

Polynomial Neural Sheaf Diffusion: A Spectral Filtering Approach on Cellular Sheaves

Alessio Borgi

Sapienza University of Rome, University of Cambridge
alessio.borgi@uniroma1.it

Fabrizio Silvestri

Sapienza University of Rome
fsilvestri@diag.uniroma1.it

Pietro Liò

University of Cambridge
pl219@cam.ac.uk

Abstract

Sheaf Neural Networks equip graph structures with a cellular sheaf: a geometric structure which assigns local vector spaces (stalks) and a linear learnable restriction/transport maps to nodes and edges, yielding an edge-aware inductive bias that handles heterophily and limits oversmoothing. However, common Neural Sheaf Diffusion implementations rely on SVD-based sheaf normalization and dense per-edge restriction maps, which scale with stalk dimension, require frequent Laplacian rebuilds, and yield brittle gradients. To address these limitations, we introduce *Polynomial Neural Sheaf Diffusion (PolyNSD)*, a new sheaf diffusion approach whose propagation operator is a degree- K polynomial in a normalised sheaf Laplacian, evaluated via a stable three-term recurrence on a spectrally rescaled operator. This provides an explicit K -hop receptive field in a single layer (independently of the stalk dimension), with a trainable spectral response obtained as a convex mixture of $K+1$ orthogonal polynomial basis responses. PolyNSD enforces stability via convex mixtures, spectral rescaling, and residual/gated paths, reaching new state-of-the-art results on both homophilic and heterophilic benchmarks, inverting the Neural Sheaf Diffusion trend by obtaining these results with just diagonal restriction maps, decoupling performance from large stalk dimension, while reducing runtime and memory requirements.

1 Introduction

Graph Neural Networks (GNNs) Goller and Kuchler [1996], Gori et al. [2005], Scarselli et al. [2008], Bruna et al. [2013], Defferrard et al. [2016], Velickovic et al. [2017], Gilmer et al. [2017] have become a standard tool for learning on relational data, yet they often underperform on *heterophilic* graphs Zhu et al. [2020] and suffer from *oversmoothing* Nt and Maehara [2019], Rusch et al. [2023] when stacked deeply. The former stems from the homophily bias of isotropic message passing (neighbors are presumed similar), while the latter arises from repeated low-pass aggregation that gradually washes out class-separating signals.

A possible solution provided by Bi et al. [2024] is to modify the graph by rewiring the graph with homophilic edges and pruning heterophilic ones. A more principled way to avoid that, modeling heterophily in the graph’s underlying topology, is via (cellular) *sheaves* Hansen and Gebhart [2020], Bodnar et al. [2022]: each node/edge carries a local feature space (a stalk) and edges carry linear restriction/transport maps that specify how to align and compare features across incidences. The resulting sheaf Laplacian implements *transport-aware* diffusion that can better accommodate heterophily than conventional, isotropic graph filters. However, existing neural sheaf-diffusion layers

are (i) effectively one-step propagators, (ii) rely on dense, per-edge restriction maps, (iii) require heavy normalizations or even decompositions during training, and (iv) their performance is highly dependent on the stalk’s high-dimensionality. These factors inflate parameters and runtime, couple cost and stability to the stalk dimension, and can yield brittle optimization.

We propose *Polynomial Neural Sheaf Diffusion (PolyNSD)*, which replaces one-step updates with a learnable polynomial filter of the (rescaled) sheaf Laplacian, evaluated by stable k -term recurrences (e.g., Chebyshev) using only sparse matrix–vector products. This yields an explicit K -hop receptive field in a single layer, and a tunable spectral response (low-/band-/high-pass) without eigendecompositions or SVDs. To promote stability, we combine spectral rescaling with convex mixing of the $K+1$ basis responses to obtain a non-expansive operator, and we include residual/gated paths that preserve gradient flow and counteract training brittleness. On top of that, PolyNSD is *parameter-efficient*: the propagation is controlled by only $K+1$ scalar coefficients per layer (optionally shared across channels), while reusing the learned transports rather than introducing new dense edge-wise maps. This decouples performance from large stalk dimensions, reduces memory and latency, and preserves the expressive, transport-aware bias that mitigates oversmoothing on challenging, heterophilic graphs.

Contributions. Our main contributions may be summarised as follows:

1. We introduce *PolyNSD*, a general framework for optimising the neural sheaf diffusion process employing *orthogonal-polynomial spectral filters* on a *spectrally rescaled* sheaf Laplacian, evaluated via stable recurrences. This module can be further processed with any existing architecture designed for sheaf data.
2. We demonstrate the benefit of modelling both homophilic and heterophilic data using a single unified model, achieving *state-of-the-art* results on several benchmarks of both types, with no need anymore to rely on tailored architectures for each of the cases.
3. We *reduce model complexity*, by inverting the prevailing Neural Sheaf Diffusion trend toward dense per-edge restrictions by showing that *diagonal restriction maps*, coupled with our polynomial filter, are sufficient to attain competitive and *superior accuracy*. This choice therefore *decouples empirical performance from the need to inflate high stalk dimension*, implicitly lowering runtime and memory footprint.
4. We provide a systematic empirical study of polynomial sheaf diffusion, including depth sweeps, polynomial-order and spectral-scaling ablations, and synthetic stress tests probing heterophily, scalability, and feature noise. These experiments reveal that moderate polynomial orders are consistently beneficial, that higher-order filters *strictly improve* over first-order NSD at fixed depth, and that PolyNSD maintains strong performance at depth, offering a depth-efficient, spectrally controllable generalisation of Neural Sheaf Diffusion.

2 Related Work

Sheaf Neural Networks, Heterophily and Oversmoothing. GNNs have evolved from early message-passing formulations to a family of architectures that trade off locality, expressivity, and efficiency. Canonical baselines include spectral and spatial convolutions Defferrard et al. [2016], Bruna et al. [2013], Kipf and Welling [2016], attention mechanisms Velickovic et al. [2017], principled aggregates Gilmer et al. [2020] and transformer-based versions Yun et al. [2019], Dwivedi and Bresson [2020]. Despite this progress, two persistent pathologies limit standard GNNs. The former, *Oversmoothing* arises as layers deepen, since repeated low-pass propagation collapses node features toward a near-constant signal, eroding class margins and yielding accuracy drop-offs beyond shallow depth Nt and Maehara [2019], Rusch et al. [2023]. *Heterophily* further stresses isotropic message passing: when adjacent nodes often belong to different classes or carry contrasting attributes, averaging blurs precisely the high-frequency signals that separate classes, and performance degrades as homophily decreases Zhu et al. [2020]. These phenomena are tightly linked in practice and motivate transport-aware architectures that decouple *who* communicates from *how* features are compared.

Sheaf Neural Networks. Cellular sheaf theory Shepard [1985], Curry [2014] equips graphs with local feature spaces (stalks) and linear restriction/transport maps on edges, enabling transport-aware diffusion that can better handle heterophily than isotropic message passing. Sheaf Neural Networks, originally introduced using an hand-crafted sheaf with a single dimensionality in Hansen and Gebhart [2020] and further improved by allowing to learn the sheaf using a learnable parametric function

Bodnar et al. [2022], demonstrated strong performance under heterophily and against oversmoothing, by instantiating diffusion on the sheaf Laplacian. Subsequent works explored attention on sheaves Barbero et al. [2022b], learning the graph connection Laplacian directly from data and at preprocessing time Barbero et al. [2022a], positional encoding based on sheaves He et al. [2023], introducing non-linearities in the process Zaghen [2024], injecting joint diffusion biases Caralt et al. [2024], handling graph heterogeneity Braithwaite et al. [2024], extensions to sheaf hypergraphs Duta et al. [2023], Mule et al. [2025] and applications to recommendation systems Purificato et al. [2023] and federated learning settings Nguyen et al. [2024]. This new research path highlight the modeling benefits of expressing alignment across incidences via learned transports.

