Deep Neural Sheaf Diffusion

Rémi Bourgerie [1] **Šarūnas Girdzijauskas** [1] **Viktoria Fodor** [1]

Abstract

Deep Graph Neural Networks (GNNs) are essential for capturing complex dependencies in graph-structured data. However, scaling GNNs to depth remains challenging, as stacking layers leads to representation collapse and diminishing sensitivity due to repeated aggregation. While Neural Sheaf Diffusion (NSD) provides strong theoretical guarantees against such collapse, these guarantees do not translate to practice: as depth increases, the disagreement signal of the sheaf Laplacian vanishes, limiting the contribution of deeper layers. We identify mechanisms that hinder NSD effectiveness at depth and propose *Deep Neural Sheaf Diffusion* (DNSD), which replaces the sheaf Laplacian with a sheaf adjacency operator to maintain informative signals across layers. This is complemented by normalization, odd nonlinearities, and gating. To provide a principled explanation of the expected performance improvement, we contrast sheaf diffusion to graph attention mechanisms, highlighting that DNSD replaces scalar attention scores with matrix-valued edge functions and normalizes node representations rather than attention scores. We demonstrate empirically that DNSD effectively utilizes deep aggregation in graph tasks, outperforming GNN and NSD baselines with up to 30pp accuracy on synthetic long-range datasets, and consistently outperforming them on real-world benchmarks. These results position sheaf-based architectures as a promising building block for graph foundation models by supporting effective deep architectures. Code available at: `https://github.com/remibourgerie/deep-neural-sheaf-diffusion`.

[1]School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden. Correspondence to: Rémi Bourgerie <remibo@kth.se>.

Accepted at the Workshop on Graph Foundation Models at the 43rd International Conference on Machine Learning (ICML 2026), Seoul, South Korea.

1. Introduction

Learning expressive representations on graphs requires architectures that propagate and transform information across multiple layers. However, training deep architectures is inherently difficult due to issues such as vanishing updates and unstable optimization. In domains such as language and vision, these challenges have been mitigated by architectural design principles including residual connections, normalization, and gating (He et al., 2016; Ba et al., 2016; Qiu et al., 2025). In contrast, how to achieve similar depth scaling in GNNs remains far less understood (Bechler-Speicher et al., 2026).

Beyond these general challenges of deep models, GNNs exhibit a distinct difficulty arising from the message passing paradigm itself. Each layer aggregates information from neighbors through weighted averaging, and increasing depth results in repeated aggregation across the graph. In most message passing architectures, this aggregation takes the form of a convex combination of neighbor representations (Kipf & Welling, 2017; Veličković et al., 2018). As this process is composed across layers, node representations become progressively less distinguishable, a phenomenon known as oversmoothing (Chen et al., 2020). At the same time, repeated aggregation reduces the sensitivity of node representations to individual inputs, a phenomenon often referred to as oversquashing (Di Giovanni et al., 2023). These effects reflect a deeper limitation: standard message passing is constrained to convex aggregation, which progressively reduces both representation diversity and sensitivity as depth increases. This suggests that the challenge is not only to stabilize deep architectures, but to move beyond convex aggregation as the interaction mechanism between nodes.

This limitation directly impacts the development of graph foundation models, which rely on depth to capture long-range structure and learn expressive, generalizable representations across tasks and datasets (Wang et al., 2025). Improving depth scaling is therefore a necessary architectural step toward graph foundation models.

To address this bottleneck, a natural direction is to move beyond scalar aggregation and generalize how information is combined across edges. Network sheaves (Curry, 2014) provide a principled topological framework for modeling information flow on graphs by equipping edges with lin-

ear maps, allowing interactions to depend on richer local structure.

In contrast to standard message passing, where messages are scalar-weighted by the graph structure or attention, *Neural* Sheaf Diffusion (NSD) (Bodnar et al., 2022) exploits learned, edge-specific linear maps to transform node representations along edges, through a non-linear diffusion process governed by the sheaf Laplacian.

These richer edge interactions lead to strong theoretical guarantees: in the linear setting, given an appropriate sheaf, the stationary distribution of sheaf diffusion can separate almost any label assignment, preventing representation collapse under repeated aggregation (Bodnar et al., 2022). These results suggest that sheaf-based architectures provide a promising foundation for building deep and expressive GNNs.

However, despite these guarantees, existing NSD architectures do not fully realize this potential in practice. We identify several mechanisms that limit their scalability to depth. Most importantly, the sheaf Laplacian measures disagreement between connected nodes. As diffusion reduces this disagreement, the update signal progressively vanishes, causing deeper layers to receive diminishing inputs. Beyond this signal degeneration, repeated composition introduces additional challenges. First, the scale of representations can vary across layers, leading to instability as depth increases. Second, standard nonlinearities such as ReLU introduce an asymmetric truncation, suppressing negative values under repeated composition. Finally, repeated aggregation propagates all contributions uniformly, allowing noise to accumulate across layers.

In this paper we address these challenges by proposing DNSD, a sheaf diffusion architecture that scales to depth, realizing the potential of neural sheaf diffusion. Our main contributions are as follows:

1. We propose DNSD, a modified sheaf diffusion architecture for deep GNNs. DNSD replaces the sheaf Laplacian with a sheaf adjacency operator, shifting from measuring difference to aggregating dependency between neighboring representations. This is complemented by normalization, odd nonlinearities, and gating.
2. We contrast sheaf diffusion with graph attention mechanisms, highlighting that DNSD uses matrix-valued edge functions instead of scalar attention scores, and normalizes node representations rather than attention scores. These differences explain why DNSD maintains informative signals under deep aggregation.
3. We evaluate DNSD on a synthetic benchmark designed to probe depth scaling, as well as on real-world datasets. We show that DNSD improves performance in deep architectures, and remains effective as depth increases, with the adjacency formulation as the dominant factor. These results suggest that adjacency-based sheaf diffusion is a promising building block for graph foundation model architectures that scale in depth.

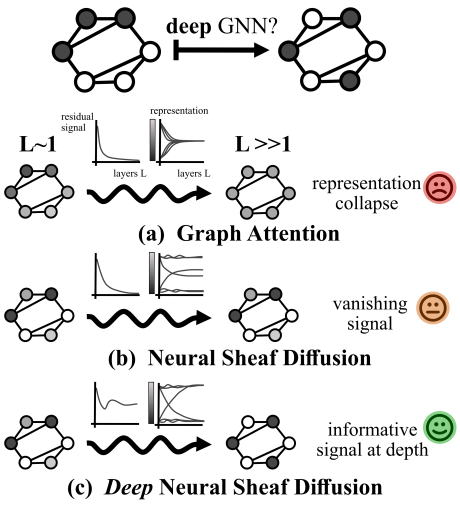

Figure 1. As depth increases ($L \gg 1$), standard message passing mechanisms degrade: graph attention (a) collapses representations, while neural sheaf diffusion (b) produces vanishing signals that limits the effective depth. Deep Neural Sheaf Diffusion (c) maintains informative signal at depth.

The rest of the paper is organized as follows. Section 2 introduces the necessary background on sheaf diffusion. Section 3 presents the proposed DNSD architecture. Section 4 discusses its connection to attention mechanisms. Section 5 reports empirical results, and Section 6 concludes the paper.

2. Background and Related Work

Cellular sheaves on graphs. Sheaves provide a framework for describing how local data defined on a space can be consistently combined into global structure (Bredon, 1997). Cellular sheaves adapt this framework to discrete structures such as graphs (Curry, 2014), by associating vector spaces to nodes and linear maps to edges. In this work, we restrict to real vector spaces, yielding a linear-algebraic formulation amenable to computation. Accordingly, a cellular sheaf $C(\mathcal{G}, \mathcal{F})$ on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with n vertices is specified by the triple $\langle \mathcal{F}(v), \mathcal{F}(e), \mathcal{F}_{v \trianglelefteq e} \rangle$, where for each $v \in \mathcal{V}$ and $e \in \mathcal{E}$, $\mathcal{F}(v) = \mathbb{R}^d$ are node stalks, $\mathcal{F}(e) = \mathbb{R}^d$ are edge stalks, and $\mathcal{F}_{v \trianglelefteq e} : \mathcal{F}(v) \to \mathcal{F}(e)$ are linear restriction maps for each incident node–edge pair. In contrast to standard message passing in GNNs, where interactions are scalar-weighted by the graph, sheaf-based models allow node representations to be transformed along edges through learned linear maps.

