PRESERVING LEARNED PROPAGATION OPERATORS FOR GRAPH SPARSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph sparsification has traditionally focused on preserving the graph Laplacian through spectral approximation. However, many modern GNNs propagate information through model-dependent or learned operators that can differ substantially from the input graph Laplacian. Consequently, preserving the input graph alone may be insufficient to preserve the behavior of a trained model. We propose an operator-aware perspective on graph sparsification that instead preserves the learned propagation operator. We instantiate this perspective for Neural Sheaf Diffusion (NSD) through a no-retraining framework for constructing sparse sheaf operators while preserving learned propagation behavior. We provide theoretical analysis connecting operator approximation to stable representations and predictions, and experiments on four benchmark datasets demonstrate consistently higher sparse-model fidelity than conventional reweighting. These results suggest operator-action preservation as an effective post-hoc reweighting principle for no-retraining sparsification of trained NSD models.

1 INTRODUCTION

Graph neural networks (GNNs) (Bruna et al., 2013; Defferrard et al., 2016; Gilmer et al., 2017) rely on repeated message passing over graph edges to propagate information between nodes. While this propagation is responsible for the expressive power of GNNs, it also becomes a major computational bottleneck on large or dense graphs (Zeng et al., 2020; Chen et al., 2018). Moreover, redundant or noisy edges may unnecessarily increase computation and even degrade the quality of learned representations (Rong et al., 2020). Graph sparsification seeks to alleviate these issues by replacing the original graph with a smaller weighted graph that preserves the information needed for downstream tasks (Spielman & Srivastava, 2011).

A classical line of graph sparsification focuses on preserving the graph Laplacian through spectral approximation (Spielman & Srivastava, 2011). Effective-resistance sampling is a representative example, where edges are sampled according to their importance in the graph Laplacian and then reweighted to preserve its spectral properties. Such methods provide strong theoretical guarantees for preserving the geometry induced by the input graph and have become a foundation of modern graph sparsification.

However, the graph Laplacian is often not the operator that actually governs the forward dynamics of a trained GNN. Modern GNNs propagate information using model-dependent operators, such as learned edge weights, attention coefficients, diffusion operators, or feature-dependent transformations (Veličković et al., 2017; Gasteiger et al., 2018). Consequently, preserving the input graph Laplacian alone does not necessarily preserve the behavior of the trained model (Krishnagopal, 2026). This observation suggests a different principle for sparsification: rather than preserving only the input graph, sparse graphs should preserve the propagation operator learned by the model itself.

We argue that this mismatch is not merely a limitation of existing graph sparsification methods, but a consequence of viewing sparsification from the wrong object. This mismatch highlights a limitation of classical spectral sparsification, whose preservation target is the geometry of the input graph. In contrast, we study a complementary post-hoc setting in which a trained model is fixed and sparsification is formulated as preserving the action of its learned propagation operator on task-relevant representations.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We investigate this idea in the setting of Neural Sheaf Diffusion (NSD) (Bodnar et al., 2022). Unlike conventional GNNs, NSD explicitly learns a sheaf Laplacian through learned restriction maps, and this learned operator directly governs the diffusion process used during inference. As a result, NSD provides an ideal testbed for studying operator-aware sparsification, since the propagation operator is explicitly learned rather than implicitly determined by the input graph.

Based on this perspective, we propose an operator-aware no-retraining sparsification framework that preserves the action of the learned sheaf operator. Our key component, Probe Least Squares (**Probe-LS**), calibrates the weights of retained edges by solving a representation-aware least-squares problem on probe representations extracted from the trained dense model. Unlike conventional reweighting methods, Probe-LS does not attempt to reconstruct the input graph Laplacian. Instead, it directly preserves the learned propagation behavior of the trained model, making it particularly suitable for no-retraining sparsification.

We further provide a stability-based justification showing that preserving the learned operator action controls representation drift and logit drift after sparsification. Experiments on four node-classification benchmarks demonstrate that Probe-LS consistently improves same-support sparsification over effective resistance reweighting (ER reweighting) while substantially reducing learned-operator drift and logit drift. These results suggest that preserving learned operator action is an effective principle for sparsifying NSD models without retraining.

Our contributions are summarized as follows:

- We formulate post-hoc sparsification of a trained NSD model as preserving learned operator action on task-relevant representations.
- We propose Probe-LS, a support-agnostic nonnegative least-squares reweighting method that calibrates any retained edge support to match the dense learned operator on probe representations.
- We provide a stability analysis connecting probe-subspace operator-action error to representation and logit stability, and study Sheaf-ER as one compatible support-selection strategy.

2 Related Work

Spectral graph sparsification. Spectral graph sparsification constructs a sparse weighted graph whose Laplacian approximately preserves the quadratic form of the original graph. Classical effective-resistance sparsification samples edges according to leverage scores and provides spectral approximation guarantees for the input graph Laplacian (Spielman & Srivastava, 2011). More recent work has also used spectral graph pruning to improve information flow in GNNs (Jamadandi et al., 2024). These methods primarily define their preservation target in terms of the input graph Laplacian or its spectral geometry. Our setting instead starts from a trained model and asks how sparsification can preserve the action of the propagation operator used by that model.

Model- and task-aware graph sparsification for GNNs. Beyond spectral sparsification, several methods incorporate model or task information when selecting graph structure. DropEdge removes edges during training as regularization (Rong et al., 2020), NeuralSparse learns task-dependent edge selection (Zheng et al., 2020), and recent methods combine semantic and topological criteria or adapt sparsity locally (Zhang et al., 2024; 2025). WIS preserves a GNN's capacity to model vertex interactions (Razin et al., 2023), while Graph Lottery Ticket methods seek sparse graph/model subnetworks that retain predictive performance (Yue et al., 2025). In contrast, we consider a post-hoc setting in which the trained model is frozen and only retained edge weights are calibrated to preserve learned operator action.

Neural sheaf models. Neural sheaf models generalize graph diffusion through sheaf Laplacians (Bodnar et al., 2022; Barbero et al., 2022; Hansen & Gebhart, 2020). Existing methods focus on constructing expressive sheaf propagation operators, either by learning restriction maps (Bodnar et al., 2022) or by using connection Laplacians (Barbero et al., 2022). In contrast, we study how to sparsify a trained sheaf diffusion model while preserving its learned propagation operator.