Spectral and Polynomial Graph Filters. A large body of work designs filters in the spectrum of the graph Laplacian, from early spectral CNNs and graph wavelets Bruna et al. [2013], Hammond et al. [2011], Shuman et al. [2013] to localized polynomial approximations that avoid explicit eigen-decompositions Defferrard et al. [2016], Kipf and Welling [2016]. Chebyshev-style recurrences and related rational/polynomial filters (e.g., CayleyNets and ARMA layers) offer stable, scalable propagation and interpretable frequency responses Levie et al. [2018], Bianchi et al. [2021]. Further developments include Lanczos-based filterpersonalized PageRank diffusion, and precomputed multi-hop schemes Liao et al. [2019], Klicpera et al. [2019], Wu et al. [2019], Rossi et al. [2020], Chien et al. [2021]. These ideas established that multi-hop context and frequency selectivity can be achieved with sparse matrix–vector primitives instead of decompositions. Our work adopts this philosophy, but lifts it from the graph Laplacian to the *sheaf* Laplacian.

We extend the spectral/polynomial filtering idea from the graph Laplacian to the *sheaf* Laplacian, yielding *Polynomial Neural Sheaf Diffusion (PolyNSD)*: a decomposition-free, recurrence-evaluated filter that preserves sheaves’ transport-aware inductive bias while inheriting the scalability and control of graph-spectral methods. In contrast to prior NSD layers, PolyNSD (i) avoids eigen/SVD steps via stable k -term recurrences, (ii) exposes an explicit K -hop receptive field with tunable low/high/band-pass behavior, and (iii) attains *accuracy per parameter* by using only $K+1$ coefficients per layer and, when desired, *diagonal* restriction maps, decoupling performance from large stalk dimension and reducing runtime/memory.

3 Preliminaries

We briefly overview the necessary background, starting with the graph introduction and cellular sheaf theory and concluding with neural sheaf diffusion. For further discussion on sheaves, please refer to Curry [2014], Rosiak [2022] and to Bodnar et al. [2022] for the full theoretical results of neural sheaf diffusion.

3.1 Cellular Sheaves on Graphs

Definition 1. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a finite undirected graph with an arbitrary orientation on edges. A Cellular Sheaf \mathcal{F} on \mathcal{G} assigns a finite-dimensional inner-product space (the stalk) $\mathcal{F}(v)$ to each vertex $v \in \mathcal{V}$ and $\mathcal{F}(e)$ to each edge $e \in \mathcal{E}$, together with linear restriction/transport maps:

$$\mathcal{F}_{v \trianglelefteq e} : \mathcal{F}(v) \rightarrow \mathcal{F}(e) \quad \forall \text{ incident pair } v \trianglelefteq e \quad (1)$$

The 0- and 1-cochain spaces are then defined as: $C^0(\mathcal{G}; \mathcal{F}) := \bigoplus_{v \in \mathcal{V}} \mathcal{F}(v)$ and $C^1(\mathcal{G}; \mathcal{F}) := \bigoplus_{e \in \mathcal{E}} \mathcal{F}(e)$. Given the chosen orientation (say $e = (u \rightarrow v)$), we can use them to construct the *sheaf coboundary* $\delta : C^0(\mathcal{G}; \mathcal{F}) \rightarrow C^1(\mathcal{G}; \mathcal{F})$ which acts edgewise by:

$$\delta(x)_e = \mathcal{F}_{v \trianglelefteq e} x_v - \mathcal{F}_{u \trianglelefteq e} x_u, \quad x \in C^0(\mathcal{G}; \mathcal{F}) \quad (2)$$

Definition 2. The Sheaf Laplacian is defined as $L_{\mathcal{F}} := \delta^T \delta$, or node-wise as:

$$L_{\mathcal{F}}(x)_u = \sum_{u, v \trianglelefteq e} \mathcal{F}_{u \trianglelefteq e}^\top (\mathcal{F}_{u \trianglelefteq e} x_u - \mathcal{F}_{v \trianglelefteq e} x_v) \quad (3)$$

The sheaf Laplacian is a positive semidefinite block matrix with block diagonals of $L_{\mathcal{F}uu} = \sum_{u \trianglelefteq e} \mathcal{F}_{u \trianglelefteq e}^\top \mathcal{F}_{u \trianglelefteq e}$, and off-diagonal blocks $L_{\mathcal{F}uv} = -\mathcal{F}_{u \trianglelefteq e}^\top \mathcal{F}_{v \trianglelefteq e}$. A normalized Laplacian is obtained by reweighting the sheaf’s inner products (normalizing the stalks) so that the associated

operator, denoted as *normalised* sheaf Laplacian is then defined as $\Delta_{\mathcal{F}} := D^{-1/2} L_{\mathcal{F}} D^{-1/2}$, where D is the block diagonal of $L_{\mathcal{F}}$. This can also be interpreted under an *opinion dynamics* point of view of sheaves on graphs Hansen and Gebhart [2020]. The node stalk $\mathcal{F}(u)$ encodes the node’s *private opinion*, while the restriction map $\mathcal{F}_{u \leq e} : \mathcal{F}(u) \rightarrow \mathcal{F}(e)$ sends it into the edge’s *public discourse* space and the expressed opinion on edge e is $\mathcal{F}_{u \leq e} x_u$. The coboundary δ measures disagreement across incidences, and the (vertex) sheaf Laplacian $L_{\mathcal{F}}$ quantifies the aggregated disagreement at each node.

3.2 Neural Sheaf Diffusion (NSD)

Consider an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each node $u \in \mathcal{V}$ carries a d -dimensional feature vector $x_u \in \mathcal{F}(u)$. Stacking node features gives a column $x \in C^0(\mathcal{G}; \mathcal{F})$, and by allowing up to f feature channels, we can collect them in a matrix $X \in \mathbb{R}^{nd \times f}$, whose j -th column lies in $C^0(\mathcal{G}; \mathcal{F})$. Following Bodnar et al. [2022], the edge-wise restriction/transport maps that define the sheaf at layer (or time) t are computed from incident node features via a learnable function. For $e = (u, v)$, we have:

$$\mathcal{F}_{u \leq e}^{(t)} = \Phi^{(t)}(x_u, x_v), \quad \mathcal{F}_{v \leq e}^{(t)} = \Phi^{(t)}(x_v, x_u) \quad (4)$$

where, typically, $F_{u \leq e}^{(t)} = \Phi^{(t)}(x_u, x_v) = \text{MLP}(x_u || x_v)$ and similarly for $F_{v \leq e}^{(t)}$. Here, these can produce matrices of the appropriate shape (diagonal, bundle, or general). With these transports we build the sheaf Laplacian $\Delta_{\mathcal{F}(t)}$ and perform a sheaf-aware diffusion step on X , which in the discretised approach are:

$$X^{(t+1)} = X^{(t)} - \sigma\left(\Delta_{\mathcal{F}(t)}(I_{nd} \otimes W_1^{(t)}) X^{(t)} W_2^{(t)}\right) \quad (5)$$

where $W_1^{(t)}$ and $W_2^{(t)}$ are trainable weight matrices, I_{nd} is the identity, \otimes denotes the Kronecker product, and $\sigma(\cdot)$ is a nonlinearity. Sheaf Diffusion replaces standard graph diffusion by measuring disagreements *after* transporting node features into each edge’s discourse space, and then aggregates them back to nodes via the sheaf Laplacian.