Sheaf diffusion. This structure induces a sheaf Laplacian $L_{\mathcal{F}}$, which generalizes the graph Laplacian and quantifies disagreement between neighboring node representations after transformation by the edge maps (Hansen & Ghrist, 2019). Acting on stacked stalk signals $\mathbf{X} \in \mathbb{R}^{nd}$, it admits a block matrix representation with diagonal blocks $(L_{\mathcal{F}})_{uu} = \sum_{v \in \mathcal{N}(u)} \mathcal{F}_{u \trianglelefteq e}^{\top} \mathcal{F}_{u \trianglelefteq e}$ and off-diagonal blocks $(L_{\mathcal{F}})_{uv} = -\mathcal{F}_{u \trianglelefteq e}^{\top} \mathcal{F}_{v \trianglelefteq e}$ for $(u, v) \in \mathcal{E}$. Let $\Delta_{\mathcal{F}} = D_{\mathcal{F}}^{-1/2} L_{\mathcal{F}} D_{\mathcal{F}}^{-1/2}$ denote the normalized sheaf Laplacian. A linear diffusion process then iteratively reduces disagreement as

$$\mathbf{X}^{(t+1)} = (I_{nd} - \Delta_{\mathcal{F}})\, \mathbf{X}^{(t)}. \tag{1}$$

As $t \to \infty$, this process converges to $\ker(\Delta_{\mathcal{F}})$, corresponding to globally consistent sections. Bodnar et al. (2022) show that, given appropriate restriction maps, any node classification task can be reduced to achieving sufficient separability in this space. This property suggests that, in the linear setting, sheaf diffusion may avoid representation collapse induced by repeated aggregation.

Neural Sheaf Diffusion (NSD). In practice, the diffusion steps are finite and the restriction maps are learned from data. At each layer, they are computed from the current node embeddings, making the sheaf layer-dependent. Bodnar et al. (2022) parameterize this process as a neural architecture. Node features are first mapped to stalk representations $\mathbf{X}^{(0)} \in \mathbb{R}^{nd \times f}$, where d is the stalk dimension and f the feature width per stalk. For each node v, we denote $\mathbf{X}_v^{(l)} \in \mathbb{R}^{d \times f}$ as its stacked stalk representation. At layer l, restriction maps are inferred via

$$\mathcal{F}_{v \trianglelefteq e}^{(l)} = \mathrm{reshape}_{d \times d}\Big(\sigma\big(V^{(l)}(\mathrm{vec}(\mathbf{X}_v^{(l)}) \,\|\, \mathrm{vec}(\mathbf{X}_u^{(l)}))\big)\Big),$$

where $V^{(l)} \in \mathbb{R}^{d^2 \times 2df}$ is learnable and σ is a nonlinearity. The nonlinear diffusion step is then

$$\mathbf{X}^{(l+1)} = (1+\epsilon^{(l)})\mathbf{X}^{(l)} - \sigma\left(\Delta_{\mathcal{F}}^{(l)}\left(I_n \otimes W_1^{(l)}\right)\mathbf{X}^{(l)}\right) W_2^{(l)}, \tag{2}$$

where $W_1^{(l)} \in \mathbb{R}^{f \times f}$ and $W_2^{(l)} \in \mathbb{R}^{d \times d}$ act on feature and stalk dimensions, respectively. Restriction maps may be *full*, *orthogonal*, or *diagonal*, trading off expressivity and parameter efficiency.

Related work. Neural Sheaf Diffusion was introduced by Bodnar et al. (2022) as a generalization of message passing using edge-conditioned linear maps. Subsequent works have explored extensions including precomputed maps (Barbero et al., 2022a), complex-valued edge maps (Amado et al., 2025), attention-based variants (Barbero et al., 2022b), nonlinear sheaf diffusion (Zaghen et al., 2024), alternative communication schemes (Ribeiro et al., 2026), and multi-hop aggregation (Bamberger et al., 2025; Borgi et al., 2025). However, these approaches do not address the challenge of scaling sheaf-based architectures in depth.

3. Deep Neural Sheaf Diffusion Architecture

Our goal is to design an architecture that maintains informative representations as the depth of aggregation increases.

To this end, we introduce DNSD, a modified sheaf diffusion architecture. The key modification is the replacement of the sheaf Laplacian with a sheaf adjacency operator, which prevents the vanishing of diffusion signals at depth. We complement this change with additional design elements — normalization, odd nonlinearities, and gating — that improve stability under repeated feature mixing. Detailed architectural specifications are provided in Appendix A.

We now describe the four modifications to the NSD update rule (2). The modified update (DNSD) for Deep Neural Sheaf Diffusion takes the form:

$$\begin{aligned}\mathbf{X}^{(l+1)} = \mathrm{LN}\Big(&(1+\epsilon^{(l)})\mathbf{X}^{(l)} \\ &- (\mathbf{G}^{(l)} \otimes \mathbf{1}_f^{\top}) \odot \sigma_{\mathrm{odd}}\Big(A_{\mathcal{F}}^{(l)}\mathbf{X}^{(l)}\, W_1^{(l)}\Big) W_2^{(l)}\Big)\end{aligned} \tag{3}$$

where $A_{\mathcal{F}}$ is the sheaf adjacency operator, σ_{odd} is an odd activation, LN denotes layer normalization applied independently to each node stalk (i.e., row-wise over $\mathbf{X}_v^{(l)} \in \mathbb{R}^{d \times f}$), and $\mathbf{G}^{(l)} \in [0,1]^{nd}$ is the output of node-level per-stalk gate.

(i) Sheaf Convolution The original NSD (2) applies the nonlinearity to $\Delta_{\mathcal{F}}\mathbf{X}$, i.e. the sheaf Laplacian of the current signal. In this setting, each diffusion step corresponds to a network layer, so increasing depth corresponds to iterating the diffusion process. Since $\Delta_{\mathcal{F}}$ measures disagreement between adjacent stalks, its output decreases as diffusion reduces this disagreement, eventually vanishing in the diffusion limit, i.e. at infinite depth (Bodnar et al., 2022). As a result, deeper layers tend to operate on progressively smaller residual signals and receive diminishing inputs to their nonlinearities.

This behavior reflects a limitation of using Laplacian-based updates in deep architectures. As depth increases, the model relies on $\Delta_{\mathcal{F}}\mathbf{X}$ to produce updates, yet this quantity is driven toward small magnitudes by the diffusion process itself. While this effect is formally established for linear diffusion, it remains a useful approximation of the behavior of neural sheaf diffusion in practice. Consequently, deeper layers can produce weaker updates and the loss can become less sensitive to their parameters. At initialization, a related issue appears: randomly initialized restriction maps can produce small and noisy disagreement signals, further reducing the effectiveness of deeper layers.

To address this limitation, we suggest to replace $\Delta_{\mathcal{F}}$ with $A_{\mathcal{F}}$, the sheaf adjacency operator, defined as block matrix with blocks $(A_{\mathcal{F}})_{uv} = \mathcal{F}_{u \trianglelefteq e}^{\top} \mathcal{F}_{v \trianglelefteq e}$ for $(u, v) \in \mathcal{E}$. This yields a sheaf convolution, where updates are computed

by aggregating dependency between neighboring representations rather than measuring residual disagreement. The resulting update $\sigma(A_{\mathcal{F}}\mathbf{X}W_1)W_2$ therefore operates on an informative signal, ensuring meaningful inputs to the nonlinearity both at initialization and throughout depth.

Remark 3.1 (Relation to prior formulations). Bodnar et al. (2022) introduced this sheaf convolution operator as part of a discretization of sheaf diffusion (their Eq. 4). However, their final architecture applies the nonlinearity only to the Laplacian term, effectively removing the identity component and feeding only the disagreement signal to the activation. As discussed above, this choice drives the signal toward small magnitudes at depth, while retaining the full adjacency operator preserves informative signals throughout the network.

(ii) Layer Normalization Replacing the Laplacian with the adjacency operator ensures that updates remain informative at depth. However, repeated application of these updates still introduces a second challenge: as depth increases, the scale of node representations can vary significantly across layers due to accumulated transformations and nonlinearities. This variation makes optimization unstable, as different layers operate on signals with inconsistent magnitudes.