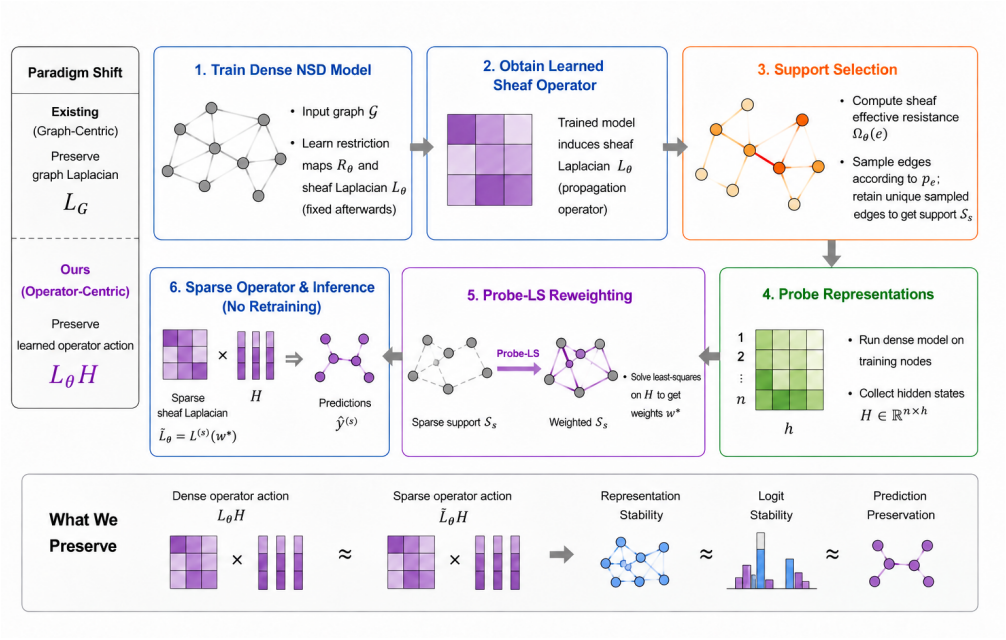


Figure 1: Overview of the proposed operator-aware no-retraining sparsification framework for NSD.

Functional fidelity and pruning. Graph explanation methods often evaluate sparse structures by their fidelity to the original prediction (Schlichtkrull et al., 2021; Ying et al., 2019). This is closely related to knowledge distillation (Hinton et al., 2015), where compressed models match the outputs of a reference model. Our use of logit drift and prediction consistency follows the same functional-fidelity perspective.

Position of this work. Prior graph sparsification methods preserve different objects, including spectral graph geometry, task performance, interaction capacity, or learned representation geometry. We study a distinct post-hoc preservation target: the action of the propagation operator already learned by a trained model.

More specifically, given a fixed sparse support S, Probe-LS calibrates nonnegative edge weights to match the dense operator action on probe representations extracted from the frozen model. Thus, our contribution is a probe-subspace post-selection reweighting objective rather than a claim that operator-aware support selection is universally preferable.

3 Background

Cellular Sheaves on Graphs A cellular sheaf $\mathcal{F}$ on an undirected graph $G = (V, E)$ assigns vector spaces $\mathcal{F}(v)$ and $\mathcal{F}(e)$ to nodes and edges, together with linear restriction maps

$$\mathcal{F}_{v \trianglelefteq e} : \mathcal{F}(v) \to \mathcal{F}(e)$$

for every incident node–edge pair (Hansen & Ghrist, 2019). For a node signal $x = (x_v)_{v\in V}$, fix an arbitrary orientation $e = (u, v)$ and define the sheaf coboundary operator edgewise as

$$(\delta_{\mathcal{F}} x)_e = \mathcal{F}_{v \trianglelefteq e} x_v - \mathcal{F}_{u \trianglelefteq e} x_u.$$

The sheaf Laplacian is defined as

$$L_{\mathcal{F}} = \delta_{\mathcal{F}}^\top \delta_{\mathcal{F}},$$

with quadratic form

$$x^\top L_{\mathcal{F}} x = \sum_{e=(u,v)\in E} \|\mathcal{F}_{v \trianglelefteq e} x_v - \mathcal{F}_{u \trianglelefteq e} x_u\|_2^2 .$$

If B_e denotes the edge block of $\delta_{\mathcal{F}}$, the edge-wise decomposition

$$L_{\mathcal{F}} = \sum_{e \in E} B_e^\top B_e$$

plays a central role in our method, as it enables sparsification through selecting and reweighting individual edge operators.

Effective-Resistance Graph Sparsification Classical spectral graph sparsification aims to construct a sparse weighted graph $\widetilde{G}$ that approximates the original graph G in terms of its Laplacian quadratic form. Let L_G be the graph Laplacian of G. A sparse graph $\widetilde{G}$ with Laplacian $L_{\widetilde{G}}$ is called a spectral sparsifier if, for all vectors x,

$$(1-\varepsilon)x^\top L_G x \leq x^\top L_{\widetilde{G}} x \leq (1+\varepsilon)x^\top L_G x.$$

This condition means that the sparse graph preserves the energy of all graph signals up to a multiplicative error.

A standard way to construct such sparsifiers is effective-resistance sampling (Spielman & Srivastava, 2011). For an edge $e = (u, v)$, its effective resistance is

$$R_e = (\mathbf{e}_u - \mathbf{e}_v)^\top L_G^\dagger (\mathbf{e}_u - \mathbf{e}_v),$$

where $L_G^\dagger$ is the Moore–Penrose pseudoinverse of the graph Laplacian. Intuitively, R_e measures how important the edge is for preserving global connectivity and diffusion geometry. Edges with high effective resistance tend to be structurally important and are sampled with higher probability.

Typically, each edge is sampled with probability proportional to its leverage score

$$\tau_e = w_e R_e,$$

where w_e is the edge weight. When an edge is sampled, its weight is rescaled by the inverse of its sampling probability so that the resulting sparse Laplacian remains an unbiased approximation of the original Laplacian. Under an appropriate sampling budget, this procedure yields a sparse graph whose Laplacian approximately preserves the quadratic form of the original graph. Effective-resistance sparsification therefore preserves the geometry induced by the graph Laplacian. In NSD, however, message passing is governed by a learned sheaf Laplacian rather than the graph Laplacian. This difference motivates the operator-aware formulation presented in the next section.

4 Method: Operator-Preserving Sparsification for Graph Diffusion Models

Unified Operator View Many graph neural networks propagate information through a graph-dependent linear operator. In classical graph neural networks such as GCN, this operator is the graph Laplacian or a normalized adjacency matrix, whereas in NSD it is the learned sheaf Laplacian induced by learned restriction maps. Instead of preserving only the input graph, we view sparsification as preserving the action of the operator that actually governs information propagation in the trained model. Let $L_{\mathcal{M}}$ denote the propagation operator associated with a trained model $\mathcal{M}$. Our objective is to construct a sparse operator $\widetilde{L}_{\mathcal{M}}$ whose action closely approximates that of $L_{\mathcal{M}}$. The operator-centric viewpoint is conceptually applicable beyond NSD when the propagation operator admits a suitable decomposition. In this work, we focus on NSD, where the learned sheaf Laplacian provides an explicit edge-wise PSD decomposition. Classical GCNs can be interpreted as using a trivial sheaf with identity restriction maps (Bodnar et al., 2022), in which case the propagation operator reduces to the standard graph Laplacian. Therefore, preserving the graph Laplacian is a special case of our operator-preserving viewpoint.