4 Polynomial Neural Sheaf Diffusion (PolyNSD)

We extend Neural Sheaf Diffusion Bodnar et al. [2022] with *polynomial spectral filters* on sheaf Laplacians, reaching long-range mixing in a single layer while retaining stability and interpretability. In particular, we replace the one-step sheaf diffusion update with a *degree- K operator polynomial* in a (normalised or unnormalised) sheaf Laplacian, evaluated via a stable three-term recurrence on a spectrally rescaled operator. The resulting *Polynomial Neural Sheaf Diffusion* (PolyNSD) layer implements an explicit K -hop receptive field in one layer, with a *trainable spectral response* (low/band/high-pass) obtained as a convex mixture of orthogonal polynomial bases (Chebyshev by default, but the construction is basis-agnostic). To control oversmoothing, we introduce a *high-pass correction* and a *gated residual* that keep the layer nonexpansive while re-injecting high-frequency information. Finally, we show that Chebyshev-PolyNSD with $K=1$ recovers the same operator class as NSD, and strictly generalises it for $K>1$.

Our construction relies on three elementary spectral facts about the (vertex) sheaf Laplacian $L_{\mathcal{F}}$: (i) $L_{\mathcal{F}}$ is real symmetric PSD and thus admits an orthonormal basis of *sheaf Fourier modes* together with a spectral decomposition $L_{\mathcal{F}} = U \Lambda U^{\top}$; (ii) the Dirichlet energy decomposes as $\langle x, L_{\mathcal{F}} x \rangle = \sum_i \lambda_i \hat{x}_i^2$, so small eigenvalues correspond to smooth, globally aligned sheaf signals while large eigenvalues encode oscillatory disagreement; and (iii) for any continuous f , the spectral multiplier $f(L_{\mathcal{F}}) = U f(\Lambda) U^{\top}$ has operator norm $\|f(L_{\mathcal{F}})\|_2 = \max_{\lambda \in \sigma(L_{\mathcal{F}})} |f(\lambda)|$, which we exploit to keep PolyNSD layers nonexpansive.

4.1 Polynomial Filters on Sheaf Laplacians

Let L denote a symmetric positive semidefinite vertex-sheaf Laplacian (either $L_{\mathcal{F}}$ or its normalised version $\Delta_{\mathcal{F}}$, as defined in Equation 3). A *degree- K polynomial filter* in L is then defined as:

$$p(L) = \sum_{k=0}^K c_k L^k, \quad y = p(L) x, \quad (6)$$

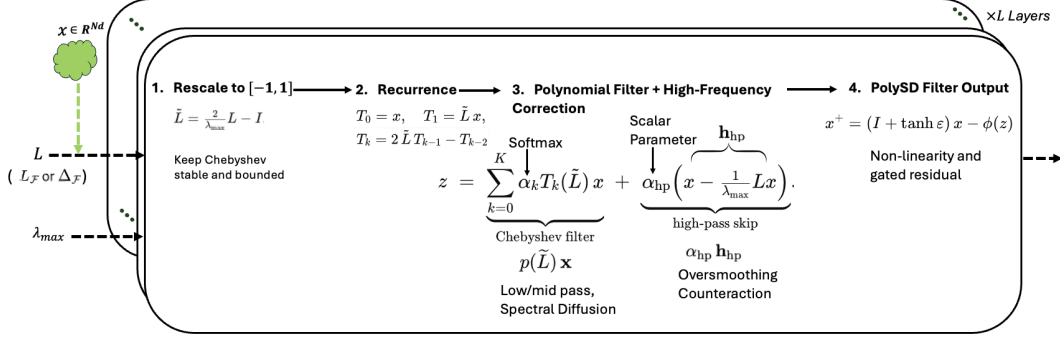


Figure 1: **PolyNSD Layer (degree K)**. The layer first rescales the sheaf Laplacian L to \tilde{L} , evaluates a degree- K Chebyshev polynomial via the three-term recurrence, and adds a learnable high-pass correction and gated residual.

with scalar coefficients $\{c_k\}_{k=0}^K$ shared across nodes and stalk coordinates. Since L is real symmetric PSD, there exists an eigendecomposition $L = U \Lambda U^\top$ with $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_{nd})$ and an orthonormal basis U of *sheaf Fourier modes*. By the polynomial functional calculus, we have:

$$p(L) = U p(\Lambda) U^\top, \quad p(\Lambda) = \text{diag}(p(\lambda_1), \dots, p(\lambda_{nd})). \quad (7)$$

Thus p acts as a *scalar spectral multiplier* $p(\lambda) = \sum_{k=0}^K c_k \lambda^k$ that scales the i -th sheaf Fourier mode by $p(\lambda_i)$, directly controlling the layer’s frequency response (low-/band-/high-pass). In view of the Dirichlet-energy identity $\langle x, Lx \rangle = \sum_i \lambda_i \hat{x}_i^2$, designing $p(\lambda)$ decreasing in λ emphasises smooth, globally aligned sheaf signals, while band-pass or high-pass choices selectively retain or amplify disagreement patterns. This extends the spectral theory of cellular sheaves Hansen and Ghrist [2019] to *learnable* filters and also connects with the opinion-dynamics interpretation: PolyNSD layers implement data-driven operators $p(L)$ that explicitly shape how agreement and disagreement propagate across the sheaf.

K-hop Locality. Spatially, Equation 6 has an explicit K -hop receptive field: multiplying by L propagates information across edges, and L^k involves paths of length up to k .

Proposition 1 (K-hop Locality). *Let L be a sparse block Laplacian on a graph $G = (V, E)$ such that the off-diagonal block (v, u) is nonzero only when $(v, u) \in E$, and diagonal blocks are arbitrary (PSD). Let $p(L)$ be as in Equation 6. Then:*

$$(p(L))_{vu} = 0 \quad \text{whenever} \quad \text{dist}_G(v, u) > K. \quad (8)$$

Thus a single PolyNSD layer achieves K -hop mixing without stacking K message-passing steps.

Commutation and Dirichlet Energy. Polynomial filters in L enjoy two structural properties that make them natural diffusion operators. First, any two such filters share the same eigenbasis and therefore commute: composing $p(L)$ and $q(L)$ is equivalent to applying a single polynomial $(pq)(L)$. In particular, stacking linear PolyNSD layers without nonlinearities or parameter changes does not create new operator types beyond a higher-degree polynomial in L . Second, if the scalar multiplier $p(\lambda)$ is pointwise bounded by $0 \leq p(\lambda) \leq 1$ on the spectrum, then $p(L)$ can only decrease the Dirichlet energy: it damps disagreement modes and never amplifies them.

Proposition 2 (Commutation and non-increasing Dirichlet Energy). *Let L be a symmetric PSD sheaf Laplacian and p, q be real polynomials. Then:*

$$p(L) q(L) = q(L) p(L) = (pq)(L). \quad (9)$$

Moreover, if $L = U\Lambda U^\top$, $x = U\hat{x}$ and $p(\lambda)$ satisfies $0 \leq p(\lambda) \leq 1$ on $\sigma(L)$, then:

$$\langle p(L)x, Lp(L)x \rangle = \sum_{i=1}^{nd} \lambda_i p(\lambda_i)^2 \hat{x}_i^2 \leq \sum_{i=1}^{nd} \lambda_i \hat{x}_i^2 = \langle x, Lx \rangle. \quad (10)$$

Thus diffusion-like filters with $0 \leq p(\lambda) \leq 1$ act as stable, nonexpansive regularisers on the sheaf Dirichlet energy. The commutation identity and the energy bound follow from the spectral theorem and polynomial functional calculus.

