} - \boldsymbol{S}^*\|_2 \\
&\leq \frac{1}{d_{\min}} \|\tilde{\boldsymbol{A}} - \boldsymbol{A}^*\|_2 + \frac{2}{\sqrt{d_{\min}}} \|\boldsymbol{A}^*\|_2 \|\tilde{\boldsymbol{S}} - \boldsymbol{S}^*\|_2.
\end{aligned}$$

Since $\|\boldsymbol{A}^*\|_2$ is symmetric, $\|\boldsymbol{A}^*\|_2 \leq \|\boldsymbol{A}^*\|_\infty$. The maximum absolute row-sum of $\boldsymbol{A}^*$ is $d^*_{\max} + 1$. Thus $\|\boldsymbol{A}^*\|_2 \leq d^*_{\max} + 1$. Using (35) yields

$$\|\hat{\tilde{\boldsymbol{A}}} - \hat{\boldsymbol{A}}^*\|_2 \leq \frac{1}{d_{\min}} \|\tilde{\boldsymbol{A}} - \boldsymbol{A}^*\|_2 + \frac{d^*_{\max} + 1}{d^{3/2}_{\min}} \|\tilde{\boldsymbol{D}} - \boldsymbol{D}^*\|_2. \tag{37}$$

Taking expectations and applying (36) gives

$$\begin{aligned}
&\mathbb{E}\left[\|\hat{\tilde{\boldsymbol{A}}} - \hat{\boldsymbol{A}}^*\|_2\right] \\
&\leq \frac{1}{d_{\min}} \mathbb{E}\left[\|\tilde{\boldsymbol{A}} - \boldsymbol{A}^*\|_2\right] + \frac{4(d^*_{\max} + 1)}{d^{3/2}_{\min}} \left(k - \mathbb{E}[|\tilde{\mathcal{E}} \cap \mathcal{E}^*|]\right).
\end{aligned}$$

By Lemma B.2 in the diffuse regime,

$$\mathbb{E}[\|\tilde{\boldsymbol{A}} - \boldsymbol{A}^*\|_2] \leq 2\,\Delta_{\text{sam}}(p^*, \tilde{p}),$$

and,

$$k - \mathbb{E}[|\tilde{\mathcal{E}} \cap \mathcal{E}^*|] \leq k - k^2(1-12\rho) \sum_{j=1}^{m} \frac{(p^*_j + \tilde{p}_j - \epsilon)^2}{4} = \Delta_{\text{sam}}(p^*, \tilde{p})^2.$$

Thus,

$$\mathbb{E}\left[\|\hat{\tilde{\boldsymbol{A}}} - \hat{\boldsymbol{A}}^*\|_2\right] \leq \frac{2}{d_{\min}} \Delta_{\text{sam}}(p^*, \tilde{p}) + \frac{4(d^*_{\max} + 1)}{d^{3/2}_{\min}} \Delta_{\text{sam}}(p^*, \tilde{p})^2.$$

Also $E\left[\|\hat{\tilde{\boldsymbol{A}}} - \hat{\boldsymbol{A}}^*\|_2\right] \leq E\left[\|\hat{\tilde{\boldsymbol{A}}}\|_2 + \|\hat{\boldsymbol{A}}^*\|_2\right] \leq 2$, since both $\|\hat{\tilde{\boldsymbol{A}}}\|_2$ and $\|\hat{\boldsymbol{A}}^*\|_2$ are degree-normalized adjacency matrix with maximum eigenvalue at most 1. Hence,

$$\begin{aligned}
&\mathbb{E}\left[\|\hat{\tilde{\boldsymbol{A}}} - \hat{\boldsymbol{A}}^*\|_2\right] \\
&\leq \min\{2, \frac{2}{d_{\min}} \Delta_{\text{sam}}(p^*, \tilde{p}) + \frac{4(d^*_{\max} + 1)}{d^{3/2}_{\min}} \Delta_{\text{sam}}(p^*, \tilde{p})^2\}.
\end{aligned}$$

□

Theorem 5.2. *(Error in GCN encodings) Let $\tilde{\boldsymbol{H}}$ and $\boldsymbol{H}^*$ be the corresponding learned node encodings from an L-layer GCN trained on the sparse graph $\tilde{\mathcal{G}}$ and true optimal sparse graph $\mathcal{G}^*$, respectively. Let the samplers sample k edges successively, without replacement following their respective distributions $\tilde{p}$ and p^*. Let us assume the approximation error $\|p^* - \tilde{p}\|_\infty \leq \epsilon$, such that in the diffuse (non-concentrated) regime: there exists $\rho \in (0, 1/4]$ such that $p^*_{\max} \leq \frac{\rho}{k}$, $\tilde{p}_{\max} \leq \frac{\rho}{k}$. If for all layer ℓ, GCN weights $\|\boldsymbol{W}^{(\ell)}\|_2 \leq \alpha$, $\alpha \in [0, 1]$ and hidden states $\|\tilde{\boldsymbol{H}}^{(\ell)}\|_2, \|\boldsymbol{H}^{*(\ell)}\|_2 \leq \beta$, $\beta > 0$, then for sufficiently large L,*

$$\begin{aligned}
&\mathbb{E}[\lim_{L\to\infty} \|\tilde{\boldsymbol{H}}^{(L)} - \boldsymbol{H}^{*(L)}\|_2] \\
&\leq \frac{\beta\alpha}{1-\alpha} \min\{2, \frac{2}{d_{\min}} \Delta_{\text{sam}}(p^*, \tilde{p}) + \frac{4(d^*_{\max} + 1)}{d^{3/2}_{\min}} \Delta_{\text{sam}}(p^*, \tilde{p})^2\},
\end{aligned}$$

where $d^*_{\max}$ is the maximum degree in $\mathcal{G}^*$, $\Delta_{\text{sam}}(p^*, \tilde{p})$ is the sampling disagreement radius

$$\Delta_{\text{sam}}(p^*, \tilde{p}) \triangleq \sqrt{k\left(1 - k(1-12\rho) \sum_{j=1}^{m} \frac{(p^*_j + \tilde{p}_j - \epsilon)^2}{4}\right)}.$$

Proof.

$$\tilde{\boldsymbol{H}}^{(L)} - \boldsymbol{H}^{*(L)} = \sigma(\hat{\tilde{\boldsymbol{A}}} \tilde{\boldsymbol{H}}^{(L-1)} \boldsymbol{W}^{(L-1)}) - \sigma(\hat{\boldsymbol{A}}^* \boldsymbol{H}^{*(L-1)} \boldsymbol{W}^{(L-1)})$$