To address this issue, layer normalization (Ba et al., 2016) is applied to the representation $\tilde{\mathbf{X}}_u^{(l)} \in \mathbb{R}^{d\times f}$ obtained *after* aggregation and residual addition, independently for each stalk by normalizing across the feature dimension f as

$$\mathrm{LN}^{(l)}(\tilde{\mathbf{X}}_u^{(l)}) = \mathbf{1}_d\gamma^{(l)\top} \odot \frac{\tilde{\mathbf{X}}_u^{(l)} - \mu_u\mathbf{1}_f^\top}{\sigma_u\mathbf{1}_f^\top} + \mathbf{1}_d\beta^{(l)\top} \quad (4)$$

where $\mu_u, \sigma_u \in \mathbb{R}^d$ are the per-stalk mean and standard deviation computed across the feature dimension f, and $\gamma^{(l)}, \beta^{(l)} \in \mathbb{R}^f$ are learnable parameters shared across all nodes and stalks. By normalizing each representation at every layer, LayerNorm stabilizes both the forward signal and the backward gradient, ensuring consistent magnitudes throughout depth. The learnable affine parameters $(\gamma^{(l)}, \beta^{(l)})$ allow the model to adapt the scale and offset of normalized features at each layer.

(iii) Odd and Bounded Activation Functions While the adjacency formulation preserves an informative signal at depth, deep architectures repeatedly apply the same transformation across layers. In this regime, the choice of nonlinearity directly affects how representations evolve under repeated composition.

The NSD in (2) uses ReLU as its pointwise nonlinearity. ReLU introduces an asymmetric truncation: negative activations are suppressed, while positive ones are preserved. In our setting, where the message term is subtracted from a residual, this results in a one-sided update that can only decrease coordinates. As depth increases, this asymmetric suppression accumulates, progressively reducing variation across features and inducing drift in the representation geometry.

To address this issue, we replace ReLU with an odd and bounded activation function σ_{odd} (specifically $\tanh$). Odd activations preserve sign symmetry, allowing both positive and negative interactions to contribute equally, while boundedness controls the magnitude of updates under repeated composition. Together, these properties maintain a balanced and stable transformation of representations across layers, which is particularly important when stacking many layers.

(iv) Node-Level Per-Stalk Gating While the previous modifications ensure that signals remain informative and transformations remain stable at depth, repeated weighted aggregation introduces an additional challenge: each layer combines contributions from neighboring nodes and stalk dimensions, and these contributions may contain noise. As depth increases, such noisy components can accumulate under repeated composition, reducing the quality of representations at increasing depth.

This phenomenon affects NSD, which applies updates uniformly across all representation dimensions as in (2). However, not all contributions are equally relevant at every node or layer. Under repeated composition, this uniform treatment propagates all components indiscriminately, allowing noisy contributions to persist and accumulate. This phenomenon is analogous to the *attention sink* effect observed in deep attention models (Xiao et al., 2024; Darcet et al., 2024), where certain components accumulate disproportionate signal without contributing meaningfully to the representation, thereby degrading its quality.

Therefore, we suggest to introduce a node-level gating mechanism that selectively modulates the contribution of each representation dimension at every layer, that is,

$$\left[(\mathbf{G}^{(l)})_u\right]_s = \mathrm{sigmoid}\left(w_g^{(l)}\begin{bmatrix}\mathbf{X}_{u,s}^{(l)}\\ \bar{\mathbf{X}}_{u,s}^{(l)}\end{bmatrix} + b_g^{(l)}\right) \in [0,1] \quad (5)$$

where $w_g^{(l)} \in \mathbb{R}^{1\times 2f}$ and $b_g^{(l)} \in \mathbb{R}$ are shared across all stalks, and $\mathbf{X}_{u,s}^{(l)}, \bar{\mathbf{X}}_{u,s}^{(l)} \in \mathbb{R}^f$ denote the s-th stalk of the current embedding and of the aggregated signal before σ, respectively. The gate modulates the update in (3) via $(\mathbf{G}^{(l)} \otimes \mathbf{1}_f^\top) \odot (\cdot)$, broadcasting each per-stalk scalar uniformly over the f feature dimensions.

This selective filtering limits the accumulation of noisy components under repeated composition, preserving representation quality when stacking layers.

Table 1. Comparison of message passing mechanisms along key design axes: the update operator (difference- vs dependency-based), the form of edge transformations (scalar vs matrix-valued), the normalization strategy, and the resulting behavior at depth.

Model	Update operator	Edge transformation	Normalization	Behavior at depth
GATs (Brody et al., 2022)	dependency (adjacency operator)	scalar (attention scores)	attention scores (softmax)	averaging tendency
NSD (Bodnar et al., 2022)	difference (laplacian)	linear map (matrix-valued)	operator (degree normalization)	vanishing signal
DNSD (ours)	dependency (adjacency operator)	linear map (matrix-valued)	operator + representations (LN)	mitigates signal decay

4. Sheaf Diffusion through the Lens of Graph Attention

We analyze DNSD through the lens of graph attention mechanisms, focusing on how it relates to existing message passing architectures. We use "GATs" to refer to the class of graph attention models analyzed in Brody et al. (2022).

In DNSD the residual update at layer l at node u can be written as $\sum_{v\in\mathcal{N}(u)}[\mathbf{A}_{\mathcal{F}}^{(l)}]_{uv}\mathbf{X}_v^{(l)}$, where $[\mathbf{A}_{\mathcal{F}}^{(l)}]_{uv}$ are edge operators computed by a learnable function of pairwise node representations $(\mathbf{X}_u^{(l)}, \mathbf{X}_v^{(l)})$. This update aggregates neighboring node representations using edge-dependent functions of node pairs, aligning it with attention mechanisms where aggregation weights are likewise computed from such pairwise interactions. However, DNSD differs from standard graph attention along several structural axes, as summarized in Table 1.

Update operator. Graph attention mechanisms and DNSD both aggregate *dependency* between neighboring node representations through adjacency-based operators. In contrast, NSD applies the Laplacian, which measures *difference* between neighboring node representations. While DNSD and attention share this dependency-based formulation, they differ in how these interactions are represented and combined, as discussed next.

Edge functions. Graph attention mechanisms and DNSD both compute edge-dependent functions from pairs of node representations. In attention, these functions map $(\mathbf{X}_u^{(l)}, \mathbf{X}_v^{(l)})$ to scalar attention scores that weight neighboring representations. In contrast, DNSD maps these pairs to linear maps $\mathcal{F}_{uv}^{(l)}$, which are applied directly to node representations.

Normalization. Attention mechanisms normalize scalar attention scores using softmax, which enforces non-negative weights that sum to one and results in convex aggregation of neighboring node representations. In contrast, DNSD does not normalize the aggregation operators, but instead applies normalization to node representations after aggregation. While DNSD inherits operator normalization from the sheaf construction, the dominant normalization occurs at the level of node representations, allowing aggregation to be non-convex.

Behavior at depth. These mechanisms lead to distinct behavior at increasing depth. In attention-based models, softmax normalization of scalar scores induces convex aggregation, resulting in an averaging tendency and potential representation collapse (Wu et al., 2023). In NSD, updates are computed from differences between neighboring representations, which decrease as diffusion progresses, leading to vanishing signals. DNSD mitigates both effects by combining dependency-based aggregation with representation-level normalization, maintaining informative signals across layers.

5. Experiments

In this section, we present the results of two sets of experiments, one with a synthetic long-range dataset, and one with real-world heterophilic benchmarks.

Synthetic Experiments. We design a dataset with provable long-lange dependence, suitable benchmarking deep MPNNs as follows. Building on the synthetic experiments from (Zaghen et al., 2024), we construct different sets of graphs based on the following process. We consider a family graph with 3 communities of 500 nodes each, with node features as 2D Gaussian features ($\sigma = 3$, shifted centers respectively $(0,0), (1,0), (0,1)$). The connectivity is defined by connecting each node to k-nearest neighbours ($k = 8$) within its community, forming three homophilic KNN graphs. We then gradually introduce heterophilic noise, in each step rewiring a fraction (10%) of the initial intra-community edges to random cross-community nodes. This leads to eleven perturbation levels from G0 (homophilic) to G10 (fully heterophilic). Solving this problem requires a deep receptive field of the GNN. On the one hand, guessing the right label must be performed by aggregating a sufficient number of features to compute the mean of community features. On the other hand, possible combinatorial ties have to be resolved for entire connected components. In the experiments training uses 6 graphs (9000 nodes, seeds 42–47) with 80/20 train/val split, testing uses 3 separate graphs (4500 nodes, seeds 100–102). Learning is evaluated based on the 3-class node classification accuracy on all test nodes.