Choice of Basis and Chebyshev Parameterisation. Directly learning the monomial coefficients $\{c_k\}$ in Equation 6 is theoretically sound but, at the same time, numerically fragile since the associated Vandermonde systems are exponentially ill-conditioned in K on any nontrivial interval. Instead, we decide to follow classical spectral-approximation practice and expand p in an *orthogonal polynomial basis* on a compact interval. More in concrete, let $\sigma(L) \subset [0, \lambda_{\max}]$ and define the affine rescaling:

$$\tilde{L} = \frac{2}{\lambda_{\max}} L - I, \quad \sigma(\tilde{L}) \subset [-1, 1] \quad (11)$$

Here, λ_{\max} denotes the spectrum of L upper-bound, and can be set in this way: for the degree-normalised sheaf Laplacian $\Delta_{\mathcal{F}}$ we use the canonical bound $\lambda_{\max} = 2$, while for the unnormalised $L_{\mathcal{F}}$ we either adopt a fast Gershgorin-type analytic bound or a short power iteration. A detailed discussion of these choices is given in subsection A.3.

This rescaling is a structural requirement for Chebyshev parameterisations. First-kind Chebyshev polynomials, in particular, satisfy $T_k(\xi) = \cos(k \arccos \xi)$ for $|\xi| \leq 1$, hence $|T_k(\xi)| \leq 1$ on $[-1, 1]$, while for $|\xi| > 1$ they grow exponentially. The affine rescaling in Equation 11 is precisely what ensures that the Chebyshev recurrence operates in the regime where $|T_k(\xi)| \leq 1$, so that convex mixtures of $\{T_k\}$ induce uniformly bounded spectral multipliers and, in turn, stable diffusion operators. More in general, we can then write p as follows:

$$p(\lambda) = \sum_{k=0}^K \theta_k B_k(\xi(\lambda)), \quad \xi(\lambda) = \frac{2}{\lambda_{\max}} \lambda - 1 \quad (12)$$

where $\{B_k\}_{k=0}^K$ is an orthogonal basis (not necessarily a Chebyshev-based one) on $[-1, 1]$, and K indicates the degree of the polynomial. In our default instantiation we choose first-kind Chebyshev polynomials $B_k(\xi) = T_k(\xi)$, and parameterise θ as a *convex mixture* via $\theta = \text{softmax}(\eta)$, $\eta \in \mathbb{R}^{K+1}$. This choice yields: (i) near-minimax uniform approximation on $[-1, 1]$ for many smooth spectral targets, (ii) a numerically stable three-term recurrence, and (iii) an automatic nonexpansive bound $|p(\xi)| \leq 1$ for all $\xi \in [-1, 1]$ when $\theta \in \Delta^K$, which in turn implies $\|p(\tilde{L})\|_2 \leq 1$ and non-increasing Dirichlet energy by Proposition 2.

4.2 PolyNSD Layer and Architecture

This subsection aims to provide a general view of how the PolyNSD block wraps the polynomial filters into a standard sheaf neural network layer. More detailed and further explanations on the single steps could be found in Appendix A. In summary, given a graph $G = (V, E)$ with its associated raw node features x_v^{raw} , an input MLP lifts them to stalk features $x_v \in \mathbb{R}^d$, forming a 0-cochain $x \in C^0(\mathcal{G}; \mathcal{F}) \cong \mathbb{R}^{Nd \times C}$. A sheaf learner Ψ then predicts edge-wise restriction maps $\{\mathcal{F}_{v \leq e}\}$ (diagonal, bundle/orthogonal, or general), which are assembled into a vertex sheaf Laplacian L (unnormalised $L_{\mathcal{F}}$ or degree-normalised $\Delta_{\mathcal{F}}$).

In Figure 1, we can find the part of the architecture that extends the usual sheaf neural networks workflow. From L we obtain an upper spectral bound λ_{\max} (whose value depends on whether we are using $\Delta_{\mathcal{F}}$ or $L_{\mathcal{F}}$), used to form the rescaled operator \tilde{L} with spectrum in $[-1, 1]$. A degree- K Chebyshev-PolyNSD filter $p_\theta(\tilde{L})$ is then evaluated via the three-term Chebyshev recurrence on \tilde{L} , with coefficients θ given by a convex mixture over the orthogonal basis.

High-Pass Skip and Gated Residual. To mitigate the intrinsic low-pass bias of diffusion, we augment the polynomial response with a simple high-pass skip and a gated residual. Given λ_{\max} and L we define:

$$h_{\text{hp}} = x - \lambda_{\max}^{-1} Lx, \quad z = p_\theta(\tilde{L})x + \alpha_{\text{hp}} h_{\text{hp}} \quad (13)$$

with learned scalar $\alpha_{\text{hp}} \in \mathbb{R}$, and apply a 1-Lipschitz nonlinearity ϕ and a diagonal residual gate ε :

$$x^+ = (I + \tanh \varepsilon)x - \phi(z). \quad (14)$$

The resulting linear multiplier has an explicit spectral form, yields a global Lipschitz bound, and avoids collapsing all nonharmonic modes under mild assumptions on p_θ and $\alpha_{\text{hp}} > 0$.

A full PolyNSD architecture stacks one or more of these blocks: each block (1) lifts raw features to stalks, (2) predicts restriction maps and assembles $L_{\mathcal{F}}$ (optionally normalised), (3) applies the Chebyshev-PolyNSD update (14), and (4) maps the final stalk features to task outputs through a linear head. The sparse pattern of L is reused within a block across the polynomial recurrence, and restriction maps can be recomputed at each depth in the nonlinear case.

PolyNSD and NSD Comparison. PolyNSD is a strict generalisation of Neural Sheaf Diffusion: for $K=1$ the update reduces to an operator of the form $aI + bL$ acting on x , recovering the first-order diffusion step of NSD up to normalisation, while $K>1$ yields genuine higher-order polynomials in L with an explicit K -hop receptive field. A degree- K PolyNSD layer therefore requires K sparse-dense multiplications with L (or \tilde{L}), for a cost $O(K \text{ nnz}(L) C)$, matching the asymptotic cost of stacking K NSD layers but without repeated sheaf prediction and Laplacian re-assembly. Full derivations and complexity details are given in subsection A.5.

5 Experimental Evaluation

We empirically evaluate Polynomial Neural Sheaf Diffusion (PolyNSD) layers from section 4 on both real-world and synthetic node-classification benchmarks, with the aim to assess: (i) accuracy on standard sheaf benchmarks across the homophily-heterophily spectrum, (ii) robustness to depth and oversmoothing, (iii) the effect of the polynomial order K and spectral scaling and (iv) the trade-off between accuracy and parameter/compute when replacing NSD with PolyNSD. We summarise the main findings in this section and refer to Appendix B for full details, hyperparameters, and extensive and additional experiments.

Datasets, Splits, and Metrics. We follow the sheaf-learning literature and evaluate on the standard *real-world node-classification benchmarks* Sen et al. [2008], Tang et al. [2009], Namata et al. [2012], Pei et al. [2020], Rozemberczki et al. [2021]: TEXAS, WISCONSIN, FILM, SQUIRREL, CHAMELEON, CORNELL, CITESEER, PUBMED, and CORA, ordered by increasing homophily. We adopt the fixed per-class split protocol used in prior work Bodnar et al. [2022]: 48%/32%/20% train/validation/test, and report test accuracy (mean \pm standard deviation) over 10 fixed splits. For *synthetic experiments*, instead, we use the controlled benchmark of Caralt et al. [2024], which decouples feature complexity from graph connectivity by letting us independently tune: feature noise, heterophily coefficient het , and data scale (N, K, n_c) . We treat these as stress tests for heterophily, data scaling, and covariate noise.