No-retraining setting. We first train the dense NSD model and then freeze all learned model parameters, including the restriction maps, feature transformations, and classifier. Sparsification only optimizes new nonnegative coefficients for the retained edge operators; no original model parameter is updated.

Learned-Sheaf Operator for NSD NSD propagates node features using a learned sheaf Laplacian induced by restriction maps. Unlike the graph Laplacian, this operator is learned during training and therefore governs the actual message-passing dynamics of the trained model.

A key observation is that the learned sheaf Laplacian admits an edge-wise positive semidefinite decomposition,

$$L_{\mathcal{F}} = \sum_{e \in E} L_e, \qquad L_e = B_e^\top B_e,$$

where B_e is a matrix representation of coboundary operator for each edge, and each edge contributes an independent PSD operator determined by its restriction maps.

This decomposition turns operator-preserving sparsification into a sparse operator approximation problem.

Specifically, we seek

$$\widetilde{L} = \sum_{e \in S} w_e L_e,$$

whose action approximates the learned dense operator. This naturally raises two questions:

(i) Which edge operators should be retained?

(ii) How should the retained operators be weighted?

We address these questions through sheaf effective-resistance support selection and Probe-LS reweighting, respectively.

Sheaf Effective-Resistance Support Sampling and Reweighting Building on the two-stage formulation above (support selection and weight reconstruction), we implement support sampling via sheaf effective-resistance scores and then apply reweighting on the sampled support. Concretely, we compute a sheaf effective-resistance score for every edge using the learned sheaf Laplacian, sample edges according to these scores, and compare different reweighting rules on the sampled support.

Let

$$L_{\mathcal{F}_\theta} = \sum_{e \in E} L_e, \qquad L_e = B_e^\top B_e$$

be the edge-wise decomposition of the learned sheaf Laplacian.

Motivated by effective-resistance leverage scores for graph Laplacians (Spielman & Srivastava, 2011), we define a sheaf analogue using the learned sheaf Laplacian and the edge coboundary block B_e from the sheaf Laplacian construction (Hansen & Ghrist, 2019):

$$s_e = \operatorname{tr}\left(B_e (L_{\mathcal{F}_\theta} + \lambda I)^{-1} B_e^\top\right).$$

When multiple sheaf layers are present, we normalize the layer-wise scores and take their mean to obtain a single score per edge. The aggregated scores are then converted to sampling probabilities

$$p_e = \frac{s_e}{\sum_{e' \in E} s_{e'}}.$$

As in classical leverage-score sparsification (Spielman & Srivastava, 2011), given a sampling budget q, we sample edges independently with replacement according to p_e. Let c_e denote the number of times edge e is sampled. The sparse support is the set of unique sampled edges

$$S = \{e \in E : c_e > 0\}.$$

The achieved unique keep ratio is random and may differ slightly from the target. In our implementation, we choose the sampling budget to match a target unique-edge ratio as closely as possible. Given a retained sparse support S, Probe-LS calibrates the coefficients of the retained edge operators so that the resulting sparse sheaf operator preserves the action of the learned dense sheaf operator on task-relevant representations. Let $L_{\mathcal{F}_\theta}$ denote the learned dense sheaf operator and let L_e be the contribution of edge e. Given a representation or probe matrix X, Probe-LS computes nonnegative edge weights by solving the following representation-aware least-squares problem:

$$\alpha^\star = \arg\min_{\alpha_e \geq 0} \left\| \left(L_{\mathcal{F}_\theta} - \sum_{e \in S} \alpha_e L_e \right) X \right\|_F^2.$$

This objective directly matches the dense sheaf diffusion action on the directions encountered by the trained model. The resulting sparse operator is

$$\widetilde{L}_{\mathcal{F}_\theta}^{\text{probe}} = \sum_{e \in S} \alpha_e^\star L_e.$$

Unlike inverse-probability ER reweighting, which is tied to the sampling distribution, Probe-LS is a post-selection calibration method and can be applied to any retained support.

From Operator Approximation to Prediction Preservation. Rather than approximating the learned operator uniformly over the ambient space, Probe-LS preserves its action on representative probe features. For a fixed support S, it is, by construction, the nonnegative edge-weight assignment that minimizes the operator-action error on the probe matrix X. We next connect this probe-subspace approximation to energy preservation, representation and logit stability, and prediction preservation under a margin condition.

Energy Preservation on the Probe Subspace We next relate operator-action preservation to preservation of the sheaf energy.

Proposition 1 (Probe-Subspace Energy Preservation)**.** *Let L and $\widetilde{L}$ be symmetric positive semidefinite operators. Suppose*

$$\|(L - \widetilde{L})X\|_F \leq \varepsilon.$$

Then for any vector $x = Xa$ in the span of the probe representations,

$$|x^\top L x - x^\top \widetilde{L} x| \leq \varepsilon \|Xa\|_2 \|a\|_2.$$

In particular, if $\|Xa\|_2 \leq B$ and $\|a\|_2 \leq A$, then

$$|x^\top L x - x^\top \widetilde{L} x| \leq AB\varepsilon.$$

This result clarifies the role of Probe-LS: it preserves the learned sheaf energy not globally, but on the representation subspace probed from the trained dense model. This is precisely the regime relevant to no-retraining sparsification, where the sparse model is expected to follow the dense model's forward trajectory.

One-Layer Representation Stability. Having established operator approximation on the probe subspace, we next examine how operator-action error affects the hidden representations. Consider a diffusion layer

$$H^+ = \Phi(L, H),$$

where L is the learned dense propagation operator and H is the input representation. The corresponding sparse layer is

$$\widetilde{H}^+ = \Phi(\widetilde{L}, \widetilde{H}).$$

Theorem 1 (One-Layer Representation Stability)**.** *Assume that the layer map Φ satisfies the following two stability conditions. First, for a fixed operator $\widetilde{L}$, it is Lipschitz with respect to its representation input:*

$$\|\Phi(\widetilde{L}, H) - \Phi(\widetilde{L}, \widetilde{H})\|_F \leq C_H \|H - \widetilde{H}\|_F.$$

Second, for a fixed representation H, it is Lipschitz with respect to the operator action:

$$\|\Phi(L, H) - \Phi(\widetilde{L}, H)\|_F \leq C_L \|(L - \widetilde{L})H\|_F.$$

Then the one-layer representation drift satisfies

$$\|H^+ - \widetilde{H}^+\|_F \leq C_H \|H - \widetilde{H}\|_F + C_L \|(L - \widetilde{L})H\|_F.$$

The bound separates inherited representation drift from the additional perturbation caused by sparsifying the propagation operator. In particular, for a shared input H, the representation error is directly controlled by $\|(L - \widetilde{L})H\|_F$, the quantity targeted by Probe-LS.