We compare DNSD variants by gradually adding (i) adjacency, (iii) odd activation, and (iv) gating. For all DNSD

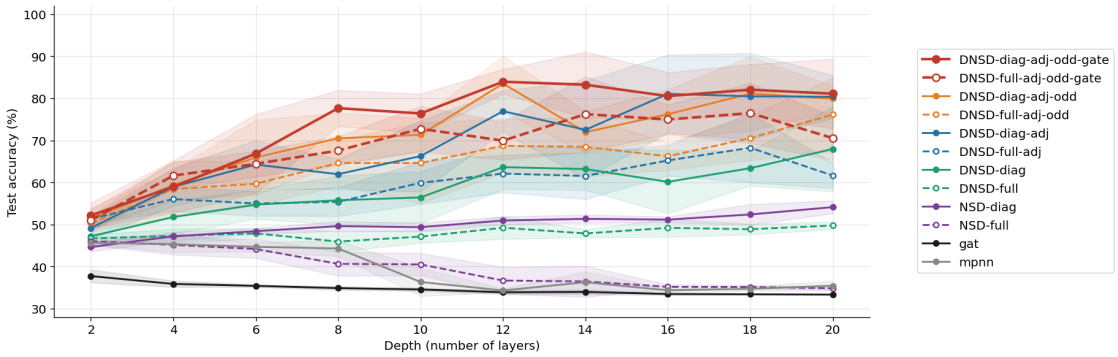

Figure 2. **Test accuracy on the synthetic community detection dataset G5 as a function of depth.** Each curve corresponds to a model variant, where `adj`, `odd`, and `gate` denote the use of adjacency-based operator, odd nonlinearities, and gating, respectively, and `diag` / `full` refer to diagonal or full restriction maps. Shaded regions represent one standard deviation over multiple runs.

variants, normalization is applied, as this is standard practice when training deep neural networks. We compare these models, using both full and diagonal maps, against *NSD* from (Bodnar et al., 2022) and the non-sheaf baselines MLP, MPNN (Gilmer et al., 2017), and GAT (Veličković et al., 2018). We restrict our experiments to diagonal and full restriction maps; orthogonal maps were found to be difficult to train robustly at depth in our setting, and we leave a systematic study of orthogonal parameterizations to future work. To assess the effect of depth, we evaluate all models across a range of layers $\{2, 4, 8, 12, 16\}$. Experimental details are given in Appendix B.1.

Real-World Experiments We evaluate on the heterophilic benchmark suite of Platonov et al. (2023) as a standard real-world setting, including large-scale graphs such as Penn94 (Lim et al., 2021). Following the concerns raised by Bechler-Speicher et al. (2025) regarding the ability of current benchmarks to isolate specific modeling factors, we use the synthetic benchmark to explicitly study depth, while the real-world benchmark serve as a complementary evaluation of robustness across diverse graph structures. We re-run NSD and DNSD under the same hyperparameter budget. Due to computational constraints, we restrict the number of layers to at most 8 and focus on diagonal maps, as motivated by results on synthetic benchmark in Table 2, which offer a favorable trade-off between performance and efficiency. Details can be found in Appendix B.2.

Results on synthetic experiments. Table 2 reports test accuracy across perturbation levels together with the depth of the best-performing models. Baselines perform poorly across all settings, often close to random guessing. For DNSD, the adjacency operator is the dominant factor: variants with adjacency outperform NSD by 20–30 percentage points at mid-heterophily levels (G3–G9), while variants without adjacency remain near the baseline. The odd activation provides additional gains, particularly at moderate heterophily (G2–G5), whereas gating mainly reduces variance across runs. Overall, the best performance is achieved by configurations combining adjacency, odd activation, and gating, followed by variants without gating, and then without both gating and odd activation.

Depth behavior. Figure 2 shows performance as a function of depth. DNSD improves with depth, reaching its best performance between 12 and 16 layers, consistent with Table 2. Similar trends are observed on real-world benchmarks (Table 3), where DNSD typically achieves its best performance at maximum depths 8. In contrast, graph attention models degrade at larger depths, while NSD shows limited improvement and often plateaus or decreases. DNSD remains effective at larger depths, unlike these baselines.

Results on real-world datasets. On real-world heterophilic benchmarks (Table 3), gains are smaller but remain consistent across datasets. DNSD (diag, Adj+Odd) achieves the best or second-best performance on 5 out of 9 datasets, with the largest improvements on Roman Empire (+4.1pp) and Penn94 (+3.7pp over NSD diag). Across datasets, diagonal restriction maps consistently match or outperform full maps, despite having fewer parameters.

6. Discussion Future Directions and Conclusions

Discussion of architectural choices. The experimental results highlight a clear hierarchy among the proposed modifications. The dominant factor is the replacement of the Laplacian with the adjacency operator, which shifts the update from measuring difference to aggregating dependency between neighboring representations. This change consistently yields the largest gains and is the primary driver of improved behavior at depth. The use of diagonal restriction maps further shows that that high-dimensional restriction maps are not required for strong performance, and that sim-

Table 2. Synthetic community detection: test accuracy (%) across perturbation levels G0–G10 on the test sets. All DNSD variants include LayerNorm. Each entry reports mean ± std at the best layer depth (in parentheses) for that (level, model) pair. **1st**, **2nd**, **3rd** best per column.

		Fixes			Perturbation level										
	Map	Adj	Odd	Gate	G0	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
MLP	—				41.3±0.3(L2)	41.4±0.2(L2)	41.3±0.3(L2)	41.3±0.3(L2)	41.3±0.1(L2)	41.3±0.2(L2)	41.2±0.2(L2)	41.2±0.3(L2)	41.3±0.2(L2)	41.2±0.2(L2)	41.5±0.2(L2)
MPNN	—				53.9±2.0(L2)	47.0±1.0(L2)	49.6±3.3(L4)	47.7±1.2(L2)	47.1±1.5(L2)	45.6±0.4(L2)	45.0±1.1(L2)	45.6±0.7(L2)	46.6±0.5(L2)	50.2±1.0(L4)	94.7±2.2(L8)
GAT	—				**67.2±15.9(L8)**	45.6±2.1(L4)	47.9±2.0(L4)	43.9±1.2(L4)	38.6±2.2(L4)	37.7±1.5(L2)	36.7±1.8(L2)	36.1±1.0(L2)	41.8±1.1(L2)	46.7±1.1(L4)	64.8±2.0(L4)
NSD	diag				53.0±16.4(L12)	51.0±6.4(L16)	55.8±0.9(L12)	54.7±2.3(L16)	51.2±2.1(L16)	51.2±0.7(L16)	49.1±1.7(L16)	49.1±1.2(L12)	50.1±1.0(L16)	50.5±1.5(L16)	85.5±4.7(L16)
	full				53.1±5.6(L16)	46.4±1.4(L2)	49.2±1.8(L4)	46.0±0.5(L2)	46.4±1.2(L2)	46.1±0.9(L2)	45.2±0.7(L2)	45.5±0.4(L2)	46.7±0.7(L2)	49.3±1.1(L8)	84.0±4.0(L12)
DNSD (diag)	diag	✓	✓	✓	45.7±10.1(L16)	51.0±9.1(L4)	**73.9±5.7(L12)**	**83.5±4.3(L12)**	**82.4±7.3(L12)**	**83.9±2.9(L12)**	**86.1±1.8(L12)**	**81.5±5.5(L16)**	**75.6±4.7(L16)**	**63.4±4.4(L12)**	96.2±1.3(L16)
	diag	✓	✓		51.1±6.1(L16)	**57.3±3.5(L12)**	**74.5±1.9(L12)**	**75.4±11.1(L12)**	**79.6±8.7(L12)**	**83.5±6.6(L12)**	**83.3±4.5(L12)**	**82.2±4.5(L16)**	**77.1±2.0(L16)**	**64.3±1.8(L16)**	96.7±0.7(L16)
	diag	✓		✓	48.0±9.5(L4)	49.9±7.6(L16)	58.2±1.8(L16)	62.0±10.1(L8)	**72.5±6.2(L12)**	**80.4±9.4(L12)**	**81.0±8.0(L12)**	**77.3±5.2(L12)**	**74.4±6.3(L12)**	**61.0±3.3(L12)**	95.7±1.4(L16)
	diag		✓	✓	**59.2±20.9(L12)**	51.0±13.2(L16)	63.3±6.2(L16)	**65.5±14.9(L16)**	66.9±8.0(L12)	71.4±11.8(L16)	71.1±7.5(L16)	70.6±7.4(L16)	62.8±5.2(L16)	55.5±4.3(L16)	95.3±2.6(L16)
	diag				50.4±6.4(L16)	49.8±5.0(L16)	54.9±4.1(L4)	53.5±5.3(L16)	60.8±9.8(L16)	63.6±5.3(L12)	60.4±7.7(L16)	58.8±6.2(L16)	57.8±4.5(L16)	53.4±1.7(L16)	96.4±0.8(L16)
DNSD (full)	full	✓	✓	✓	52.5±4.4(L12)	52.5±4.8(L12)	65.4±3.3(L12)	63.3±4.9(L16)	69.0±3.7(L16)	75.0±3.5(L16)	72.4±5.1(L16)	69.0±2.6(L16)	64.6±4.4(L16)	56.1±1.9(L12)	**97.1±0.7(L16)**
	full	✓	✓		52.1±2.0(L12)	**54.0±3.9(L16)**	**64.3±4.4(L16)**	61.2±3.2(L12)	63.0±3.4(L16)	68.7±6.4(L12)	63.5±5.8(L16)	59.5±2.3(L12)	55.3±2.8(L12)	53.2±1.3(L16)	**97.3±1.3(L16)**
	full	✓		✓	51.7±2.3(L16)	**53.3±4.8(L8)**	61.7±4.9(L8)	61.7±5.4(L12)	68.0±11.0(L16)	73.0±4.7(L16)	66.2±7.7(L16)	63.9±5.3(L16)	63.3±5.0(L16)	55.2±1.6(L16)	**97.5±0.8(L16)**
	full		✓	✓	**54.3±5.6(L16)**	51.4±4.4(L16)	58.9±1.7(L16)	56.6±2.7(L16)	57.2±4.3(L16)	58.0±2.6(L12)	54.3±5.5(L12)	53.7±3.2(L16)	53.1±2.7(L16)	52.7±1.0(L12)	95.5±1.6(L16)
	full				47.5±0.8(L12)	47.9±4.1(L2)	52.4±1.6(L16)	49.0±2.4(L16)	50.3±3.2(L16)	49.2±2.7(L12)	46.6±2.5(L16)	46.4±1.8(L16)	47.6±1.1(L16)	50.4±0.9(L16)	96.7±1.9(L16)