Model Variants and Baselines. We evaluate three PolyNSD variants, corresponding to the three standard sheaf map classes: *DiagPolyNSD* (diagonal restriction maps), *BundlePolyNSD* (orthogonal/bundle transports), and *GeneralPolyNSD* (general symmetric transports). Unless stated otherwise, we use the Chebyshev parameterisation. We compare PolyNSD to multiple families of baselines: *Classical GNNs* like GCN Kipf and Welling [2016], GAT Velickovic et al. [2017], GraphSAGE Hamilton et al. [2017], *Heterophily-oriented GNNs* such as GGCN Yan et al. [2022], Geom-GCN Pei et al. [2020], H2GCN Zhu et al. [2020], GPRGNN Chien et al. [2020], FAGCN Bo et al. [2021], MixHop Abu-El-Haija et al. [2019], *Oversmoothing remedies* including GCNII Chien et al. [2020], PairNorm Chen et al. [2020], and other *Sheaf-based models* inclusive of NSD Bodnar et al. [2022] in its diagonal, orthogonal, and general variants, sheaf-attention models (SAN, ANSD) Barbero et al. [2022b], connection-based sheaves (Conn-NSD) Barbero et al. [2022a], and joint-diffusion sheaf models (RiSNN, JdSNN) Caralt et al. [2024].

Real-World Node Classification under Heterophily. We first compare PolyNSD to the sheaf and non-sheaf baselines across the nine real-world benchmarks, ordered by increasing homophily. Table 1 reports accuracy \pm std on all datasets. We highlight four consistent trends: (i) *PolyNSD matches or improves on state-of-the-art sheaf models*. On the most heterophilous graphs (TEXAS,

Table 1: Accuracy \pm stdev on node classification benchmarks. Datasets are ordered by increasing homophily. Our techniques are denoted *DiagChebySD*, *BundleChebySD* and *GeneralChebySD*. The first section includes Sheaf Neural Networks models, while the second includes other GNN models. The top three models for each dataset are coloured by **First**, **Second** and **Third**, respectively.

	Texas	Wisconsin	Film	Squirrel	Chameleon	Cornell	Citeseer	Pubmed	Cora
<i>Homophily level</i>	0.11	0.21	0.22	0.22	0.23	0.30	0.74	0.80	0.81
#Nodes	183	251	7,600	5,201	2,277	183	3,327	18,717	2,708
#Edges	295	466	26,752	198,493	31,421	280	4,676	44,327	5,278
#Classes	5	5	5	5	5	5	7	3	6
DiagChebySD	88.68\pm3.12	87.84 \pm 4.45	37.14 \pm 1.26	56.61\pm2.06	70.41\pm2.47	86.49\pm5.54	77.74\pm1.26	89.67 \pm 0.34	88.67\pm1.29
BundleChebySD	89.74\pm5.32	87.65 \pm 3.29	37.47 \pm 0.86	54.33 \pm 2.67	69.29\pm1.88	85.40 \pm 7.94	77.57\pm1.55	89.75\pm0.34	88.12 \pm 1.35
GeneralChebySD	88.94\pm4.53	88.23 \pm 4.56	37.20 \pm 0.77	53.88 \pm 1.65	67.34 \pm 2.45	86.49\pm5.80	77.10 \pm 1.30	89.73 \pm 0.41	88.47\pm1.19
RiSNN - NoT	87.89 \pm 4.28	88.04 \pm 2.39	N/A	51.24 \pm 1.71	66.58 \pm 1.81	82.97 \pm 6.17	75.07 \pm 1.56	87.91 \pm 0.55	85.86 \pm 1.31
RiSNN	86.84 \pm 3.72	87.84 \pm 2.60	N/A	53.30 \pm 3.30	65.15 \pm 2.40	85.95 \pm 6.14	76.23 \pm 1.81	88.00 \pm 0.42	85.27 \pm 1.11
JdSNN - NoW	87.30 \pm 4.53	88.43 \pm 2.83	N/A	51.28 \pm 1.80	66.45 \pm 3.46	84.59 \pm 6.95	75.93 \pm 1.41	88.09 \pm 0.49	84.39 \pm 1.47
JdSNN	87.37 \pm 5.10	89.22\pm3.42	N/A	49.89 \pm 1.71	66.40 \pm 2.33	85.41 \pm 4.55	73.27 \pm 1.86	88.19 \pm 0.55	85.43 \pm 1.73
Conn - NSD	86.16 \pm 2.24	88.73 \pm 4.47	37.91 \pm 1.28	45.19 \pm 1.57	65.21 \pm 2.04	85.95 \pm 7.72	75.61 \pm 1.93	89.28 \pm 0.38	83.74 \pm 2.19
SAN	84.05 \pm 5.33	86.47 \pm 3.87	37.09 \pm 1.18	50.96 \pm 1.40	67.46 \pm 1.90	84.32 \pm 5.64	72.57 \pm 1.50	87.12 \pm 0.30	85.90 \pm 1.85
ANSD	85.68 \pm 4.69	87.45 \pm 3.19	37.66 \pm 1.40	54.39 \pm 1.76	68.38 \pm 2.14	84.59 \pm 5.93	76.81 \pm 1.82	89.21 \pm 0.37	87.20 \pm 1.03
Diag - NSD	85.67 \pm 6.95	88.63 \pm 2.75	37.79\pm1.01	54.78 \pm 1.81	68.68 \pm 1.73	86.49\pm7.35	77.14 \pm 1.85	89.42 \pm 0.43	87.14 \pm 1.06
O(d) - NSD	85.95 \pm 5.51	89.41\pm4.74	37.81\pm1.15	56.34\pm1.32	68.04 \pm 1.58	84.86 \pm 4.71	76.70 \pm 1.57	89.49 \pm 0.40	86.90 \pm 1.13
Gen - NSD	82.97 \pm 5.13	89.21\pm3.84	37.80\pm1.22	53.17 \pm 1.51	67.93 \pm 1.58	85.68 \pm 6.51	76.32 \pm 1.65	89.33 \pm 0.35	87.30 \pm 1.15
GGCN	84.86 \pm 4.55	86.86 \pm 3.29	37.54 \pm 1.56	55.17\pm1.58	71.14\pm1.84	85.68 \pm 6.63	77.14 \pm 1.45	89.15 \pm 0.37	87.95 \pm 1.05
H2GCN	84.86 \pm 7.23	87.65 \pm 4.98	35.70 \pm 1.00	36.48 \pm 1.86	60.11 \pm 2.15	82.70 \pm 5.28	77.11 \pm 1.57	89.49 \pm 0.38	87.87 \pm 1.20
GPRGNN	78.38 \pm 4.36	82.94 \pm 4.21	34.63 \pm 1.22	31.61 \pm 1.24	46.58 \pm 1.71	80.27 \pm 8.11	77.13 \pm 1.67	87.54 \pm 0.38	87.95 \pm 1.18
FAGCN	82.43 \pm 6.89	82.94 \pm 7.95	34.87 \pm 1.25	42.59 \pm 0.79	55.22 \pm 3.19	79.19 \pm 9.79	N/A	N/A	N/A
MixHop	77.84 \pm 7.73	75.88 \pm 4.90	32.22 \pm 2.34	48.30 \pm 1.48	60.50 \pm 2.53	73.51 \pm 6.34	76.26 \pm 1.33	85.31 \pm 0.61	87.61 \pm 0.85
GCNII	77.57 \pm 3.83	80.39 \pm 3.40	37.44 \pm 1.30	38.47 \pm 1.58	63.86 \pm 3.04	77.86 \pm 3.79	77.33 \pm 1.48	90.15\pm0.43	88.37\pm1.25
Geom - GCN	66.76 \pm 2.72	64.51 \pm 3.66	31.59 \pm 1.15	38.15 \pm 0.92	60.00 \pm 2.81	60.54 \pm 3.67	78.02\pm1.15	89.95\pm0.47	85.35 \pm 1.57
PairNorm	60.27 \pm 4.34	48.43 \pm 6.14	27.40 \pm 1.24	50.44 \pm 2.04	62.74 \pm 2.82	58.92 \pm 3.15	73.59 \pm 1.47	87.53 \pm 0.44	85.79 \pm 1.01
GraphSAGE	82.43 \pm 6.14	81.18 \pm 5.56	34.23 \pm 0.99	41.61 \pm 0.74	58.73 \pm 1.68	75.95 \pm 5.01	76.04 \pm 1.30	88.45 \pm 0.50	86.90 \pm 1.04
GCN	55.14 \pm 5.16	51.76 \pm 3.06	27.32 \pm 1.10	53.43 \pm 2.01	64.82 \pm 2.24	60.54 \pm 5.30	76.50 \pm 1.36	88.42 \pm 0.50	86.98 \pm 1.27
GAT	52.16 \pm 6.63	49.41 \pm 4.09	27.44 \pm 0.89	40.72 \pm 1.55	60.26 \pm 2.50	61.89 \pm 5.05	76.55 \pm 1.23	87.30 \pm 1.10	86.33 \pm 0.48
MLP	80.81 \pm 4.75	85.29 \pm 3.31	36.53 \pm 0.70	28.77 \pm 1.56	46.21 \pm 2.99	81.89 \pm 6.40	74.02 \pm 1.90	75.69 \pm 2.00	87.16 \pm 0.37