Multi-Layer Representation Stability. The previous theorem separates the representation drift at each layer into the drift inherited from the previous layer and the additional perturbation caused by replacing the dense propagation operator with its sparse approximation. Applying this bound recursively yields a multi-layer stability result.

Corollary 1 (Multi-Layer Representation Stability). *Let*

$$\Delta_k = \|H^{(k)} - \widetilde{H}^{(k)}\|_F, \qquad \epsilon_k = \|(L^{(k)} - \widetilde{L}^{(k)})H^{(k)}\|_F.$$

Under the assumptions of Theorem 1,

$$\Delta_{k+1} \le C_H^{(k)}\Delta_k + C_L^{(k)}\epsilon_k.$$

If $\Delta_0 = 0$*, then*

$$\Delta_K \le \sum_{k=0}^{K-1} C_L^{(k)} \left(\prod_{j=k+1}^{K-1} C_H^{(j)} \right) \epsilon_k.$$

Thus, for a fixed-depth model with bounded layer-wise stability constants, controlling operator-action error along the dense trajectory controls the final representation drift. The proof is given in Appendix A.

Corollary 2 (Logit Stability). *If the classifier head* g *is* C_g*-Lipschitz, with* $z = g(H^{(K)})$ *and* $\widetilde{z} = g(\widetilde{H}^{(K)})$*, then*

$$\|z - \widetilde{z}\|_F \le C_g \|H^{(K)} - \widetilde{H}^{(K)}\|_F.$$

Prediction Preservation Under a Margin Condition. For node i, let

$$c_i = \arg\max_c z_{i,c}, \qquad m_i = z_{i,c_i} - \max_{c \neq c_i} z_{i,c}$$

denote the dense-model prediction and its logit margin. If

$$\|z_i - \widetilde{z}_i\|_\infty < \frac{m_i}{2},$$

then

$$\arg\max_c z_{i,c} = \arg\max_c \widetilde{z}_{i,c}.$$

Thus, sufficiently small per-node logit perturbation preserves the dense-model prediction whenever it is smaller than half of the corresponding prediction margin.

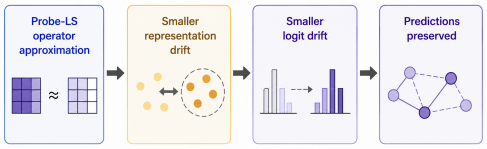


Figure 2: Stability chain motivating Probe-LS. Operator-action approximation controls representation and logit perturbations under the stated Lipschitz assumptions; prediction preservation additionally requires a per-node margin condition.

5 Experiments

We evaluate whether a sparse graph can replace the dense graph in a trained NSD model without retraining. Our goal is not to train a new sparse model, but to measure how well sparsification preserves the behavior of the already trained dense model. This no-retraining setting directly tests whether the sparse operator preserves the learned diffusion geometry used by the model.

Datasets We evaluate on four node classification benchmarks: Chameleon, Squirrel, PubMed, and Roman-empire. Chameleon and Squirrel are heterophilic webpage graphs (Pei et al., 2020), PubMed is a homophilic citation network (Sen et al., 2008), and Roman-empire is a challenging heterophilic benchmark with substantially different graph topology (Platonov et al., 2023). These datasets allow us to evaluate operator-aware sparsification under diverse graph structures and homophily regimes.

Dense model We first train a dense NSD model on each dataset and keep all learned parameters fixed thereafter. Sparsification is applied only at inference time, so any performance difference reflects the fidelity of the sparse operator rather than additional optimization.

Probe Representations Unless otherwise stated, the probe matrix consists of the final hidden representations on the training nodes before the classifier, capturing the task-relevant feature subspace without using validation or test information.

Table 1: No-retraining sparsification accuracy across four datasets. Dense denotes the original trained NSD model. Graph_ER + ER reweighting uses graph effective-resistance support with sampling-based reweighting. Random_rw uses random reweighting. Sheaf_ER + Probe-LS uses sheaf-based support with representation-aware least-squares reweighting. Higher accuracy is better.

Dataset	Dense	Ratio	Graph_ER + ER reweighting	Random_rw	Sheaf_ER + Probe-LS
Chameleon	0.6634 ± 0.0196	0.25	0.5772 ± 0.0248	0.4859 ± 0.0290	$\mathbf{0.6194 \pm 0.0213}$
		0.35	0.6076 ± 0.0207	0.5396 ± 0.0254	$\mathbf{0.6423 \pm 0.0195}$
		0.45	0.6218 ± 0.0208	0.5728 ± 0.0256	$\mathbf{0.6503 \pm 0.0185}$
		0.55	0.6329 ± 0.0197	0.6057 ± 0.0211	$\mathbf{0.6554 \pm 0.0179}$
		0.65	0.6425 ± 0.0195	0.6193 ± 0.0222	$\mathbf{0.6574 \pm 0.0197}$
Squirrel	0.5726 ± 0.0323	0.25	0.5226 ± 0.0250	0.3016 ± 0.0244	$\mathbf{0.5658 \pm 0.0315}$
		0.35	0.5404 ± 0.0239	0.3608 ± 0.0374	$\mathbf{0.5716 \pm 0.0328}$
		0.45	0.5510 ± 0.0283	0.4077 ± 0.0282	$\mathbf{0.5715 \pm 0.0331}$
		0.55	0.5600 ± 0.0245	0.4488 ± 0.0258	$\mathbf{0.5712 \pm 0.0323}$
		0.65	0.5651 ± 0.0251	0.4955 ± 0.0250	$\mathbf{0.5718 \pm 0.0320}$
PubMed	0.8884 ± 0.0050	0.25	0.8360 ± 0.0046	0.8289 ± 0.0070	$\mathbf{0.8463 \pm 0.0052}$
		0.35	0.8491 ± 0.0036	0.8420 ± 0.0046	$\mathbf{0.8593 \pm 0.0041}$
		0.45	0.8637 ± 0.0053	0.8542 ± 0.0048	$\mathbf{0.8688 \pm 0.0047}$
		0.55	0.8721 ± 0.0047	0.8638 ± 0.0061	$\mathbf{0.8773 \pm 0.0048}$
		0.65	0.8791 ± 0.0049	0.8714 ± 0.0051	$\mathbf{0.8816 \pm 0.0060}$
Roman-empire	0.8546 ± 0.0054	0.25	0.4083 ± 0.0176	0.4085 ± 0.0174	$\mathbf{0.5076 \pm 0.0129}$
		0.35	0.4631 ± 0.0191	0.4651 ± 0.0190	$\mathbf{0.5691 \pm 0.0115}$
		0.45	0.5068 ± 0.0157	0.5100 ± 0.0161	$\mathbf{0.6250 \pm 0.0089}$
		0.55	0.5568 ± 0.0174	0.5584 ± 0.0143	$\mathbf{0.6754 \pm 0.0086}$
		0.65	0.6009 ± 0.0142	0.5968 ± 0.0130	$\mathbf{0.7240 \pm 0.0093}$