Since σ is a Lipschitz continuous function, we have

$$\begin{aligned}
\|\tilde{\boldsymbol{H}}^{(L)} - \boldsymbol{H}^{*(L)}\|_2 &\leq L_\sigma \|\hat{\tilde{\boldsymbol{A}}} \tilde{\boldsymbol{H}}^{(L-1)} \boldsymbol{W}^{(L-1)} - \hat{\boldsymbol{A}}^* \boldsymbol{H}^{*(L-1)} \boldsymbol{W}^{(L-1)}\|_2 \\
&= \|\hat{\tilde{\boldsymbol{A}}} \tilde{\boldsymbol{H}}^{(L-1)} \boldsymbol{W}^{(L-1)} - \hat{\boldsymbol{A}}^* \boldsymbol{H}^{*(L-1)} \boldsymbol{W}^{(L-1)}\|_2 \\
= \|(\hat{\tilde{\boldsymbol{A}}} - \hat{\boldsymbol{A}}^*) \tilde{\boldsymbol{H}}^{(L-1)} \boldsymbol{W}^{(L-1)} + \hat{\boldsymbol{A}}^* (\tilde{\boldsymbol{H}}^{(L-1)} - \boldsymbol{H}^{*(L-1)}) \boldsymbol{W}^{(L-1)}\|_2
\end{aligned}$$

For notational convenience, suppose that $D^{(L)} = \|\tilde{\boldsymbol{H}}^{(L)} - \boldsymbol{H}^{*(L)}\|_2$. Applying the sub-multiplicative property of the spectral norm and triangle inequality, we obtain the following recurrence relation

$$\begin{aligned}
D^{(L)} &\leq \|(\hat{\tilde{\boldsymbol{A}}} - \hat{\boldsymbol{A}}^*)\|_2 \|\tilde{\boldsymbol{H}}^{(L-1)}\|_2 \|\boldsymbol{W}^{(L-1)}\|_2 + \|\hat{\boldsymbol{A}}^*\|_2 D^{(L-1)} \|\boldsymbol{W}^{(L-1)}\|_2 \\
&\leq \|(\hat{\tilde{\boldsymbol{A}}} - \hat{\boldsymbol{A}}^*)\|_2 \beta\alpha + \|\hat{\boldsymbol{A}}^*\|_2 D^{(L-1)} \alpha \\
&\leq \|(\hat{\tilde{\boldsymbol{A}}} - \hat{\boldsymbol{A}}^*)\|_2 \beta\alpha + D^{(L-1)} \alpha
\end{aligned}$$

The last inequality holds because the degree-normalized adjacency matrix satisfies $\|\hat{\boldsymbol{A}}^*\|_2 \leq 1$, as it is symmetric positive semi-definite with largest eigenvalue of $\hat{\boldsymbol{A}}^*$ at most 1. By unrolling the recursion from the earlier inequality:

$$D^{(L)} \leq \|(\hat{\tilde{\boldsymbol{A}}} - \hat{\boldsymbol{A}}^*)\|_2 \beta\alpha \sum_{l=0}^{L-1} \alpha^l + D^{(0)} \alpha^L$$

$D^{(0)} = \|\tilde{\boldsymbol{H}}^{(0)} - \boldsymbol{H}^{*(0)}\|_2 = \|\boldsymbol{X} - \boldsymbol{X}\|_2 = 0$. Since $\alpha < 1$, The geometric series simplifies to:

$$\sum_{l=0}^{L-1} \alpha^l = \frac{1 - \alpha^L}{1 - \alpha}$$

$$\lim_{L\to\infty} \sum_{l=0}^{L-1} \alpha^l = \frac{1}{1 - \alpha}$$

Thus our earlier inequality becomes:

$$\lim_{L\to\infty} D^{(L)} \leq \frac{\beta\alpha}{1-\alpha} \|(\hat{\tilde{\boldsymbol{A}}} - \hat{\boldsymbol{A}}^*)\|_2$$

Taking expectation on both sides gives us our desired result:

$$\begin{aligned}
&\mathbb{E}[\lim_{L\to\infty} \|\tilde{\boldsymbol{H}}^{(L)} - \boldsymbol{H}^{*(L)}\|_2] = \mathbb{E}[D^{(L}] < \frac{\beta\alpha}{1-\alpha} \mathbb{E}[\|(\hat{\tilde{\boldsymbol{A}}} - \hat{\boldsymbol{A}}^*)\|_2] \\
&\leq \frac{\beta\alpha}{1-\alpha} \min\{2, \frac{2}{d_{\min}} \Delta_{\text{sam}}(p^*, \tilde{p}) + \frac{4(d^*_{\max} + 1)}{d^{3/2}_{\min}} \Delta_{\text{sam}}(p^*, \tilde{p})^2\}
\end{aligned}$$

□

B.4 Impact of Degree Prior p_{prior}.

Theorem 5.1. *If the simple undirected sparsified graph $\tilde{\mathcal{G}}$ is formed by successively sampling without replacement k edges from $\tilde{p}_a$, then for every node $u \in \mathcal{V}$, the probability of its degree being 0 in $\tilde{\mathcal{G}}$*

$$\Pr[d_{u,\tilde{\mathcal{G}}} = 0] \;\le\; \exp\left(-\frac{k(1-\lambda)}{2|\mathcal{V}|}\right).$$

Proof. Let us fix a node $u \in \mathcal{V}$ and define the set of edges incident to u as $E_u \triangleq \{e \in \mathcal{E} : u \in e\}$. Let π_u be the total mass concentrated at E_u:

$$\pi_u \triangleq \sum_{e \in E_u} \tilde{p}_a(e).$$

We first lower bound π_u using only the degree prior.