WISCONSIN, FILM, SQUIRREL, CHAMELEON, CORNELL), PolyNSD variants occupy top-3 positions in almost all cases. Adding PolyNSD’s learnable spectral response further improves performance, especially on the most challenging graphs, confirming that controlling the frequency response of sheaf diffusion is useful beyond first-order NSD steps. (ii) *PolyNSD remains competitive on homophilous citation graphs*. On CORA, CITESEER, and PUBMED, PolyNSD performs on par with strong GNN baselines (GCNII, Geom-GCN) and with the best existing sheaf models (ANSD, NSD), often achieving top-3 ranking. This shows that learning higher-order spectral filters on the sheaf Laplacian does not sacrifice performance in benign (homophilous) regimes. (iii) *PolyNSD achieves its best performance with diagonal restriction maps*. In contrast to the prevailing NSD design choice of dense (i.e., bundle/general) per-edge restrictions, we show that simple diagonal maps, when combined with our polynomial spectral filter, are already sufficient to reach state-of-the-art accuracy while reducing parameter count and model complexity. (iv) *PolyNSD attains these results with low stalk dimension*. Strong performance is obtained without inflating the stalk dimension (refer for more details to subsection B.3), indicating that our spectral design is responsible for the gains, and directly yielding lower memory footprint and runtime.

Depth Robustness and Oversmoothing. To study oversmoothing and depth robustness, we sweep the number of layers $L \in \{2, 4, 8, 16, 32\}$ on both homophilous citation graphs (CORA, CITESEER) and heterophilous benchmarks (CORNELL, CHAMELEON). All other hyperparameters are kept fixed, and we mark out-of-memory runs as OOM and numerical instabilities as INS (full results are given in subsection B.4). We summarise here the key trends: (i) *Classical GNNs degrade quickly with depth*. Basic Neural Networks, exhibit the expected sharp performance drop as layers increase, often collapsing to near-random performance beyond 8–16 layers, especially on heterophilous graphs. (ii) *GCNII and sheaf-attention are strong deep baselines*. These models maintain competitive accuracy, confirming that residuals and attention-based sheaf transports mitigate oversmoothing. (iii) *PolyNSD combines depth robustness with high accuracy*. PolyNSD variants are consistently among the strongest models, matching or exceeding competitor models, while remaining stable. On heterophilous datasets, PolyNSD obtains top-3 performance across almost all depths, peaking around

8–16 layers. This supports the view that explicit parallel transport plus polynomial spectral control provides a complementary route to depth robustness.

Effect of Polynomial Order and Spectral Scaling. We next isolate the effect of PolyNSD’s polynomial order K and the choice of spectral bound λ_{\max} . We vary only $K \in \{1, 2, 4, 8, 12, 16\}$ and the λ_{\max} strategy, keeping all other architectural and training hyperparameters fixed (full details are reported in subsection B.5). Concerning the *Polynomial Order K* , across the nine real-world graphs and three PolyNSD variants, we observe that: (i) *Moderate orders ($K \approx 4$ –8) are often optimal on homophilous graphs.* On heterophilous graphs such as SQUIRREL and CHAMELEON, the optimum shifts to larger K ($K \in \{8, 16\}$), consistent with the need for longer-range, multi-frequency propagation. (ii) *Polynomial filters strictly improve over NSD ($K=1$).* For all three transport classes, the best $K > 1$ configuration outperforms the best $K=1$ configuration, meaning that PolyNSD’s higher-order filters capture operator behaviours that NSD cannot express at fixed depth.

PolyNSD vs NSD: Accuracy–Efficiency Trade-offs. Finally, we quantify how much accuracy and parameter efficiency we gain by replacing NSD layers with PolyNSD layers. We focus on three large benchmarks with distinct homophily levels: PUBMED (high homophily), CHAMELEON and SQUIRREL (low homophily). For each dataset and transport class, we: (i) sweep the PolyNSD polynomial degree K at fixed depth and width, (ii) sweep NSD depth L at fixed width, and (iii) sweep NSD width (hidden channels) at fixed depth. We report accuracy and parameter counts in subsection B.6. The *Key Trade-offs* that we found can be summarised as follows: (i) *PolyNSD matches or slightly improves NSD on homophilous graphs with far fewer parameters.* On PUBMED, the best PolyNSD configuration typically improves over the best NSD configuration (always having more layers and more hidden channels) by 0.1%–0.3% test accuracy while using between 1% and 20% of the parameters of the widest NSD models. This reflects PolyNSD’s ability to emulate deep NSD behaviour via spectral polynomials at fixed depth. (ii) *On heterophilous graphs, PolyNSD tends to dominate at comparable or lower parameter counts.* On CHAMELEON, PolyNSD attains gains of up to +6%–+13% accuracy over the best NSD configuration of the same transport class, often at comparable or *smaller* parameter counts. On SQUIRREL, NSD can occasionally match or slightly surpass PolyNSD accuracy by using extremely wide models (two orders of magnitude more parameters), but at realistic parameter budgets PolyNSD remains clearly favourable. (iii) *Depth vs. degree: trading layers for spectral order.* Sweeping NSD depth while holding PolyNSD at a small, fixed depth shows that PolyNSD can achieve comparable or higher accuracy using a single shallow layer with $K > 1$, while NSD requires significantly more layers and parameters to approximate the same effective propagation. This empirically supports the theoretical view that PolyNSD trades depth for spectral expressivity within the same sheaf Laplacian at each layer.

Synthetic Stress Tests We conclude with three controlled synthetic studies designed to probe: (i) heterophily, (ii) data scalability, and (iii) robustness to feature noise. All use the synthetic benchmark of Caralt et al. [2024] and compare PolyNSD to MLP, GCN, VanillaSheaf, and simple sheaf baselines, as well as RiSNN/JdSNN configurations. Plots and detailed settings are reported in subsection B.7. More in particular: (i) *Heterophily Sweeps.* We vary the heterophily coefficient $het \in \{0, 0.25, 0.5, 0.75, 1\}$ across several class counts and two synthetic regimes (RiSNN-style and Diff-style). Classical GNNs degrade rapidly as heterophily increases, often approaching MLP performance when edges become uninformative or adversarial. Sheaf-based models are more robust, and PolyNSD variants consistently occupy the top of the accuracy curves across all het values, confirming that learning spectral responses on sheaf Laplacians is effective even when the graph is strongly heterophilous. (ii) *Data Scalability.* We jointly increase the number of nodes $N \in \{100, 500, 1000\}$ and degree $K \in \{2, 6, 10\}$ at fixed high heterophily. PolyNSD variants maintain near-saturated performance (often $\approx 98\%$) across all scales, while baseline methods improve more slowly or plateau at lower accuracies. These results confirm that PolyNSD scales favourably with data. (iii) *Robustness to Feature Noise.* We inject i.i.d. Gaussian noise into node features while keeping the graph maximally heterophilous and sweep the noise level. Across both synthetic regimes, PolyNSD variants remain among the most robust models: they degrade more slowly, exhibit smaller variance, and retain significant performance even at high noise. This supports the interpretation of PolyNSD as a *structure-aware denoiser*: transport aligns features before comparison, and the spectral polynomial suppresses high-frequency noise while preserving informative modes.