Sparsification methods. We compare three main sparse variants. First, **Graph_ER + ER Reweighting** selects edges using effective-resistance scores from the standard graph Laplacian and applies the classical inverse-probability reweighting used in spectral graph sparsification. Second, **Random_rw** uses the same graph-based support family but assigns random edge weights. This method is not intended as a competitive baseline; rather, it serves as a sanity check illustrating the importance of weight calibration. Third, **Sheaf_ER + Probe-LS** selects edges using sheaf effective-resistance scores and calibrates the retained edge weights through the proposed representation-aware least-squares optimization.

Metrics. Following our theoretical analysis, we evaluate operator preservation, functional fidelity, and downstream performance using relative sheaf operator drift, logit drift, prediction consistency, and test accuracy. Relative sheaf operator drift is defined as $\|\tilde{L} - L\|_F / \|L\|_F$, where L and $\tilde{L}$ denote the dense and sparse sheaf propagation operators, respectively. Logit drift is the average ℓ_2 distance between dense and sparse logits. Prediction consistency is the fraction of nodes for which the sparse model predicts the same label as the dense model. Lower values of operator and logit drift and higher values of prediction consistency and test accuracy indicate better performance.

5.1 Main Results

Table 1 summarizes the no-retraining sparsification results across all datasets and sparsity ratios.

Across all datasets, Sheaf_ER + Probe-LS consistently achieves the highest sparse accuracy and substantially reduces the gap to the dense model, whereas Random_rw performs markedly worse. The advantage is particularly pronounced at more aggressive sparsification levels, where preserving the learned operator becomes more challenging.

5.2 Ablation Study

Effect of Probe-LS reweighting. To isolate the contribution of Probe-LS from that of support selection, we first fix the sparse support and compare different reweighting strategies. Table 2 shows results for representative keep ratios. Probe-LS consistently improves both graph-based and sheaf-based supports, indicating that the observed gains arise from operator-aware reweighting rather than support selection alone.

Table 2: Same-support reweighting ablation. G/S denote graph/sheaf support; Probe denotes Probe-LS.

Dataset	Ratio	G-ER	G-Probe	S-ER	S-Probe
Cham.	.35	.6076	**.6319**	.6176	**.6423**
Cham.	.55	.6329	**.6551**	.6413	**.6554**
Squi.	.35	.5404	**.5599**	.5595	**.5716**
Squi.	.55	.5600	**.5692**	.5679	**.5712**
Pub.	.35	.8491	**.8593**	.8518	**.8593**
Pub.	.55	.8721	**.8770**	.8714	**.8773**

Table 3: Average Probe-LS accuracy gain over natural reweighting.

Dataset	G-ER	S-ER
Cham.	+.0216	+.0191
Squi.	+.0147	+.0094
Pub.	+.0065	+.0069

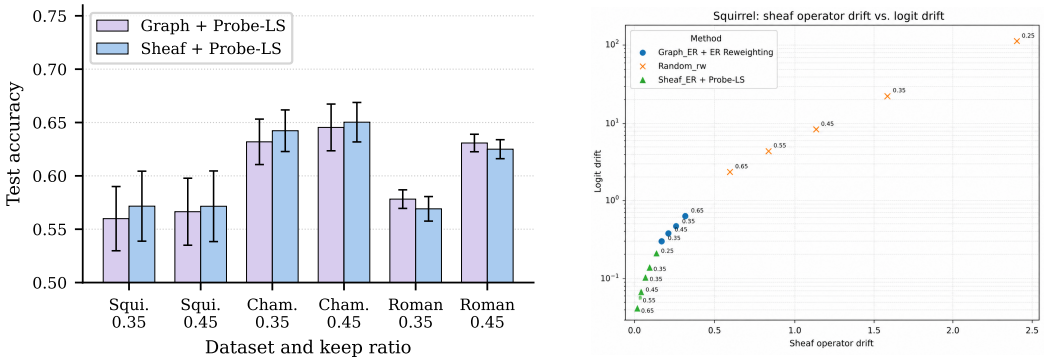


Figure 3: Sparsification analysis. Left: Support-selection ablation with Probe-LS fixed. Right: Relationship between sheaf operator drift and logit drift on Squirrel.

Effect of Support Selection. Left panel of Figure 3 compares graph and sheaf effective-resistance supports with Probe-LS fixed. Sheaf support yields higher mean accuracy on Squirrel and Chameleon, whereas graph support performs better on Roman-empire. Thus, unlike the consistent benefit of Probe-LS reweighting, support selection remains dataset dependent.

5.3 Operator Drift Explains Functional Drift

Right panel of Figure 3 shows that lower sheaf operator drift is associated with lower logit drift. Probe-LS moves sparse models toward the lower-left region, supporting our theoretical motivation that preserving the learned operator improves functional fidelity. The same qualitative trend holds on the remaining datasets (Appendix).

6 Conclusion

We introduced Probe-LS, a post-hoc reweighting method that preserves the action of a trained NSD propagation operator on task-relevant probe representations. Across multiple sparse supports, Probe-LS consistently improves no-retraining fidelity, while our analysis connects probe-subspace operator-action error to representation and prediction stability. These results suggest operator-action matching as a useful post-selection calibration principle for trained graph diffusion models. Future work includes cheaper calibration and extensions beyond NSD.

AI use statement

Generative AI tools were used to assist with language editing, the layout and presentation of figures and tables, suggestions on the organization and structure of the manuscript, and code debugging. They were not used to generate the core research ideas, theoretical results, methodology, experimental design, or interpretation of the results. All AI-assisted outputs were reviewed and verified by the authors. The authors take full responsibility for the final content of the manuscript.

Reproducibility statement

Implementation details, hyperparameters, computing environments, and random seeds are provided in the appendix. We report repeated sparsification runs and statistical significance tests, and provide complete proofs of the theoretical results in Appendix A. Code and additional materials are included in the supplementary material.

References

Federico Barbero, Cristian Bodnar, Haitz Sáez de Ocáriz Borde, Michael Bronstein, Petar Veličković, and Pietro Liò. Sheaf neural networks with connection laplacians. In *Topological, Algebraic and Geometric Learning Workshops 2022*, pp. 28–36. PMLR, 2022.