By definition of $\tilde{p}_a$ and since $\tilde{p}(e), p_{\text{prior}}(e) \ge 0$,

$$\pi_u = \sum_{e \in E_u} \tilde{p}_a(e) \;\ge\; (1-\lambda) \sum_{e \in E_u} p_{\text{prior}}(e).$$

Let $N(u)$ be the neighbors of u in $\mathcal{G}$ and $d_u = |N(u)|$. Then

$$\sum_{e=(u,v)\in E_u} p_{\text{prior}}(u,v) = \frac{\sum_{v\in N(u)} \left(\frac{1}{d_u} + \frac{1}{d_v}\right)}{\sum_{(i,j)\in\mathcal{E}} \left(\frac{1}{d_i} + \frac{1}{d_j}\right)} = \frac{d_u \cdot \frac{1}{d_u} + \sum_{v\in N(u)} \frac{1}{d_v}}{\sum_{(i,j)\in\mathcal{E}} \frac{1}{d_i} + \sum_{(i,j)\in\mathcal{E}} \frac{1}{d_j}}.$$

For the denominator, note that in an undirected graph

$$\sum_{(i,j)\in\mathcal{E}} \frac{1}{d_i} = \sum_{i\in\mathcal{V}} \sum_{j:(i,j)\in\mathcal{E}} \frac{1}{d_i} = \sum_{i\in\mathcal{V}} d_i \cdot \frac{1}{d_i} = |\mathcal{V}|,$$

and with a similar argument, $\sum_{(i,j)\in\mathcal{E}} \frac{1}{d_j} = |\mathcal{V}|$. Hence the denominator equals $2|\mathcal{V}|$. Since $\sum_{v\in N(u)} \frac{1}{d_v} \ge 0$, we obtain

$$\sum_{e\in I_u} p_{\text{prior}}(e) \ge \frac{1}{2|\mathcal{V}|}.$$

Therefore,

$$\pi_u \ge \frac{1-\lambda}{2|\mathcal{V}|}. \tag{38}$$

Next, we upper-bound the probability that u is never selected under the sequential weighted sampling procedure.

Let E_t denote the remaining (unsampled) edges before draw t, with $E_1 = E$. Let us define the total unassigned probability mass for all nodes at draw t as S_t and total unassigned probability mass for node u at draw t as A_t:

$$S_t \triangleq \sum_{e\in E_t} \tilde{p}_a(e), \qquad A_t \triangleq \sum_{e\in E_u\cap E_t} \tilde{p}_a(e).$$

Clearly $S_t \le A_t$, since the event $E_u \cap E_t \subseteq E_t$. Thus the conditional probability that the t-th draw selects an edge incident to u is

$$\Pr(\text{draw } t \in E_u \mid E_t) = \frac{A_t}{S_t}.$$

We bound the probability of not selecting an edge incident to u at draw t by

$$\Pr(\text{draw } t \notin E_u \mid E_t) = 1 - \frac{A_t}{S_t}.$$

Now we condition on the event that no incident edge has been selected in the first $t-1$ draws. Under this conditioning, only edges not in E_u have been removed so far, thus the total unassigned probability mass for node u remains unchanged. Hence $A_t = A_1 = \pi_u$. And $S_t \le 1$ since probability mass cannot exceed 1. Therefore, on the event "no sample edge is incident on u up to draw $t-1$",

$$\frac{A_t}{S_t} \ge \frac{\pi_u}{1} = \pi_u.$$

So,

$$\Pr(\text{draw } t \notin E_u \mid \text{no sample incident on } u \text{ so far}) \le 1 - \pi_u.$$

Applying the chain rule for conditional probabilities,

$$\begin{aligned}
&\Pr(d_{u,\tilde{\mathcal{G}}} = 0)\\
&= \Pr(\text{no incident edge selected in draws } 1,\dots,k)\\
&= \prod_{t=1}^{k} \Pr(\text{draw } t \notin E_u \mid \text{no sample incident on } u \text{ in draws } 1,\dots,t-1)\\
&\le (1-\pi_u)^k.
\end{aligned}$$

Using the lower bound Equation (38) on π_u,

$$\Pr(d_{u,\tilde{\mathcal{G}}} = 0) \le \left(1 - \frac{1-\lambda}{2|\mathcal{V}|}\right)^k.$$

Finally, applying $(1-x)^k \le e^{-kx}$ for $x \in [0,1]$ yields

$$\Pr(d_{u,\tilde{\mathcal{G}}} = 0) \le \exp\left(-\frac{k(1-\lambda)}{2|\mathcal{V}|}\right).$$

□

Implication. (i) Even if the learned distribution $\tilde{p}$ forgets a low-degree bridge node (say u), the degree prior augmentation ensures that the probability of u becoming isolated is low. This directly supports our claim that augmenting $\tilde{p}$ with p_{prior} reduces the probability that low-degree / bridge nodes become isolated, which in turn helps preserve potential bridging connections. (ii) The constant $2|\mathcal{V}|$ arises from globally normalizing the prior. The bound can be tightened by noting that $1 + \sum_{v\in N(u)} \frac{1}{d_v} \ge 1 + \frac{d_u}{d_{\max}}$, where $d_{\max}$ is the maximum node degree in $\mathcal{G}$ and using the normalization constant $Z_d \triangleq \sum_{(i,j)\in\mathcal{E}} (1/d_i + 1/d_j)$. The upper-bound then becomes tighter: $\Pr[d_{u,\tilde{\mathcal{G}}} = 0] \le \exp\left(-k(1-\lambda)\frac{1+\frac{d_u}{d_{\max}}}{Z_d}\right)$.

C Comparison between Related Sparsifiers

Table 12 shows a comparison of representative graph sparsification methods in terms of the components. It contrasts unsupervised and supervised approaches in terms of sparsity control, sampling mechanism, and memory behavior during message passing. Unsupervised methods typically rely on heuristics or structural priors and construct sparse graphs without considering downstream tasks. Supervised methods leverage the downstream task's training loss to identify informative edges but often retain gradients of the entire edge list. In contrast, SGS-GNN explicitly constructs sparse subgraphs with global sparsity control, restricts gradient flow to sampled edges, and incorporates priors and regularizers for improved efficiency and performance.

Table 12: Component-wise comparison of supervised and unsupervised graph sparsification methods discussed in this paper.

Component / Property	DropEdge [34]	DSpar [23]	FastGAT [39]	MoG [50]	SparseGAT [48]	SGCN [18]	AD-GCL [41]	NeuralSparse [51]	SGS-GNN (Ours)
High-level categorization									
Supervision available in design	✗	✗	✗	✗	✓	✓	✓	✓	✓
Used primarily as unsupervised	✓	✓	✓	✓	✗	✗	✗	✗	✗
Type (Implicit = full-graph mask/regularizer)	✗	✗	✗	✗	✓	✗	✓	✗	✗
Type (Explicit = constructs sparse graph for message passing)	✓	✓	✓	✓	✗	✓	✗	✓	✓
Control & sampling									
Global sparsity control	✓	✓	✓	✓	✗	✓	✗	✗	✓
Local sparsity control	✗	✗	✗	✗	✗	✗	✗	✓	✗
Sampler is heuristic (no learned scorer)	✓	✓	✓	✓	✗	✓	✗	✗	✗
Learned edge scorer	✗	✗	✗	✗	✓	✗	✓	✓	✓
Differentiable sampling relaxation	✗	✗	✗	✗	✓	✗	✓	✓	✗
Memory / gradient flow (SpMM relevance)									
Backprop requires full-graph edge tensors/masks	✗	✗	✗	✗	✓	✗	✓	✓	✗
Backprop can be restricted to sampled edges (explicit)	✓	✓	✓	✓	✗	✓	✗	✗	✓
SpMM memory savings during GNN forward/backward	✓	✓	✓	✓	✗	✓	✗	✗	✓
Priors, regularizers, and SGS-specific components									
Uses spectral / ER-inspired prior or approximation	✗	✓	✓	✓	✗	✗	✗	✗	✓
Structure-aware scorer	✗	✗	✗	✗	✗	✗	✗	✗	✓
Consistency / stability regularizer across samples	✗	✗	✗	✗	✗	✗	✗	✗	✓
Conditional update of scorer to reduce compute	✗	✗	✗	✗	✗	✗	✗	✗	✓
Gradient checkpointing	✗	✗	✗	✗	✗	✗	✗	✗	✓