Table 3. Node classification accuracy (%) on heterophilic benchmarks. All DNSD variants use diagonal restriction maps and include LayerNorm. Each entry reports mean ± std at the best layer depth (in parentheses) for that (dataset, model) pair. **1st**, **2nd**, **3rd** best per column.

		Fixes			Dataset					
	Map	Adj	Odd	Gate	Roman Empire	Amazon Ratings	Minesweeper	Tolokers	Questions	Penn94
MLP	—				66.4±0.1(L2)	40.9±0.4(L2)	80.0±0.0(L2)	78.2±0.0(L2)	97.0±0.0(L2)	76.2±0.0(L2)
MPNN	—				78.9±0.8(L2)	46.6±0.3(L4)	87.4±1.3(L4)	79.1±0.1(L2)	97.0±0.0(L2)	—
GAT	—				56.9±1.1(L2)	46.0±0.6(L2)	80.3±0.0(L2)	78.4±0.1(L4)	97.0±0.0(L2)	74.1±2.0(L2)
NSD	diag				79.1±0.5(L8)	44.6±0.3(L8)	87.5±0.6(L8)	81.5±0.2(L8)	97.1±0.0(L2)	76.3±0.1(L2)
	full				77.1±0.9(L4)	45.4±0.7(L4)	86.1±0.3(L4)	81.4±0.3(L2)	97.1±0.0(L2)	76.1±0.6(L4)
DNSD (diag)	diag	✓	✓	✓	**83.4±0.9(L8)**	**47.8±0.4(L8)**	**89.4±0.8(L8)**	**81.8±0.5(L4)**	97.1±0.0(L4)	**78.7±0.9(L8)**
	diag	✓	✓		**83.2±0.7(L8)**	**49.1±0.7(L8)**	**88.9±0.4(L8)**	**82.0±0.2(L8)**	97.1±0.0(L2)	**80.0±0.9(L8)**
	diag	✓		✓	83.0±0.4(L8)	**47.5±1.0(L4)**	88.1±0.8(L8)	**81.8±0.7(L8)**	97.1±0.0(L4)	**78.6±0.9(L8)**
	diag		✓	✓	**83.4±0.2(L8)**	46.0±0.4(L2)	**88.2±0.5(L8)**	81.5±0.4(L8)	**97.2±0.0(L2)**	76.0±0.6(L4)
	diag				82.7±0.5(L8)	46.8±0.5(L8)	87.6±0.6(L8)	81.1±0.4(L2)	**97.2±0.0(L8)**	76.5±0.4(L8)

pler per-stalk interactions suffice and are even beneficial in practice, consistent with Borgi et al. (2025). Additional components like odd nonlinearities and gating mainly act as stabilizing mechanisms: odd activations prevent biased geometric shift of representations under repeated composition, while gating reduces variance across runs without significantly affecting peak performance.

Limitations and Future Directions. DNSD is evaluated on node classification tasks on medium-scale graphs. This limits the assessment of its applicability across diverse graph learning problems. Extending the evaluation to graph-level tasks, link prediction, and larger-scale datasets is an important direction, particularly in the context of graph foundation models (Wang et al., 2025).

Our analysis of depth in real-world settings is limited by the lack of benchmarks that effectively probe deep GNN behavior, reflecting broader concerns that current graph benchmarks do not isolate properties such as depth and long-range dependencies (Bechler-Speicher et al., 2025). While the observed trends are consistent with the synthetic benchmark, current datasets do not allow for a systematic evaluation of deeper architectures. Developing benchmarks that explicitly capture long-range dependencies remains an important direction for future work.

The sheaf-based formulation introduces richer edge interactions, which may increase computational cost at scale. This raises questions about scalability and efficiency. Investigating these trade-offs in distributed settings, where sheaves can model inter-agent agreement constraints, is a promising direction for future work (Hanks et al., 2025; Bourgerie et al., 2026).

Conclusion. We introduce DNSD, a sheaf diffusion architecture for deep GNNs. The key insight is that effective depth requires dependency-based propagation together with explicit normalization of representations across layers. By replacing Laplacian updates with adjacency-based aggregation, DNSD yields architectures that remain effective as depth increases, as demonstrated on both synthetic and real-world benchmarks. These results position DNSD as a promising architectural direction for graph foundation models requiring long range propagation.

Acknowledgement

The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

References

Amado, C. L., Schwarz, T., Tian, Y., and Lambiotte, R. Complex-weighted convolutional networks: Provable expressiveness via complex diffusion. In *The Fourth Learning on Graphs Conference*, 2025.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv:1607.06450*, 2016.

Bamberger, J., Barbero, F., Dong, X., and Bronstein, M. M. Bundle neural networks for message diffusion on graphs. In *International Conference on Learning Representations*, pp. 76947–76972, 2025.

Barbero, F., Bodnar, C., de Ocáriz Borde, H. S., Bronstein, M., Veličković, P., and Liò, P. Sheaf neural networks with connection laplacians. In *Proceedings of Topological, Algebraic, and Geometric Learning Workshops 2022*, volume 196, pp. 28–36. PMLR, 2022a.

Barbero, F., Bodnar, C., de Ocáriz Borde, H. S., and Liò, P. Sheaf attention networks. In *NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations*, 2022b.

Bechler-Speicher, M., Finkelshtein, B., Frasca, F., M"uller, L., T"onshoff, J., Siraudin, A., Zaverkin, V., Bronstein, M. M., Niepert, M., Perozzi, B., Galkin, M., and Morris, C. Position: Graph learning will lose relevance due to poor benchmarks. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 81067–81089. PMLR, 2025.

Bechler-Speicher, M., Gottlieb, Y., Isakov, A., Abensur, D., Tavory, A., Haimovich, D., Guy, I., and Weinsberg, U. Billion-scale graph foundation models. *arXiv:2602.04768*, 2026.

Bodnar, C., Di Giovanni, F., Chamberlain, B. P., Liò, P., and Bronstein, M. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in GNNs. In *Advances in Neural Information Processing Systems*, volume 35, pp. 18527–18541, 2022.