Cristian Bodnar, Francesco Di Giovanni, Benjamin Chamberlain, Pietro Lio, and Michael Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. *Advances in Neural Information Processing Systems*, 35:18527–18541, 2022.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

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

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

Jakob Hansen and Thomas Gebhart. Sheaf neural networks. *CoRR*, abs/2012.06333, 2020. URL `https://arxiv.org/abs/2012.06333`.

Jakob Hansen and Robert Ghrist. Toward a spectral theory of cellular sheaves. *Journal of Applied and Computational Topology*, 3(4):315–358, 2019.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Adarsh Jamadandi, Celia Rubio-Madrigal, and Rebekka Burkholz. Spectral graph pruning against over-squashing and over-smoothing. *Advances in Neural Information Processing Systems*, 37: 10348–10379, 2024.

Sanjukta Krishnagopal. Spectral graph sparsification preserves representation geometry in graph neural networks. *arXiv preprint arXiv:2605.01136*, 2026.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-GCN: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *International Conference on Learning Representations*, 2023.

Noam Razin, Tom Verbin, and Nadav Cohen. On the ability of graph neural networks to model interactions between vertices. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020.

Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting graph neural networks for {nlp} with differentiable edge masking. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=WznmQa42ZAx.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/d80b7040b773199015de6d3b4293c8ff-Paper.pdf.

Yanwei Yue, Guibin Zhang, Haoran Yang, and Dawei Cheng. Fast track to winning tickets: Repowering one-shot pruning for graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 13152–13160, 2025. doi: 10.1609/aaai.v39i12.33435.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method, 2020. URL https://arxiv.org/abs/1907.04931.

Guibin Zhang, Yanwei Yue, Kun Wang, Junfeng Fang, Yongduo Sui, Kai Wang, Yuxuan Liang, Dawei Cheng, Shirui Pan, and Tianlong Chen. Two heads are better than one: Boosting graph sparse training via semantic and topological awareness. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 60197–60219. PMLR, 2024.

Guibin Zhang, Xiangguo Sun, Yanwei Yue, Chonghe Jiang, Kun Wang, Tianlong Chen, and Shirui Pan. Graph sparsification via mixture of graphs. In *International Conference on Learning Representations*, 2025.

Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. Robust graph representation learning via neural sparsification. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11458–11468. PMLR, 2020.

A Appendix

A.1 Proofs

Proof of Proposition 1 (Probe-Subspace Energy Preservation).

Proof. For $x = Xa$,

$$x^\top (L - \widetilde{L})x = (Xa)^\top (L - \widetilde{L})Xa$$

By Cauchy–Schwarz,

$$|x^\top (L - \widetilde{L})x| \le \|Xa\|_2 \|(L - \widetilde{L})Xa\|_2.$$

Moreover,

$$\|(L - \widetilde{L})Xa\|_2 \le \|(L - \widetilde{L})X\|_F \|a\|_2 \le \varepsilon \|a\|_2.$$

Combining the two inequalities gives the result. □

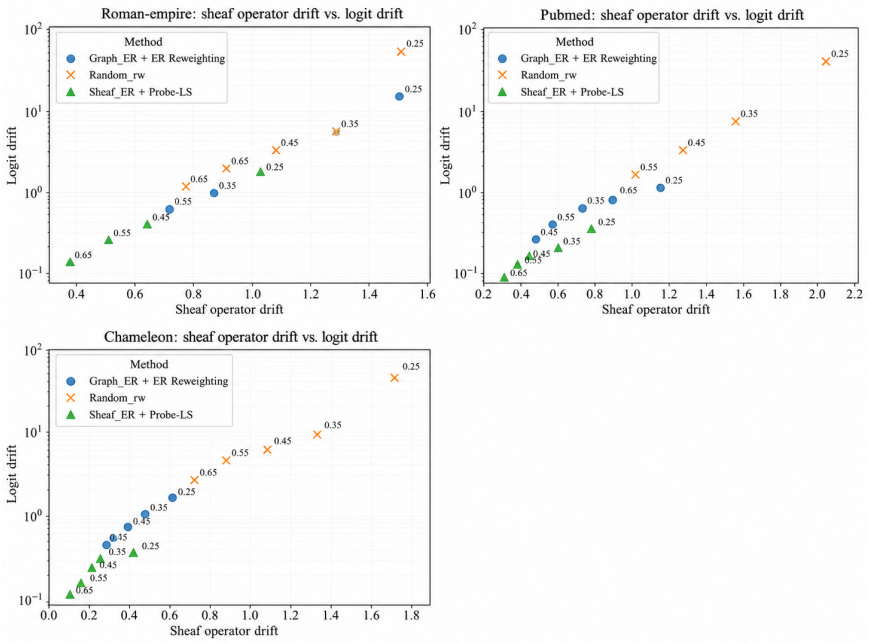


Figure 4: Dataset-wise relationship between sheaf operator drift and logit drift. Each point corresponds to a method and keep ratio. Probe-LS generally moves points toward the lower-left region, indicating better learned operator preservation and improved logit fidelity. The y-axis is shown on a logarithmic scale.

Proof of Theorem 1 (One-Layer Representation Stability).

Proof. By adding and subtracting $\Phi(\widetilde{L}, H)$ and applying the triangle inequality,

$$\begin{aligned}\|H^+ - \widetilde{H}^+\|_F &= \|\Phi(L,H) - \Phi(\widetilde{L},\widetilde{H})\|_F \\ &\le \|\Phi(L,H) - \Phi(\widetilde{L},H)\|_F + \|\Phi(\widetilde{L},H) - \Phi(\widetilde{L},\widetilde{H})\|_F \\ &\le C_L\|(L-\widetilde{L})H\|_F + C_H\|H-\widetilde{H}\|_F.\end{aligned}$$

□

Proof of Corollary 1 (Multi-Layer Representation Stability).

Proof. The recurrence follows directly from Theorem 1 applied at layer k:

$$\Delta_{k+1} \le C_H^{(k)}\Delta_k + C_L^{(k)}\epsilon_k.$$

Starting from $\Delta_0 = 0$ and recursively substituting the preceding bounds yields

$$\Delta_K \le \sum_{k=0}^{K-1} C_L^{(k)} \left(\prod_{j=k+1}^{K-1} C_H^{(j)} \right) \epsilon_k.$$

□

Proof of Corollary 2. The result follows directly from the Lipschitz continuity of g.

B Derivation of Probe-LS

This appendix derives the Probe-LS objective from the goal of preserving the learned propagation operator of a trained Neural Sheaf Diffusion (NSD) model.

B.1 MOTIVATION

Let L denote the learned dense sheaf operator and let $\widetilde{L}$ denote its sparse approximation constructed from a fixed sparse support. Our objective is not to reconstruct L itself, but to preserve its action on the representations that actually occur during the forward pass of the trained model.