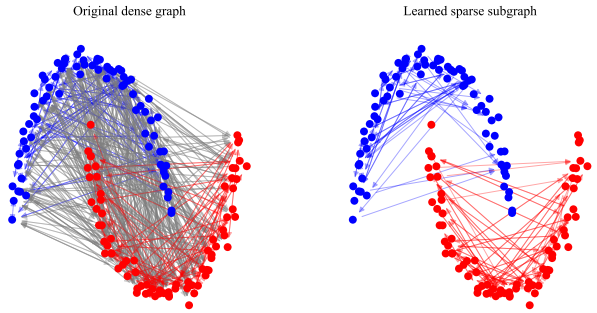


Figure 8: Toy example with two half-moons demonstrates the effectiveness of SGS-GNN. The original graph has 68% edges with different node labels; in contrast, the learned sparse subgraph from SGS-GNN contains no such bridge edges.

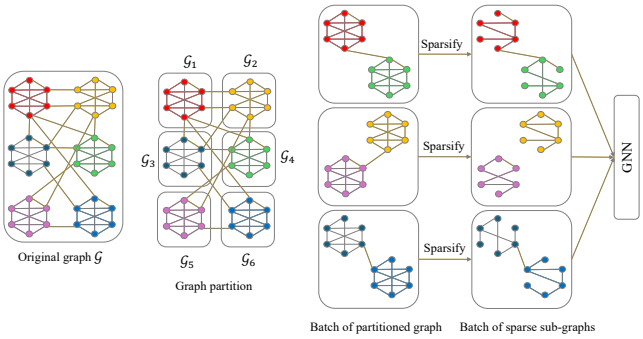


Figure 9: Demonstration of how graph partitioning works in conjunction with SGS-GNN.

D Analyzing the Effectiveness of SGS-GNN with a Synthetic Graph

In this section, we demonstrate and analyze the effectiveness of SGS-GNN with a synthetically generated heterophilic graph.

Synthetic Graph: Moon. The *Moon* dataset has the following properties: number of nodes $|\mathcal{V}| = 150$, number of edges $|\mathcal{E}| = 870$, average degree $d = 5.8$, node homophily $\mathcal{H}_n = 0.2$, edge homophily $\mathcal{H}_e = 0.32$, training/test split = 30%/70%, and 2D coordinates of the points representing the nodes are the node features $\boldsymbol{X}$. The dataset comprises two half-moons representing two communities with 68% edges connecting them as bridge edges.

Explaining the Effectiveness of SGS-GNN *on Heterophilic graph.* Fig. 8 juxtaposes the input moon graph (Fig. 8, left) and the sparsified moon graph by SGS-GNN (Fig. 8, right). SGS-GNN removes a significant portion of bridge edges, causing an increase in edge homophily from 0.32 to 1.0. As a result, the accuracy of vanilla GCN increased from 80% on the full graph to 100% on the sparsified graph. Since heterophilous edges significantly hinder the node representation learning, SGS-GNN identifies them during training and learns to put less probability mass on such edges for downstream node classification. Due to this learning dynamics, SGS-GNN is more effective on heterophilic graphs such as the Moon graph.

E Additional algorithmic details of SGS-GNN

METIS based graph partition. Note that we used METIS instead of spectral clustering or KMeans because it directly produces balanced graph partitions while minimizing edge cuts using an efficient multilevel coarsen-refine scheme, which preserves dense intra-cluster structure and reduces cross-cluster edges at scale. In contrast, spectral clustering typically requires computing (approximate) eigenvectors of the graph Laplacian, which becomes expensive and memory-heavy on large sparse graphs. KMeans optimizes within-cluster Euclidean variance in a feature space, so without a strong embedding it can miss graph connectivity and does not target the edge-cut objective that METIS optimizes.

Conditional update of **EdgePE**. The detailed algorithm for SGS-GNN with conditional updates is in Alg. 3. Backward propagation is often the most computationally intensive part of our

Algorithm 4 SGS-GNN Inference

Input: Graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, sample % q, Ensemble size R.
$\boldsymbol{w}, \tilde{p} = \texttt{EdgePE}_\phi(\mathcal{E}, \boldsymbol{X}, T_{\text{best}})$ {T **with best validation accuracy**.}
$S_y \leftarrow \emptyset$ {Predictions}
for i in R **do**
  $\tilde{\mathcal{E}} \leftarrow \text{Sample}(\mathcal{E}_i, \tilde{p}_a, \lfloor \frac{q|\mathcal{E}|}{100} \rfloor)$ {**Learned sparse subgraph**}
  $\tilde{\boldsymbol{w}} \leftarrow \boldsymbol{w}[\tilde{\mathcal{E}}]$ {i.e., $\tilde{w}(e_{uv}) = w(e_{uv})\ \forall (u,v) \in \tilde{\mathcal{E}}$.}
  $\tilde{Y}_i \leftarrow \text{GNN}_\theta(\tilde{\mathcal{E}}, \tilde{\boldsymbol{w}}, \boldsymbol{X})$
  $S_y \leftarrow S_y \cup \tilde{Y}_i$
end for
Predict, $\tilde{Y} \leftarrow \text{Mean}(S_y)$
Return $\tilde{Y}$

training, so we employ a conditional mechanism to update EdgePE selectively. We evaluate the learned sparse subgraph (line 9, Alg. 3) against a subgraph from the prior probability distribution p_{prior} (line 11, Alg. 3). If the training F1-score from the learned sparse subgraph is better than the baseline, parameters of EdgePE are updated (line 19, Alg. 3). Otherwise, the update to EdgePE is skipped (line 22, Alg. 3).