6 Conclusions, Limitations and Future Works

We revisited neural sheaf diffusion through a spectral lens and introduced *Polynomial Neural Sheaf Diffusion* (PolyNSD), a decomposition-free framework that applies orthogonal-polynomial filters to a spectrally rescaled sheaf Laplacian. PolyNSD provides explicit multi-hop receptive fields and tunable spectral responses while reusing the learned transports, avoiding repeated SVD-based normalisation. Empirically, a single PolyNSD architecture attains strong performance on both homophilic and heterophilic benchmarks, often with simple diagonal restriction maps and moderate stalk dimensions, leading to models that are more accurate, cheaper, and easier to optimise than prior Neural Sheaf Diffusion baselines.

Experimentally, we restrict ourselves to static, undirected node-classification benchmarks of moderate size and do not yet evaluate on large-scale graphs, temporal or directed data, or graph/edge-level tasks. Nonetheless, the use of diagonal restriction maps and recurrence-based filters makes each diffusion step linear in the number of edges, so we expect PolyNSD to scale favourably in such regimes. A natural next step is to validate this empirically on truly large graphs, extend PolyNSD to richer settings (e.g., dynamic and hypergraph sheaves, continuous-time Sheaf ODEs), and develop dedicated tools for sheaf-level interpretability that connect the learned transports and spectral profiles to concrete, human-understandable patterns on real graphs.

References

- Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*, pages 21–29. PMLR, 2019.
- 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*, pages 28–36. PMLR, 2022a.
- Federico Barbero, Cristian Bodnar, Haitz Sáez de Ocáriz Borde, and Pietro Lio. Sheaf attention networks. In *NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations*, 2022b.
- Wendong Bi, Lun Du, Qiang Fu, Yanlin Wang, Shi Han, and Dongmei Zhang. Make heterophilic graphs better fit gnn: A graph rewiring approach. *IEEE Transactions on Knowledge and Data Engineering*, 36(12):8744–8757, 2024. doi: 10.1109/TKDE.2024.3441766.
- Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Graph neural networks with convolutional arma filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
- Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 3950–3957, 2021.
- 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.
- Luke Braithwaite, Iulia Duta, and Pietro Liò. Heterogeneous sheaf neural networks. *arXiv preprint arXiv:2409.08036*, 2024.
- Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.
- Ferran Hernandez Caralt, Guillermo Bernárdez Gil, Iulia Duta, Pietro Liò, and Eduard Alarcón Cot. Joint diffusion processes as an inductive bias in sheaf neural networks. *arXiv preprint arXiv:2407.20597*, 2024.

- Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International conference on machine learning*, pages 1725–1735. PMLR, 2020.
- Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. *arXiv preprint arXiv:2006.07988*, 2020.
- Eli Chien, Jianhao Pan, et al. Adaptive universal generalized pagerank graph neural network. In *ICLR*, 2021.
- Justin Michael Curry. *Sheaves, cosheaves and applications*. University of Pennsylvania, 2014.
- 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.
- Iulia Duta, Giulia Cassarà, Fabrizio Silvestri, and Pietro Liò. Sheaf hypergraph networks. *Advances in Neural Information Processing Systems*, 36:12087–12099, 2023.
- Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.
- Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. Pmlr, 2017.
- Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Message passing neural networks. In *Machine learning meets quantum physics*, pages 199–214. Springer, 2020.
- Christoph Goller and Andreas Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of international conference on neural networks (ICNN’96)*, volume 1, pages 347–352. IEEE, 1996.
- Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE international joint conference on neural networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.
- William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.
- David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 2011.
- Jakob Hansen and Thomas Gebhart. Sheaf neural networks. *arXiv preprint arXiv:2012.06333*, 2020.
- Jakob Hansen and Robert Ghrist. Toward a spectral theory of cellular sheaves. *Journal of Applied and Computational Topology*, 3(4):315–358, 2019.
- Yu He, Cristian Bodnar, and Pietro Lio. Sheaf-based positional encodings for graph neural networks. In *NeurIPS 2023 Workshop on Symmetry and Geometry in Neural Representations*, volume 9, 2023.
- Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*, 2019.
- Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. In *ICLR*, 2018.
- Renjie Liao, Marc Brockschmidt, et al. Lanczosnet: Multi-scale deep graph convolutional networks. In *ICLR*, 2019.

- Emanuele Mule, Stefano Fiorini, Antonio Purificato, Federico Siciliano, Stefano Coniglio, and Fabrizio Silvestri. Directional sheaf hypergraph networks: Unifying learning on directed and undirected hypergraphs. *arXiv preprint arXiv:2510.04727*, 2025.
- Galileo Namata, Ben London, Lise Getoor, Bert Huang, and U Edu. Query-driven active surveying for collective classification. In *10th international workshop on mining and learning with graphs*, volume 8, page 1, 2012.
- Bao Nguyen, Lorenzo Sani, Xinchu Qiu, Pietro Liò, and Nicholas D Lane. Sheaf hypernetworks for personalized federated learning. *arXiv preprint arXiv:2405.20882*, 2024.
- Hoang Nt and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.
- Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.
- Antonio Purificato, Giulia Cassarà, Federico Siciliano, Pietro Liò, and Fabrizio Silvestri. Sheaf4rec: Sheaf neural networks for graph-based recommender systems. *ACM Transactions on Recommender Systems*, 2023.
- Daniel Rosiak. *Sheaf theory through examples*. MIT Press, 2022.
- Emanuele Rossi, Fabrizio Frasca, et al. Sign: Scalable inception graph neural networks. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2020.
- Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.
- T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023.
- Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.
- Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.
- Allen Dudley Shepard. *A cellular description of the derived category of a stratified space*. Brown University, 1985.
- David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 2013.
- Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 807–816, 2009.
- Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.
- Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, et al. Simplifying graph convolutional networks. In *ICML*, 2019.
- Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 1287–1292. IEEE, 2022.
- Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. *Advances in neural information processing systems*, 32, 2019.
- Olga Zaghen. Nonlinear sheaf diffusion in graph neural networks. *arXiv preprint arXiv:2403.00337*, 2024.
- Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems*, 33:7793–7804, 2020.

Appendix Summary 13

A PolyNSD Layer 13

A.1 Chebyshev-PolyNSD Layer	13
A.2 Full Polynomial Neural Sheaf Diffusion Architecture	15
A.3 Estimating the Spectral Scale: Analytic Bound and Power Iteration	15
A.4 High-pass Skip and Residual Gating	17
A.5 PolyNSD vs Neural Sheaf Diffusion: Operator Class	18
A.5.1 Chebyshev PolyNSD with $K=1$ Recovers NSD	18
A.5.2 Strict Generalisation for $K>1$	19

B Extensive and Additional Experiments 20

B.1 Datasets	20
B.1.1 Synthetic Benchmarks	21
B.2 Hyper-Parameters	22
B.3 Stalk Dimension vs. Accuracy	23
B.4 Depth Robustness and Oversmoothing	23
B.5 Chebyshev Order K weep	24
B.6 PolyNSD vs NSD: Detailed Accuracy–Efficiency Analysis	26
B.6.1 PolyNSD VS NSD: Depth Sweep	27
B.6.2 PolyNSD VS NSD: Width Sweep	27
B.7 Synthetic Benchmarks: Heterophily, Scalability, and Noise	28
B.7.1 Heterophily Sweeps	29
B.8 Data Scalability	31
B.9 Effect of Feature Noise	31

A PolyNSD Layer

This section makes precise the layer-level construction sketched in subsection 4.2, and provides detailed operator-theoretic and approximation-theoretic guarantees. We first introduce the Chebyshev-PolyNSD block, then describe the full architecture, discuss approximation of diffusion kernels, justify our choices for estimating the spectral scale, and finally analyse the high-pass skip and residual gating.