Let

$$X = [x_1, \ldots, x_m] \in \mathbb{R}^{n \times m}$$

be a probe matrix whose columns are representative hidden representations extracted from the trained dense model. Ideally, the sparse operator should satisfy

$$LX \approx \widetilde{L}X.$$

This criterion focuses approximation on the representation subspace encountered during inference rather than the entire ambient space.

B.2 LEAST-SQUARES FORMULATION

Suppose that a sparse support $S \subseteq E$ has already been selected. The sparse operator is written as

$$\widetilde{L} = \sum_{e \in S} \alpha_e L_e,$$

where $L_e = B_e^\top B_e$ denotes the contribution of edge e and α_e is its reconstruction weight.

Substituting this expression into the operator-preservation objective yields

$$\min_{\alpha} \left\| LX - \sum_{e \in S} \alpha_e L_e X \right\|_F^2.$$

This is precisely the representation-aware least-squares objective used by Probe-LS.

Since each edge contribution is positive semidefinite, we additionally impose

$$\alpha_e \geq 0,$$

which guarantees that the reconstructed sparse operator remains positive semidefinite.

The resulting optimization problem is therefore

$$\alpha^\star = \arg\min_{\alpha_e \geq 0} \left\| LX - \sum_{e \in S} \alpha_e L_e X \right\|_F^2.$$

B.3 EQUIVALENT MATRIX FORM

The objective can be rewritten as a standard nonnegative least-squares (NNLS) problem.

Define

$$b = \mathrm{vec}(LX),$$

and

$$a_e = \mathrm{vec}(L_e X),$$

where $\mathrm{vec}(\cdot)$ stacks a matrix into a column vector.

Construct the matrix

$$A = \begin{bmatrix} a_1 & a_2 & \cdots & a_{|S|} \end{bmatrix}.$$

The optimization then becomes

$$\alpha^\star = \arg\min_{\alpha \geq 0} \|A\alpha - b\|_2^2,$$

which is a standard NNLS problem.

This formulation makes explicit that Probe-LS computes the nonnegative combination of retained edge operators whose action best matches that of the dense learned operator on the probe representations.

B.4 Optimization

Rather than solving the NNLS problem exactly, we optimize the objective using Adam. Unless otherwise specified, all experiments use

- learning rate: 0.05,
- 50 optimization steps,
- ℓ_2 regularization: 10^{-4},
- maximum edge weight: 10,
- initialization from graph effective-resistance reweighting.

These settings are fixed across all benchmark datasets.

B.5 Interpretation

Probe-LS can be interpreted as projecting the dense learned operator onto the cone generated by the retained edge operators,

$$\mathcal{C} = \left\{ \sum_{e \in S} \alpha_e L_e \;\middle|\; \alpha_e \geq 0 \right\},$$

Here the inner product is induced by the probe representations. Classical effective-resistance reweighting depends solely on sampling probabilities. By contrast, Probe-LS directly optimizes the sparse operator's action on representations encountered by the trained model, making it well suited to no-retraining sparsification.

C Experimental Details

Random seeds and repeated runs Unless otherwise specified, we evaluate no-retraining sparsification using a fixed NSD model trained with model seed 43. For each dataset, we repeat graph sparsification using 10 sampling seeds (0–9) and report the mean and standard deviation over these runs. **Computing environment** Each run used a single NVIDIA GeForce RTX 3090 GPU with 24 GB of memory. Experiments were conducted on Ubuntu 22.04.5 LTS using Python 3.9.9, PyTorch 2.0.0+cu118 with CUDA 11.8, and PyTorch Geometric 2.6.1. The server was equipped with Intel Xeon Gold 6242R CPUs and 503 GiB of RAM.

Statistical Significance To evaluate whether the improvements introduced by Probe-LS are statistically significant, we performed paired Wilcoxon signed-rank tests comparing Probe-LS with natural reweighting on the same sparse support. For both graph-based and sheaf-based supports, Probe-LS

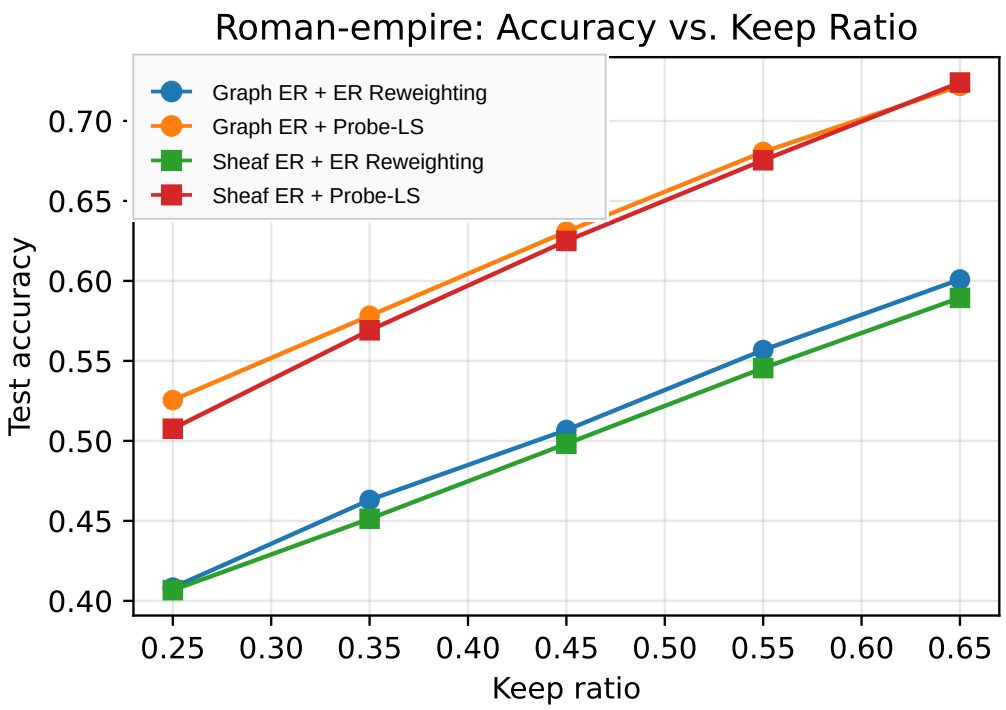


Figure 5: Test accuracy on Roman-empire before and after Probe-LS reweighting. Although graph effective-resistance support starts from a slightly stronger natural baseline, Probe-LS consistently improves both support-selection strategies across all keep ratios.

significantly improves test accuracy while simultaneously reducing the learned sheaf operator drift at every evaluated keep ratio ($p < 0.001$ in all cases). These results indicate that the observed improvements are consistent across all random sparsification trials rather than being caused by sampling variability.