Algorithm 3 SGS-GNN Training with conditional updates

1: **Input:** $\mathcal{G}(\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, sample percent q, hops, #Parts n
2: Compute $p_{\text{prior}}(u,v) \leftarrow \frac{1/d_u + 1/d_v}{\sum_{i,j\in\mathcal{E}}(1/d_i + 1/d_j)}$
3: $\mathcal{G}_{\text{parts}} = \{\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_n\} \leftarrow \text{METIS}(\mathcal{G}(\mathcal{V}, \mathcal{E}, p), n)$
4: **for** epoch = $1, 2, \ldots,$ max_epochs **do**
5: **for** $\mathcal{G}_i(\mathcal{V}_i, \mathcal{E}_i, \boldsymbol{X}_i, p^i_{\text{prior}}) \in \mathcal{G}_{\text{parts}}$ **do**
6: $\tilde{p}, \boldsymbol{w} \leftarrow \texttt{EdgePE}_\phi(\mathcal{E}_i, p^i_{\text{prior}}, \boldsymbol{X}_i, \text{hops})$
7: $\tilde{p}_a \leftarrow \lambda\tilde{p} + (1-\lambda)p^i_{\text{prior}}$
8: $\tilde{\mathcal{E}} \leftarrow \text{Sample}(\mathcal{E}_i, \tilde{p}_a, \lfloor \frac{q|\mathcal{E}|}{100} \rfloor)$ {**Learned sparse subgraph**}
9: $\tilde{\boldsymbol{w}} \leftarrow \boldsymbol{w}[\tilde{\mathcal{E}}]$ {i.e., $\tilde{w}(e_{uv}) = w(e_{uv})\ \forall (u,v) \in \tilde{\mathcal{E}}$.}
10: $\hat{Y}, \tilde{\boldsymbol{H}} \leftarrow \text{GNN}_\theta(\tilde{\mathcal{E}}, \tilde{\boldsymbol{w}}, \boldsymbol{X})$
11: $\mathcal{E}_{\text{prior}} \leftarrow \text{Sample}(\mathcal{E}_i, p_i, \lfloor \frac{q|\mathcal{E}_i|}{100} \rfloor)$ {**Sparse subgraph from prior**}
12: $\hat{Y}_{\text{prior}} \leftarrow \text{GNN}_\theta(\mathcal{E}_{\text{prior}}, \boldsymbol{X})$
13: **if** Evaluate$(\hat{Y}) \geq$ Evaluate$(\hat{Y}_{\text{prior}})$ **then**
14: $\mathcal{L}_{\text{CE}} \leftarrow \text{CrossEntropy}(Y_{\mathcal{V}_L}, \hat{Y}_{\mathcal{V}_L})$
15: $\forall_{(u,v)\in\mathcal{E}_i} u \in \mathcal{V}_L \wedge v \in \mathcal{V}_L$: mask[u, v] $\leftarrow$ True
16: $\mathcal{L}_{\text{assor}} \leftarrow \text{CrossEntropy}(\mathcal{E}[\text{mask}], \boldsymbol{w}[\text{mask}])$
17: $\mathcal{L}_{\text{cons}} \leftarrow \text{Sim}(\boldsymbol{w}, \text{Cosine}(\boldsymbol{h}_u, \boldsymbol{h}_v) : \forall_{(u,v)\in\mathcal{E}})$
18: $\mathcal{L} \leftarrow \alpha_1 \cdot \mathcal{L}_{\text{CE}} + \alpha_2 \cdot \mathcal{L}_{\text{assor}} + \alpha_3 \cdot \mathcal{L}_{\text{cons}}$
19: Backward Propagate through $\mathcal{L}$ and optimize $\texttt{EdgePE}_\phi, \text{GNN}_\theta$.
20: **else**
21: $\mathcal{L}_{\text{CE}} \leftarrow \text{CrossEntropy}(Y_{\mathcal{V}_L}, \hat{Y}_{\mathcal{V}_L})$
22: Backward Propagate through $\mathcal{L}_{\text{CE}}$ and optimize GNN_θ.
23: **end if**
24: **end for**
25: **end for**
26: **Return** $\texttt{EdgePE}_\phi, \text{GNN}_\theta$

Inference. During inference, we use the learned probability distribution from EdgePE. We keep track of the best temperature T that gave the best validation accuracy and use that to sample an ensemble of sparse subgraphs. Then, we mean-aggregate their representations to produce the final prediction on a test node. The reason we consider ensemble of subgraphs is because there are variability in the edges of the sample subgraphs even if they are all sampled from the same distribution. Thus mean-aggregation of node embeddings is an effective way to improve the robustness of the learned node embeddings.

The inference pseudocode is provided in Algorithm 4.

F Dataset Description

Table 13 shows the details of the characteristics of the graph datasets, including the splits used throughout the experimentation.

Along with synthetic dataset, for heterophily, we used, *Cornell, Texas*, *Wisconsin* from the *WebKB* [30]; *Chameleon*, *Squirrel* [35]; *Actor* [30]; *Wiki, ArXiv-year, Snap-Patents, Penn94, Pokec, Genius, reed98, amherst41, cornell5*, and *Yelp* [21]. We also experiment on some recent benchmark datasets, *Roman-empire, Amazon-ratings, Minesweeper, Tolokers*, and *Questions* from [32].

For homophily, we used *Cora* [36]; *Citeseer* [11]; *pubmed* [28]; *Coauthor-cs*, *Coauthor-physics* [37]; *Amazon-computers*, *Amazon-photo* [37]; *Reddit* [12]; and, *DBLP* [10].

G SGS-GNN vs. other Related Works

Table 14 compares the performance of SGS-GNN against other related baselines: two heterophilic GNNs: H2GCN [53] and SGC [3], one lottery ticket-based method: UGT-GCN [4], and two sparsifiers: CGP-GCN [22] and DSPar [24]. We report F1-score with the rank in parentheses, where rank 1 denotes the best valid F1-score among the compared methods on that dataset. The average-rank row summarizes these per-dataset ranks across datasets; lower average rank indicates better overall performance. OOM and compilation-error cases are not assigned ranks and are excluded from the corresponding average-rank computation.

Our results indicate that SGS-GNN outperforms these sparsifiers while remaining competitive with heterophilic GNNs (H2GCN and SGC) that use full graphs, even if in this experiment SGS-GNN used homophilic GNN. Moreover, SGS-GNN can handle large graphs (e.g., Pokec, Reddit) while CGP-GCN, UGT-GCN, SparseGAT, MoG, and NeuralSparse ran out of memory in our experiments.