Borgi, A., Silvestri, F., and Liò, P. Polynomial neural sheaf diffusion: A spectral filtering approach on cellular sheaves. *arXiv:2512.00242*, 2025.

Bourgerie, R., Girdzijauskas, S., and Fodor, V. From euclidean to graph-structured data: A survey of collaborative learning. *Transactions on Machine Learning Research*, 2026.

Bredon, G. E. *Sheaf Theory*, volume 170. Springer Science & Business Media, 1997.

Brody, S., Alon, U., and Yahav, E. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022.

Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., and Sun, X. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3438–3445, 2020.

Curry, J. M. *Sheaves, Cosheaves and Applications*. University of Pennsylvania, 2014.

Darcet, T., Oquab, M., Mairal, J., and Bojanowski, P. Vision transformers need registers. In *International Conference on Learning Representations*, 2024.

Di Giovanni, F., Giusti, L., Barbero, F., Luise, G., Liò, P., and Bronstein, M. M. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In *International Conference on Machine Learning*, pp. 7865–7885, 2023.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pp. 1263–1272, 2017.

Hanks, T., Riess, H., Cohen, S., Gross, T., Hale, M., and Fairbanks, J. Distributed multi-agent coordination over cellular sheaves. In *IEEE Conference on Decision and Control*, pp. 3057–3064, 2025.

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

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Lim, D., Hohne, F., Li, X., Huang, S. L., Gupta, V., Bhalerao, O., and Lim, S. N. Large-scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In *Advances in Neural Information Processing Systems*, volume 34, pp. 20887–20902, 2021.

Platonov, O., Kuznedelev, D., Diskin, M., Babenko, A., and Prokhorenkova, L. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *International Conference on Learning Representations*, 2023.

Qiu, Z., Wang, Z., Zheng, B., Huang, Z., Wen, K., Yang, S., Men, R., Yu, L., Huang, F., Huang, S., Liu, D., Zhou, J., and Lin, J. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Ribeiro, A., Tenório, A. L., Belieni, J., Souza, A. H., and Mesquita, D. Cooperative sheaf neural networks. In *International Conference on Learning Representations*, 2026.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Wang, Z., Liu, Z., Ma, T., Li, J., Zhang, Z., Fu, X., Li, Y., Yuan, Z., Song, W., Ma, Y., et al. Graph foundation models: A comprehensive survey. *arXiv:2505.15116*, 2025.

Wu, X., Ajorlou, A., Wu, Z., and Jadbabaie, A. Demystifying oversmoothing in attention-based graph neural networks. *Advances in Neural Information Processing Systems*, 36:35084–35106, 2023.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024.

Zaghen, O., Longa, A., Azzolin, S., Telyatnikov, L., Passerini, A., and Liò, P. Sheaf diffusion goes nonlinear: Enhancing GNNs with adaptive sheaf laplacians. In *Proceedings of the Geometry-grounded Representation Learning and Generative Modeling Workshop (GRaM)*, volume 251, pp. 264–276. PMLR, 2024.

A. DNSD Architecture Details

We give the complete layer-by-layer specification of DNSD, using the notation established in Section 2. Recall that each node v has a stalk representation $\mathbf{X}_v^{(l)} \in \mathbb{R}^{d\times f}$, where d is the stalk dimension (number of stalks) and f is the feature width per stalk, giving total per-node representation capacity $c = d \cdot f$.

Input projection. Raw node features $\mathbf{h}_v \in \mathbb{R}^F$ are projected into the stalk space:

$$\mathrm{vec}\Big(\mathbf{X}_v^{(0)}\Big) = \mathbf{W}_{\mathrm{in}}\,\mathbf{h}_v + \mathbf{b}_{\mathrm{in}}, \qquad \mathbf{W}_{\mathrm{in}} \in \mathbb{R}^{c\times F}. \tag{6}$$

Restriction maps. At each layer l, two independent restriction map builders compute per-edge matrices

$$\mathcal{F}^{(l)}_{\mathrm{src},e},\ \mathcal{F}^{(l)}_{\mathrm{tgt},e} \in \mathbb{R}^{d\times d}, \quad e = (u \to v), \tag{7}$$

from the concatenated node embeddings:

$$\mathcal{F}^{(l)}_{\bullet,e} = \mathrm{reshape}_{d\times d}\left(\phi\left(\mathbf{V}_\bullet^{(l)}\begin{bmatrix}\mathrm{vec}\Big(\mathbf{X}_u^{(l)}\Big)\\ \mathrm{vec}\Big(\mathbf{X}_v^{(l)}\Big)\end{bmatrix} + \mathbf{b}_\bullet^{(l)}\right)\right), \qquad \mathbf{V}_\bullet^{(l)} \in \mathbb{R}^{d^2\times 2c}, \tag{8}$$

where $\bullet \in \{\mathrm{src}, \mathrm{tgt}\}$ with independent parameters per layer. Three choices of ϕ are considered:

- **Full:** $\phi(\mathbf{z}) = \tanh(\mathbf{z})$ with $\mathbf{z} \in \mathbb{R}^{d^2}$, reshaped to $\mathbb{R}^{d\times d}$ by the outer reshape, with weights initialised at scale $1/\sqrt{2}$.
- **Diagonal:** $\phi(\mathbf{z}) = \mathrm{diag}(\tanh(\mathbf{z}))$ with $\mathbf{V}_\bullet^{(l)} \in \mathbb{R}^{d\times 2c}$ and $\mathbf{z} \in \mathbb{R}^d$; $\mathrm{diag}(\cdot)$ directly produces $\mathbb{R}^{d\times d}$, so the outer reshape is omitted.
- **Orthogonal:** $\phi(\mathbf{z}) = Q$ with $\mathbf{z} \in \mathbb{R}^{d^2}$, where the outer reshape and QR decomposition are absorbed into ϕ: $\mathbf{z}$ is reshaped to $\mathbb{R}^{d\times d}$, factored as $\mathbf{z} = QR$, and $Q \in \mathbb{R}^{d\times d}$ is the orthogonal factor ($Q^\top Q = I$). The raw matrix is the learnable parameter, with Q recomputed at each forward pass, ensuring $\mathcal{F}^{(l)}_{\bullet,e} \in O(d)$ without explicit constraints on the weights.

Sheaf diffusion. The per-edge coboundary for stalk $s \in \{1, \dots, d\}$ is:

$$\boldsymbol{\delta}_e^{(s,l)} = \big[\mathcal{F}^{(l)}_{\mathrm{src},e}\big]_{s,:}\,\mathbf{X}_u^{(l)} - \big[\mathcal{F}^{(l)}_{\mathrm{tgt},e}\big]_{s,:}\,\mathbf{X}_v^{(l)}, \quad \boldsymbol{\delta}_e^{(s,l)} \in \mathbb{R}^f. \tag{9}$$

With the **adjacency** flag, only the target term is retained:

$$\boldsymbol{\delta}_e^{(s,l)} = \big[\mathcal{F}^{(l)}_{\mathrm{tgt},e}\big]_{s,:}\,\mathbf{X}_v^{(l)}. \tag{10}$$

Messages are back-projected and symmetrically normalised:

$$\Big[\mathbf{m}^{(l)}_{u\leftarrow e}\Big]_{s,:} = \tilde{d}_u^{-1/2}\,\tilde{d}_v^{-1/2}\sum_{j=1}^{d}\big[\mathcal{F}^{(l)}_{\mathrm{src},e}\big]_{js}\,\boldsymbol{\delta}_e^{(j,l)}, \tag{11}$$

where $\tilde{d}_v = \max(1, \deg(v))$, and aggregated by summation: $\bar{\mathbf{X}}_v^{(l)} = \sum_{e\ni v}\mathbf{m}^{(l)}_{v\leftarrow e} \in \mathbb{R}^{d\times f}$. This corresponds to $\Delta^{(l)}_{\mathcal{F}}\mathbf{X}^{(l)}$ in the compact notation of (3) (or $A^{(l)}_{\mathcal{F}}\mathbf{X}^{(l)}$ with the adjacency formulation).