D Additional Discussion on Support Selection

The support-selection ablation exhibits a slightly different trend on Roman-empire than on the other benchmark datasets. While sheaf effective-resistance support generally performs better on Squirrel, graph effective-resistance support obtains slightly higher accuracy than sheaf effective-resistance support on Roman-empire for most keep ratios.

To better understand this behavior, Figure 5 compares the test accuracy before and after Probe-LS reweighting under both support-selection strategies. Although graph-based support starts from a slightly stronger baseline under ER reweighting, Probe-LS consistently improves both graph-based and sheaf-based supports across all sparsity levels. The improvement introduced by Probe-LS is remarkably stable, yielding approximately 0.12 absolute accuracy gain for both support-selection strategies.

Figure 6 shows the corresponding learned sheaf operator drift. Without Probe-LS, both support-selection strategies exhibit substantially larger operator drift. After Probe-LS reweighting, however, the operator drift is reduced dramatically for both supports, and the resulting curves become nearly indistinguishable. This observation indicates that Probe-LS effectively compensates for differences in the initial sparse support by reconstructing sparse operators with comparable operator fidelity.

Taken together, these observations suggest that the slight difference between graph-based and sheaf-based support on Roman-empire is largely dominated by the subsequent operator-aware reweighting stage. Support selection determines the candidate sparse operator, whereas Probe-LS calibrates the

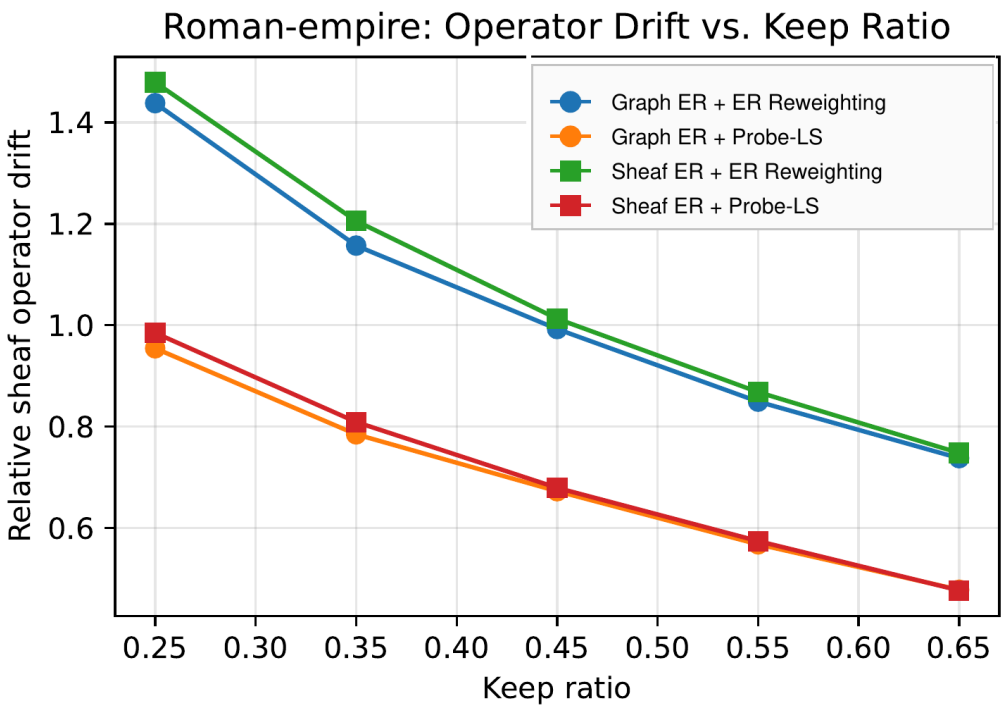


Figure 6: Relative sheaf operator drift on Roman-empire. Probe-LS substantially reduces the operator drift for both graph-based and sheaf-based supports, resulting in nearly identical operator fidelity across all keep ratios.

retained edge weights to approximate the learned propagation operator. Consequently, although the relative ranking of different support-selection strategies may vary across datasets, Probe-LS consistently reconstructs sparse operators with similar operator fidelity and delivers stable improvements in downstream performance.

E Hyperparameter Details

All experiments were conducted using the officially released Neural Sheaf Diffusion implementation with dataset-specific hyperparameters for training the dense models. The reported sparsification results were obtained from the trained checkpoints without further retraining.

E.1 Dense Model Training

The training hyperparameters for all four benchmarks appear in Table 4. Unless noted otherwise, each model uses the BundleSheaf architecture with the following settings:

```
normalised=True     left_weights=True
right_weights=True  edge_weights=True
                use_act=True
```

Optimization uses Adam. The `sheaf_decay` setting regularizes the sheaf learner; `weight_decay` regularizes all remaining parameters.

The dense-model hyperparameters were selected independently for each dataset following the original Neural Sheaf Diffusion training protocol. For Chameleon and Squirrel, the reported settings correspond to the best-performing configuration obtained during hyperparameter search. For PubMed, we adopted the same stable configuration used for Citeseer. For Roman-empire, the final configura-

Table 4: Dense NSD training hyperparameters used for the reported experiments.

	Chameleon	Squirrel	PubMed	Roman-empire
Architecture	BundleSheaf	BundleSheaf	BundleSheaf	BundleSheaf
d	3	3	2	2
Layers	5	5	2	6
Hidden dim	32	32	64	128
Learning rate	0.01	0.01	0.01	0.005
Weight decay	1.12×10^{-4}	1.12×10^{-4}	1×10^{-4}	1×10^{-5}
Sheaf decay	1.12×10^{-4}	1.12×10^{-4}	1×10^{-4}	4×10^{-5}
Input dropout	0.7	0.7	0.5	0.3
Dropout	0	0	0.3	0.5
Epochs	1000	1000	500	1200
Early stopping	100	100	200	250
Stopping metric	Accuracy	Accuracy	Loss	Accuracy
Random seed	43	43	3	43

tion was selected using successive-halving hyperparameter search. No hyperparameter tuning was performed for Probe-LS; the same sparsification hyperparameters were used across all datasets.

E.2 Sparsification and Probe-LS

Unless otherwise specified, the same sparsification hyperparameters were used for all datasets.

- Target keep ratios: $\{0.25, 0.35, 0.45, 0.55, 0.65\}$.
- Sparse supports were generated using sampling with replacement, where the sampling budget was adjusted by binary search to match the target unique-edge ratio.
- Graph effective resistance was computed using conjugate gradients with tolerance 10^{-5}, maximum 500 iterations, and a Jacobi preconditioner.
- Probe-LS used 16 Rademacher probes, 50 optimization steps, learning rate 0.05, ℓ_2 regularization 10^{-4}, maximum edge weight 10, and graph-ER reweighting as initialization.
- For large graphs and multi-layer support selection, sheaf effective resistance was approximated using randomized projections with 32 probes and a uniform mixing coefficient of 0.05.