To further strengthen the experimental evaluation, we include SGFormer [45] as a recent strong full-graph GNN baseline, and UNIFEWS [19] and SGNN-LS [8] as recent graph sparsification baselines. We also compare against GraphSAINT with random-walk sampling. Table 15 reports the F1-score with the rank in parentheses. Overall, SGS-GNN achieves the best average rank among these baselines, while remaining competitive on homophilic graphs and particularly strong on heterophilic graphs.

H Ablation Studies

This section investigates how different components of SGS-GNN behave and contribute to overall performance. We organize this section as follows,

Table 13: Additional details of the dataset are provided. $\mathcal{H}_{\text{adj}}$ refers to adjusted homophily. d corresponds to the average degree, C number of classes, and F is the feature dimension. *Tr.* is the training label rate. *SL.* - If a self-loop exists. *Iso.* - If any isolated vertices.

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	d	$\mathcal{H}_{\text{adj}}$	C	F	Tr.	SL.	Iso.	Context
Cornell	183	557	3.04	-0.42	5	1703	0.48	Y	N	Web Pages
Texas	183	574	3.14	-0.26	5	1703	0.48	Y	N	Web Pages
Wisconsin	251	916	3.65	-0.20	5	1703	0.48	Y	N	Web Pages
reed98	962	37,624	39.11	-0.10	3	1001	0.6	N	N	Social Network
amherst41	2,235	181,908	81.39	-0.07	3	1193	0.6	N	N	Social Network
penn94	41,554	2,724,458	65.56	-0.06	2	4814	0.47	N	N	Social Network
Roman-empire	22,662	65,854	2.91	-0.05	18	300	0.5	N	N	Wikipedia
cornell5	18,660	1,581,554	84.76	-0.04	3	4735	0.6	N	N	Web pages
Squirrel	5,201	396,846	76.30	-0.01	5	2345	0.48	Y	N	Wikipedia
johnshopkins55	5,180	373,172	72.04	0.00	3	2406	0.6	N	N	Web Pages
Actor	7,600	53,411	7.03	0.01	5	932	0.48	Y	N	Actors in Movies
Minesweeper	10,000	78,804	7.88	0.01	2	7	0.5	N	N	Synthetic
Questions	48,921	307,080	6.28	0.02	2	301	0.5	N	N	Yandex Q
Chameleon	2,277	62,792	27.58	0.03	5	2581	0.48	Y	N	Wiki Pages
Tolokers	11,758	1,038,000	88.28	0.09	2	10	0.5	N	N	Toloka Platform
Amazon-ratings	24,492	186,100	7.60	0.14	5	556	0.5	N	N	Co-purchase network
genius	421,961	1,845,736	4.37	0.17	2	12	0.6	N	Y	Social Network
pokec	1,632,803	44,603,928	27.32	0.42	3	65	0.6	N	N	Social Network
arxiv-year	169,343	2,315,598	13.67	0.26	5	128	0.6	N	N	Citation
snap-patents	2,923,922	27,945,092	9.56	0.21	5	269	0.6	Y	Y	Citation
ogbn-proteins	132,534	79,122,504	597.00	0.05	94	8	0.2	N	N	Protein Network
Cora	19,793	126,842	6.41	0.56	70	8710	0.2	N	N	Citation Network
DBLP	17,716	105,734	5.97	0.68	4	1639	0.2	N	N	Citation Network
Computers	13,752	491,722	35.76	0.68	10	767	0.6	N	Y	Co-purchase Network
PubMed	19,717	88,648	4.50	0.69	3	500	0.2	N	N	Social Network
Cora_ML	2,995	16,316	5.45	0.75	7	2879	0.2	N	N	Citation Network
SmallCora	2,708	10,556	3.90	0.77	7	1433	0.05	N	N	Citation Network
CS	18,333	163,788	8.93	0.78	15	6805	0.2	N	N	Co-author Network
Photo	7,650	238,162	31.13	0.79	8	745	0.2	N	Y	Co-purchase Network
Physics	34,493	495,924	14.38	0.87	5	8415	0.2	N	N	Co-author Network
CiteSeer	4,230	10,674	2.52	0.94	6	602	0.2	N	N	Citation Network
wiki	11,701	431,726	36.90	0.58	10	300	0.99	Y	Y	Wikipedia
Reddit	232,965	114,615,892	491.99	0.74	41	602	0.66	N	N	Social Network

Table 15: Comparison with recent full-graph and graph sparsification baselines. We report F1-score with rank in parentheses. SGFormer is a full-graph baseline, GraphSAINT uses random-walk sampling, and UNIFEWS, SGNN-LS, and SGS-GNN are sparsification-based methods. The best result on each dataset is highlighted in bold.

Data.	SGFormer	GraphSAINT (RandomWalk)	UNIFEWS	SGNN-LS	SGS-GNN (20% edges)
Tolo.	**82.24±0.36 (1)**	78.16±0.00 (4)	79.68±0.23 (3)	70.50±0.00 (5)	79.98±0.17 (2)
john.	64.30±0.24 (4)	62.83±0.24 (5)	65.76±0.61 (3)	**75.27±0.00 (1)**	73.80±0.33 (2)
amhe.	59.53±1.39 (4)	56.66±2.67 (5)	67.04±0.12 (3)	68.83±0.00 (2)	**72.75±0.59 (1)**
Phot.	**94.98±0.02 (1)**	93.90±0.32 (4)	94.27±0.05 (2)	93.63±0.00 (5)	93.99±0.25 (3)
CS	**95.19±0.06 (1)**	93.47±0.01 (4)	92.17±0.12 (5)	93.89±0.00 (3)	94.25±0.15 (2)
Comp.	90.44±0.21 (3)	86.55±0.63 (5)	90.75±0.11 (2)	87.38±0.00 (4)	**90.97±0.31 (1)**
Avg. rank	2.33	4.50	3.00	3.33	**1.83**

Table 14: Comparison with heterophilic GNNs and sparsification baselines. We report F1-score with rank in parentheses. Red: not sparsifiers, Blue: sparsifiers. OOM: out of memory, (-): compilation error.