Stalk-wise update. The aggregated signal is transformed via $W_1^{(l)} \in \mathbb{R}^{d\times d}$ and $W_2^{(l)} \in \mathbb{R}^{f\times f}$:

$$\tilde{\mathbf{X}}_v^{(l)} = \sigma\Big(\bar{\mathbf{X}}_v^{(l)}\,W_1^{(l)}\Big)\,W_2^{(l)}, \qquad \tilde{\mathbf{X}}_v^{(l)} \in \mathbb{R}^{d\times f}, \tag{12}$$

where $\sigma = \mathrm{ReLU}$ by default, or $\sigma = \tanh$ with the **odd** flag. Both matrices are initialised as $\mathbf{I} + 0.01\,\mathcal{N}(0,1)$.

Stalk gating. With the **gate** flag, a scalar gate per stalk modulates the update. The gate is computed from the current embedding and the aggregated signal:

$$(\mathbf{G}^{(l)})_{v,s} = \mathrm{sigmoid}\left(w_g^{(l)} \begin{bmatrix} \mathbf{X}_{v,s}^{(l)} \\ \bar{\mathbf{X}}_{v,s}^{(l)} \end{bmatrix} + b_g^{(l)} \right) \in [0,1], \qquad w_g^{(l)} \in \mathbb{R}^{1\times 2f}, \tag{13}$$

with $w_g^{(l)}$ and $b_g^{(l)}$ zero-initialised (gate ≈ 0.5 at start).

The gated update is $(\mathbf{g}_v^{(l)} \otimes \mathbf{1}_f^\top) \odot \tilde{\mathbf{X}}_v^{(l)}$, consistent with the compact form in (3).

Residual and layer normalisation. Following (3), the full layer update is:

$$\mathbf{X}_v^{(l+1)} = \mathrm{LN}\left((1+\epsilon^{(l)})\, \mathbf{X}_v^{(l)} - (\mathbf{G}^{(l)})_v \otimes \mathbf{1}_f^\top) \odot \tilde{\mathbf{X}}_v^{(l)} \right), \tag{14}$$

where $\epsilon^{(l)} \in \mathbb{R}$ is a learnable scalar (zero-initialised) and LN is layer normalisation applied per stalk as in (4), normalising across the feature dimension f independently for each stalk.

Output classifier.

$$\hat{\mathbf{y}}_v = \mathbf{W}_{\mathrm{out}}\ \mathrm{vec}\left(\mathbf{X}_v^{(L)}\right) + \mathbf{b}_{\mathrm{out}}, \qquad \mathbf{W}_{\mathrm{out}} \in \mathbb{R}^{C\times c}, \tag{15}$$

where C represents the number of classes.

Summary of flags. Table 4 summarizes the DNSD architecture options and their realizations on the base NSD equation.

Table 4. DNSD architectural flags and their effect on the base update.

Flag	Off (default)	On
`adj`	Symmetric coboundary (Eq. 9)	Target-only (Eq. 10)
`odd`	$\sigma = \mathrm{ReLU}$ in Eq. 12	$\sigma = \tanh$
`gate`	$\mathbf{g}_v^{(l)} \equiv \mathbf{1}_d$	Per-stalk sigmoid gate (Eq. 13)

B. Experimental Details

In this section, we give further details about the experiment settings on the synthetic and real-world benchmark.

B.1. Synthetic Community Detection Benchmark

Benchmark Construction. We construct a synthetic community detection benchmark based on a K-nearest-neighbour (KNN) graph with $N = 1{,}500$ nodes partitioned into $K = 3$ communities of equal size (500 nodes each). Node features are 2-dimensional, sampled from Gaussian distributions centred at community-specific means with standard deviation $\sigma = 3.0$. The KNN graph is built with $k = 8$ neighbours; the intra-community edge probability of an equivalent SBM is set to match the average KNN degree. The heterophily level $L \in \{0, 1, \ldots, 10\}$ controls the degree of inter-community mixing via edge perturbation: at $L = 0$ the graph is fully homophilic, while at $L = 10$ edges are entirely inter-community. Train and test graphs are generated independently from disjoint random seeds to avoid data leakage and cached to disk for reproducibility. The train graph uses an 80/20 node-level train/validation split; evaluation is performed on the independently generated test graph over all nodes.

Dataset statistics. Each benchmark graph has $N = 1{,}500$ nodes, $\approx$7,257 edges, $F = 2$ node features, and $C = 3$ classes. The edge count is constant across levels since perturbation rewires edges rather than adding or removing them.

Hyperparameters. We use the Adam optimiser with learning rate $\eta = 0.01$ and weight decay $\lambda = 5 \times 10^{-4}$. The learning rate is reduced on plateau (factor 0.5, patience 20 epochs). The hidden dimension is fixed at $c = 18$ and the number of stalks at $d = 3$ (so $f = 6$, equal to K). We perform a grid search over the number of layers $\in \{2, 4, 8, 12, 16\}$. Training runs for a maximum of 500 epochs with early stopping (patience 100) monitored on validation accuracy; the best checkpoint is restored at the end of training. Results are reported as mean $\pm$ std over 6 random train seeds $\{42, 43, 44, 45, 46, 47\}$, evaluated on test graphs generated from 3 independent test seeds $\{100, 101, 102\}$.

Model complexity. Table 5 reports parameter counts at each model's selected depth (chosen by mean validation accuracy across all perturbation levels and training seeds). All models use c=18, d=3, f=6, and F=2 (2D Gaussian node features).

Table 5. Parameter counts for synthetic benchmark models (c=18, d=3, f=6, node features F=2, classes C=3). Total counts shown for a representative graph; backbone excludes input/output projections.

Model	Flags	L	Total	Backbone
MLP	–	2	111	–
GAT	–	4	909	–
NSD diag	–	16	2,655	2,544
NSD full	–	8	3,159	3,048
DNSD diag	–	16	5,007	4,896
	`adj`	16	5,007	4,896
	`odd`	16	5,007	4,896
	`gate`	16	5,215	5,104
	`adj+odd`	16	5,007	4,896
	`adj+gate`	12	3,939	3,828
	`odd+gate`	16	5,215	5,104
	`adj+odd+gate`	12	3,939	3,828
DNSD full	–	16	12,111	12,000
	`adj`	16	12,111	12,000
	`odd`	16	12,111	12,000
	`gate`	16	12,319	12,208
	`adj+odd`	16	12,111	12,000
	`adj+gate`	16	12,319	12,208
	`odd+gate`	16	12,319	12,208
	`adj+odd+gate`	16	12,319	12,208

Computational resources. Experiments were run on a standard CPU workstation over approximately two weeks of wall-clock time.

B.2. Real-World Benchmark

Datasets. We evaluate on 9 real-world node classification benchmarks: Actor, Chameleon (filtered), and Squirrel (filtered), roman-empire, amazon-ratings, minesweeper, tolokers, and questions (Platonov et al., 2023), and penn94 (Lim et al., 2021). The filtered Chameleon and Squirrel variants remove duplicate nodes following (Platonov et al., 2023).

Dataset statistics. Table 6 summarises the graph statistics. Heterophily is measured as the fraction of edges connecting nodes of different classes.

Table 6. Statistics of the real-world benchmark datasets.

Dataset	**Size**	**#Nodes**	**#Edges**	**#Features**	**#Classes**	**Heterophily**
Chameleon (filtered)	small	890	4,427	2,325	5	0.764
Squirrel (filtered)	small	2,223	23,499	2,089	5	0.793
Actor	medium	7,600	15,009	932	5	0.781
Minesweeper	medium	10,000	39,402	7	2	0.317
Tolokers	medium	11,758	519,000	10	2	0.405
Roman Empire	medium	22,662	32,927	300	18	0.953
Amazon Ratings	medium	24,492	93,050	300	5	0.620
Questions	large	48,921	153,540	301	2	0.160
Penn94	large	41,554	1,362,229	4,814	2	0.530

Data splits. All datasets use a fixed stratified random split of 60/20/20 (train/validation/test), stratified per class. Splits are generated with a dataset-name-derived fixed seed and cached to disk for reproducibility across runs.

Table 7. Node classification accuracy (%) on heterophilic benchmarks. All DNSD variants use diagonal restriction maps and include LayerNorm. Each entry reports mean ± std at the best layer depth (in parentheses) for that (dataset, model) pair. **1st**, **2nd**, **3rd** best per column.