Data.	H2GCN	ASGC	CGP-GCN	UGT-GCN	DSpar-GCN	SGS-GCN
Tolo.	79.17±0.10 (2)	79.01±0.00 (3)	78.85±0.06 (4)	78.16±0.31 (6)	78.37±0.02 (5)	**79.98±0.17 (1)**
roma.	78.21±0.19 (1)	64.37±0.30 (3)	53.68±0.15 (5)	-	57.03±0.23 (4)	**64.69±0.12 (2)**
john.	63.00±0.33 (5)	75.77±0.03 (1)	69.43±0.30 (3)	66.99±1.01 (4)	61.37±1.23 (6)	**73.80±0.33 (2)**
amhe.	63.31±0.37 (4)	78.08±0.00 (1)	65.62±1.30 (3)	62.86±2.35 (5)	58.88±1.61 (6)	**72.75±0.59 (2)**
corn.	68.00±0.26 (2)	-	67.68±0.21 (3)	63.96±0.06 (4)	63.31±0.45 (5)	**69.15±0.33 (1)**
geni.	82.38±0.34 (3)	-	82.49±0.18 (2)	OOM	82.11±0.01 (4)	**82.59±0.00 (1)**
poke.	OOM	OOM	OOM	OOM	**60.92±0.01 (1)**	60.49±0.10 (2)
arxi.	40.74±0.10 (1)	-	**40.60±0.02 (2)**	OOM	37.58±0.09 (4)	38.42±0.10 (3)
Phot.	94.67±0.14 (1)	94.31±0.00 (2)	93.67±0.09 (4)	92.25±0.07 (5)	89.15±0.25 (6)	**93.99±0.25 (3)**
CS	94.70±0.06 (1)	94.10±0.00 (3)	-	93.76±0.01 (4)	91.60±0.02 (5)	**94.25±0.15 (2)**
Redd.	OOM	OOM	OOM	OOM	74.49±0.07 (2)	**91.45±0.06 (1)**
Avg. rank	2.22	2.17	3.25	4.67	4.36	**1.82**

(1) Section H.1 investigates $\mathcal{L}_{\text{assor}}$, $\mathcal{L}_{cons}$, EdgePE, GNN, and Conditional Updates mechanism. We also compare its runtime against standard SGS-GNN training vs SGS-GNN with conditional updates. We also show SGS-GNN can be used with other GNNs in Sec H.1.2.
(2) Section H.2 explores parameter settings with/without prior, different normalization and sampling methods, and inference with/without an ensemble of subgraphs.
(3) Section H.3 shows ideal settings for regularizer coefficients $\alpha_1, \alpha_2, \alpha_3$. We also show the impact of λ for augmenting the learned probability distribution p using p_{prior}.

H.1 Parameter's sensitivity

H.1.1 $\mathcal{L}_{\text{assor}}$, $\mathcal{L}_{cons}$, EdgePE, GNN, *and Conditional Updates.* Table 16 illustrates the performance of SGS-GNN with various combinations of regularizers, embedding layers in EdgePE, and convolutional layers in GNN.

(1) $\mathcal{L}_{\text{assor}}$: Case 1, 2 shows improvement in results when $\mathcal{L}_{\text{assor}}$ is used.
(2) $\mathcal{L}_{\text{cons}}$: From cases 6, 7, 8 shows L_{cons} improves results when GCN module is used in the GNN.
(3) EdgePE: In general, we found that the GCN layers for EdgePE encodings perform best (cases 7, 15).
(4) GNN: Both GCN and GAT modules yielded overall the best results (case 7, 15).
(5) Conditional updates: Case 3 shows that conditional updates can benefit some graphs.
We also investigated the runtime and quality of SGS-GNN with and without conditional updates for large-scale graphs. We found both have similar runtimes as the condition check expense gets compensated by fewer updates of EdgePE.

H.1.2 SGS-GNN *with other GNN modules.* The sampled sparse subgraphs from EdgePE can be fed into any downstream GNNs and demonstrate a couple of variants of SGS-GNN. Chebnet from Chebyshev [14], Graph Attention Network (GAT) [42], Graph Isomorphic Network (GIN) [46], Graph Convolutional Network (GCN) [17] are some of the GNNs used for demonstration.

Fig. 10 shows the performance of these GNNs on homophilic and heterophilic datasets. SGS-GCN and SGS-GAT are two best performing models.

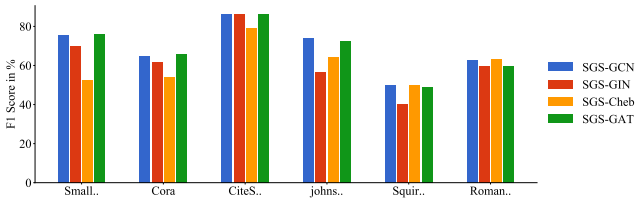


Figure 10: Performance of SGS-GNN with different GNN modules using 20% edges.

H.2 Impact of components

Impact of p_{prior}, Normalization & Sampling schemes, and Ensembling on SGS-GNN. Table 17 highlights the impact of the following components:

(1) Prior p_{prior}: Cases 2-3 show that augmenting the learned probability distribution $\tilde{p}$ with prior p_{prior} can benefit some datasets. We have also conducted an in-depth comparison between the distributions $\tilde{p}$ and augmented distribution $\tilde{p}_a$. Figure 11 shows that there $\tilde{p}_a$ is left skewed whereas $\tilde{p}$ is not. Since rare edges still get some negligible mass, it is possible for $\tilde{\mathcal{G}}$ constructed from $\tilde{p}_a$ to retain some bridge edges from these tails, if there are any.
(2) Normalization and Sampling: We considered three normalization and sampling techniques. i) sum-normalization with multinomial sampling, ii) softmax-normalization with temperature with multinomial sampling, and iii) Gumbel softmax normalization with Topk selection. Cases 3-5 show that each of these techniques can improve results in certain datasets, and thus, it is difficult to nominate a single one as best. However, in our experiments, we opted for multinomial sampling with softmax temperature annealing for training to encourage exploration in early iterations.
(3) Ensemble subgraphs during inference: Case 2 demonstrates that using multiple subgraphs for ensemble prediction yields better results than a single subgraph (Case 1).

H.3 Gridsearch for values of parameters.

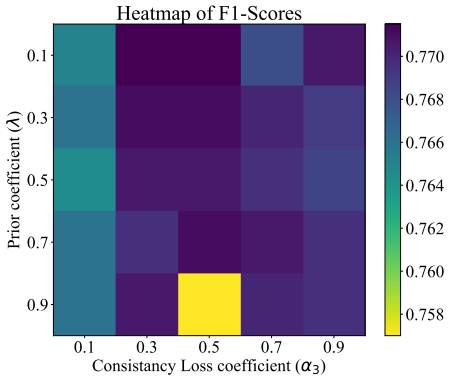


Figure 12: Grid search for the parameter λ for prior, and consistency loss, α_3 (Cora dataset).

Here we find the values for regularizer coefficient α_3 and Parameter λ. SGS-GNN computes the total loss at each epoch as $\mathcal{L} = \alpha_1\mathcal{L}_{CE} + \alpha_2\mathcal{L}_{\text{assor}} + \alpha_3\mathcal{L}_{\text{cons}}$, where $0 \leq \alpha_1, \alpha_2, \alpha_3 \leq 1$ are regularizer coefficients corresponding to the cross-entropy loss $\mathcal{L}_{CE}$, assortativity loss L_{assor} and consistency loss $\mathcal{L}_{\text{cons}}$ respectively.

Also recall that, when we use a prior probability distribution, the learned distribution values of $\tilde{p}$ are weighted through λ in $\tilde{p} = \lambda\tilde{p} + (1-\lambda)p_{\text{prior}}$. To avoid numerous combinations of values of three coefficients + the parameter λ, we have fixed $\alpha_1 = 1$, and $\alpha_2 = 1$. In the following, we investigate the performance of SGS-GNN with different values for α_3 and λ. Fig. 12 shows a grid search for different combinations of λ and α_3. As per our observation, the recommended values are $\lambda \in [0.3, 0.7]$, $\alpha_3 = 0.5$.

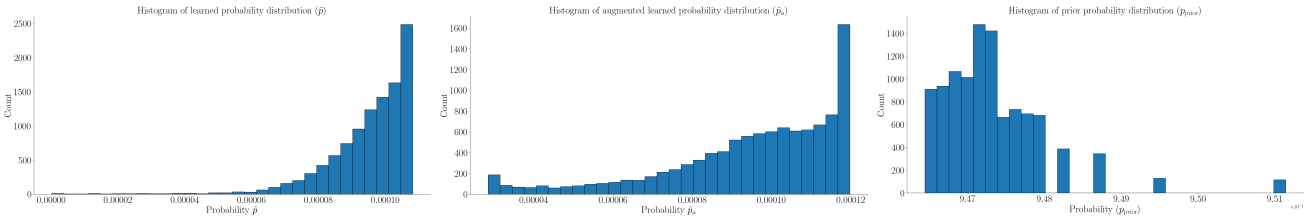
Table 16: Combination of EdgePE, GNN, Conditional update and L_{cons}.

Case	L_{assor}	L_{cons}	EdgePE	GNN	Cond.	SmallCora	CoraFull	johnshopkin
1	N	N	MLP	GCN	N	73.80 ± 0.67	61.78 ± 0.20	66.12 ± 1.38
2	Y	N	MLP	GCN	N	74.88 ± 0.15	63.99 ± 0.24	66.18 ± 1.05
3	Y	N	MLP	GCN	Y	75.82 ± 0.46	64.07 ± 0.31	66.87 ± 0.93
4	Y	Y	MLP	GCN	Y	76.58 ± 0.47	65.33 ± 0.28	69.25 ± 0.76
5	Y	Y	MLP	GCN	N	73.90 ± 0.81	64.74 ± 0.21	69.21 ± 0.60
6	Y	N	GCN	GCN	Y	75.80 ± 0.77	65.66 ± 0.14	71.06 ± 0.32
7	Y	Y	GCN	GCN	Y	77.50 ± 0.62	**66.56 ± 0.22**	70.79 ± 0.18
8	Y	Y	GCN	GCN	N	75.88 ± 0.71	65.87 ± 0.17	69.83 ± 0.67
9	Y	N	GSAGE	GCN	Y	75.82 ± 0.44	63.70 ± 0.09	67.53 ± 0.80
10	Y	Y	GSAGE	GCN	Y	77.48 ± 0.61	65.12 ± 0.11	68.63 ± 0.66
11	Y	Y	GSAGE	GCN	N	75.44 ± 1.40	64.70 ± 0.21	68.31 ± 0.50
12	Y	N	MLP	GAT	Y	77.72 ± 1.63	66.40 ± 0.08	67.92 ± 0.73
13	Y	Y	MLP	GAT	Y	75.78 ± 3.22	66.46 ± 0.16	68.17 ± 0.33
14	Y	Y	MLP	GAT	N	74.17 ± 2.47	65.89 ± 0.31	67.82 ± 0.40
15	Y	N	GCN	GAT	Y	**78.18 ± 0.74**	66.33 ± 0.20	**71.97 ± 0.59**
16	Y	Y	GCN	GAT	Y	76.94 ± 2.76	66.39 ± 0.18	71.00 ± 0.96
17	Y	Y	GCN	GAT	N	75.57 ± 1.55	65.70 ± 0.38	71.62 ± 0.86
18	Y	N	GSAGE	GAT	Y	77.98 ± 0.79	66.38 ± 0.23	69.29 ± 1.56
19	Y	Y	GSAGE	GAT	Y	75.74 ± 2.02	66.41 ± 0.25	68.82 ± 0.24
20	Y	Y	GSAGE	GAT	N	74.33 ± 2.01	65.22 ± 0.21	66.31 ± 0.52

Table 17: Ablation Studies different components of SGS-GNN.

Case	Prior	Norm.	Sampl.	Ensem.	SmallCora	Cora_ML	CiteSeer	Squirrel	johnshopkins55	Roman-empire
1	N	Sum	Mult	N	69.30 ± 1.20	81.05 ± 0.74	82.84 ± 0.47	48.90 ± 1.06	63.86 ± 0.58	63.27 ± 0.31
2	N	Sum	Mult	Y	72.84 ± 0.91	82.92 ± 0.73	**87.42 ± 0.42**	46.30 ± 1.18	65.14 ± 1.14	**64.31 ± 0.13**
3	Y	Sum	Mult	Y	75.54 ± 0.41	**83.87 ± 0.69**	86.31 ± 0.26	47.97 ± 0.60	72.68 ± 0.51	62.88 ± 0.19
4	Y	Softmax	Mult	Y	75.44 ± 0.51	83.81 ± 0.72	86.31 ± 0.26	47.90 ± 0.42	**72.97 ± 0.20**	62.98 ± 0.16
5	Y	Gumbel	TopK	Y	**76.24 ± 0.43**	83.36 ± 0.34	86.44 ± 0.16	**51.49 ± 0.72**	71.83 ± 1.00	63.00 ± 0.11

Prior: Use of prior, **Sum:** Sum-Normalization, **Softmax:** *Softmax* with temperature annealing
Mult: *Multinomial* Sampling, **Gumbel:** *Gumbel-Softmax* with *TopK*



(a) (b) (c)

Figure 11: The learned probability distribution $\tilde{p}$ (top-left), augmented distribution $\tilde{p}_a$ (top-right) and fixed prior p_{prior} (bottom). Augmentation puts negligible mass on some rare yet critical edges in the left tail of $\tilde{p}_a$.