		Fixes			Dataset								
	Map	Adj	Odd	Gate	Actor	Chameleon (filtered)	Squirrel (filtered)	Roman Empire	Amazon Ratings	Minesweeper	Tolokers	Questions	Penn94
MLP	—				**37.0**±0.3(G4)	**43.0**±3.7(G4)	**38.3**±0.6(G4)	66.4±0.1(G2)	40.9±0.4(G2)	80.0±0.0(G2)	78.2±0.0(G2)	97.0±0.0(G2)	76.2±0.0(G2)
MPNN	—				34.4±1.1(G2)	**45.5**±2.2(G4)	**43.1**±0.5(G2)	78.9±0.8(G2)	46.6±0.3(G4)	87.4±1.3(G4)	79.1±0.1(G2)	97.0±0.0(G2)	—
GAT	—				30.3±0.6(G2)	**45.2**±2.7(G4)	**37.0**±1.7(G8)	56.9±1.1(G2)	46.0±0.6(G2)	80.3±0.0(G2)	78.4±0.1(G4)	97.0±0.0(G2)	74.1±2.0(G2)
NSD	diag				**36.2**±0.6(G2)	41.5±1.2(G2)	35.4±1.5(G2)	79.1±0.5(G8)	44.6±0.3(G8)	87.5±0.6(G8)	81.5±0.2(G8)	97.1±0.0(G2)	76.3±0.1(G2)
	full				35.9±0.3(G2)	41.6±2.7(G4)	36.5±1.6(G4)	77.1±0.9(G4)	45.4±0.7(G4)	86.1±0.3(G4)	81.4±0.3(G2)	97.1±0.0(G2)	76.1±0.6(G4)
DNSD (diag)	diag	✓	✓	✓	**36.2**±0.6(G4)	37.7±1.1(G8)	33.5±1.3(G8)	**83.4**±0.9(G8)	**47.8**±0.4(G8)	**89.4**±0.8(G8)	**81.8**±0.5(G4)	97.1±0.0(G4)	**78.7**±0.9(G8)
	diag	✓	✓		36.0±0.7(G2)	37.9±0.9(G8)	32.8±1.0(G4)	**83.2**±0.7(G8)	**49.1**±0.7(G8)	**88.9**±0.4(G8)	**82.0**±0.2(G8)	97.1±0.0(G2)	**80.0**±0.9(G8)
	diag	✓		✓	**36.2**±0.6(G4)	37.9±0.9(G4)	33.1±1.2(G8)	83.0±0.4(G8)	**47.5**±1.0(G4)	88.1±0.8(G8)	**81.8**±0.7(G8)	97.1±0.0(G4)	**78.6**±0.9(G8)
	diag		✓	✓	**36.4**±0.4(G4)	38.0±0.6(G4)	33.5±1.3(G8)	**83.4**±0.2(G8)	46.0±0.4(G2)	**88.2**±0.5(G8)	81.5±0.4(G8)	**97.2**±0.0(G2)	76.0±0.6(G4)
	diag				36.1±0.4(G2)	39.1±0.9(G8)	33.1±1.4(G8)	82.7±0.5(G8)	46.8±0.5(G8)	87.6±0.6(G8)	81.1±0.4(G2)	**97.2**±0.0(G8)	76.5±0.4(G8)

Models. We evaluate DNSD with diagonal restriction maps across 8 variants obtained by combining three binary flags: adjacency operator (`adj`), odd activation (`odd`, tanh instead of ReLU), and per-stalk gating (`gate`). We also include NSD v1 (full and diagonal), MLP, and GAT as baselines.

Hyperparameters. We perform a grid search over the number of layers $\in \{2, 4, 8\}$, with fixed hidden dimension $c = 64$, number of stalks $d = 8$ (so $f = 8$), learning rate $\eta = 0.01$, and weight decay $\lambda = 5 \times 10^{-4}$. The maximum number of training epochs is 500. Best hyperparameters are selected per dataset–model combination based on validation accuracy.

Training. We use the Adam optimiser. The learning rate is reduced on plateau (factor 0.5, patience 20 epochs). Early stopping is applied with a patience of 100 epochs monitored on validation accuracy; the best checkpoint is restored at the end of training. All results are reported as mean ± std over 5 random seeds $\{42, 43, 44, 45, 46\}$. MPNN was excluded from the Penn94 benchmark due to memory overflow on the full graph.

Results. Table 3 reports test accuracy (%) per model and dataset. Each entry shows the mean ± std across seeds at the best layer depth for that (dataset, model) combination, selected by mean validation accuracy; the optimal depth is shown in parentheses. DNSD variants with diagonal restriction maps consistently outperform NSD v1 and standard GNN baselines on heterophilic datasets, with the adjacency flag (`adj`) providing the most consistent gains across datasets. On homophily-insensitive datasets such as Questions, the differences between the models are minimal.

Computational resources. Experiments were run using a node on the Dardel HPC cluster at PDC Center for High Performance Computing[1], equipped with 8 AMD MI250X GPUs, for a total of 70 GPU-core hours.

C. Sheaf Diffusion and Attention

In this section, we make explicit the connection between sheaf-based message passing and attention mechanisms by formalizing the edge-dependent functions and update operators used in NSD and DNSD. We use the term graph attention mechanisms (GATs) to refer to the family of models analyzed in Brody et al. (2022), which includes Graph Attention (GAT) (Veličković et al., 2018), its more expressive variants (GATv2) (Brody et al., 2022), and dot-product attention (Vaswani et al., 2017).

Edge-dependent functions. Both graph attention mechanisms and DNSD compute edge-dependent functions from pairs of node representations. At layer l, for a node u and its neighbor v, this takes the form

$$[\mathbf{A}_{\mathcal{F}}^{(l)}]_{uv} = \Phi^{(l)}\left(\mathbf{X}_u^{(l)}, \mathbf{X}_v^{(l)}\right),$$

[1] https://www.pdc.kth.se/

where $\Phi^{(l)}$ is a learnable function of the pair $(\mathbf{X}_u^{(l)}, \mathbf{X}_v^{(l)})$.

In graph attention mechanisms, $\Phi^{(l)}$ outputs a scalar attention score

$$e_{uv}^{(l)} = \Phi^{(l)}\left(\mathbf{X}_u^{(l)}, \mathbf{X}_v^{(l)}\right) \in \mathbb{R},$$

which is then normalized across neighbors using a softmax:

$$\alpha_{uv}^{(l)} = \frac{\exp(e_{uv}^{(l)})}{\sum_{w\in\mathcal{N}(u)} \exp(e_{uw}^{(l)})}.$$

The normalized coefficients $\alpha_{uv}^{(l)}$ are used to weight neighboring representations.

In contrast, DNSD outputs a linear map:

$$\mathbf{A}_{\mathcal{F},uv}^{(l)} \in \mathbb{R}^{d\times d},$$

which is applied directly to the representation $\mathbf{X}_v^{(l)}$.

Remark. The stalk dimension in DNSD can be viewed as playing a role loosely analogous to multiple attention heads, in that both enable multiple parallel interactions. However, the underlying mechanisms differ, as DNSD applies linear maps rather than scalar weightings.

Update operators. The distinction between NSD and DNSD lies in the operator used to compute updates. NSD applies the sheaf Laplacian:

$$\Delta_{\mathcal{F}}\mathbf{X},$$

which aggregates *differences* between neighboring node representations. In contrast, DNSD uses the sheaf adjacency operator:

$$\mathbf{A}_{\mathcal{F}}\mathbf{X},$$

which aggregates *dependencies* between neighbors.

Thus, NSD and DNSD differ in the signal used for aggregation, while DNSD and attention share a dependency-based formulation.

Aggregation mechanisms. Given edge-dependent functions, the update at node u takes the form

$$\sum_{v\in\mathcal{N}(u)} [\mathbf{A}_{\mathcal{F}}^{(l)}]_{uv}\mathbf{X}_v^{(l)}.$$

In attention-based models, scalar attention scores are normalized via softmax, producing convex combinations of neighboring representations. In DNSD, no normalization is applied to the aggregation operators, and updates are not restricted to convex combinations.

Normalization. Attention mechanisms normalize scalar attention scores using softmax:

$$\alpha_{uv}^{(l)} = \frac{\exp(e_{uv}^{(l)})}{\sum_{w\in\mathcal{N}(u)} \exp(e_{uw}^{(l)})}.$$

In DNSD, normalization is instead applied to node representations after aggregation via LayerNorm. While DNSD inherits operator normalization from the sheaf construction, the dominant normalization occurs at the level of node representations.

Summary. The comparison can be summarized along three axes: (i) the update operator (difference vs. dependency), (ii) the form of edge-dependent functions (scalar scores vs. linear maps), and (iii) the location of normalization (scores vs. representations). These structural differences explain the distinct behavior observed at depth in the main text.