A.1 Chebyshev-PolyNSD Layer

This subsection expands the Chebyshev-PolyNSD construction sketched in subsection 4.2. Let L be either the unnormalised sheaf Laplacian $L_{\mathcal{F}}$ or its degree-normalised variant $\Delta_{\mathcal{F}}$, and let λ_{\max} be an upper bound on $\sigma(L)$ (equal to 2 in the normalised case, set analytically or estimated by power iteration in the unnormalised case (see subsection A.3)). We first rescale to $[-1, 1]$ as in Equation 11,

$$\tilde{L} = \frac{2}{\lambda_{\max}}L - I, \quad \sigma(\tilde{L}) \subset [-1, 1] \quad (15)$$

Chebyshev Polynomials of the first kind satisfy the following:

$$T_0(\xi) = 1, \quad T_1(\xi) = \xi, \quad T_{k+1}(\xi) = 2\xi T_k(\xi) - T_{k-1}(\xi) \quad (16)$$

which we lift to operators via the three-term recurrence:

$$T_0 = x, \quad T_1 = \tilde{L}x, \quad T_{k+1} = 2\tilde{L}T_k - T_{k-1}, \quad k \geq 1. \quad (17)$$

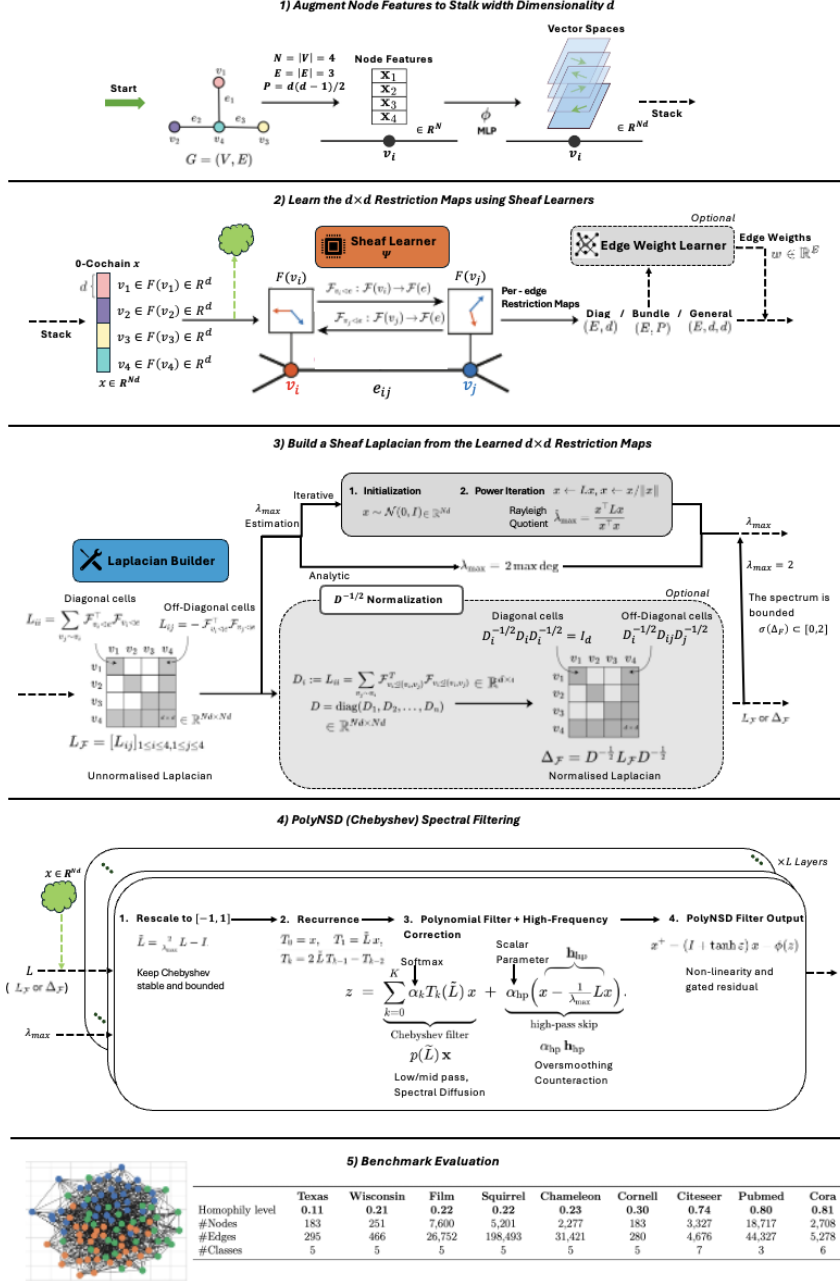


Figure 2: **End-to-end PolyNSD Architecture.** (1) Node features are lifted to stalks. (2) A sheaf learner predicts restriction maps (diagonal, bundle, or general). (3) The vertex sheaf Laplacian $L_{\mathcal{F}}$ (or $\Delta_{\mathcal{F}}$) is assembled and its spectral scale λ_{\max} is obtained analytically or via power iteration. (4) A Chebyshev-PolyNSD block performs polynomial spectral diffusion with high-pass correction and gated residual. (5) A linear head produces task outputs. A more detailed description is given in subsection A.2.

A degree- K Chebyshev-PolyNSD filter with trainable logits $\eta \in \mathbb{R}^{K+1}$ and $\theta = \text{softmax}(\eta) \in \Delta^K$ is then:

$$p_{\theta}(\tilde{L})x = \sum_{k=0}^K \theta_k T_k \quad (18)$$

By construction $|T_k(\xi)| \leq 1$ for all $\xi \in [-1, 1]$ and θ is a convex combination, so $|p_\theta(\xi)| \leq 1$ on $[-1, 1]$. Therefore $\|p_\theta(\tilde{L})\|_2 \leq 1$, and by Proposition 2 and Chebyshev-PolyNSD layers are linear *nonexpansive* maps that cannot increase the Dirichlet energy $\langle x, Lx \rangle$.

Finally, although we never diagonalise L in practice, the functional calculus viewpoint remains useful: for many propagation rules $f(L)$ with scalar response $f(\lambda)$ (e.g. heat kernels $f(\lambda) = e^{-t\lambda}$), the Chebyshev approximation results in show that $p_\theta(\tilde{L})$ with moderate degree K can approximate $f(L)$ exponentially well in operator norm.

A.2 Full Polynomial Neural Sheaf Diffusion Architecture

Figure 2 provides a detailed view of the full PolyNSD architecture, from raw node features to task outputs. We summarise each step, but in subsequent subsections, you can find extensive details for each of them.

(1) Feature Lifting to Stalks. Given a graph $G = (V, E)$ with raw node features $x_v^{\text{raw}} \in \mathbb{R}^{F_{\text{in}}}$, an input MLP $\phi : \mathbb{R}^{F_{\text{in}}} \rightarrow \mathbb{R}^d$ produces stalk features $x_v = \phi(x_v^{\text{raw}}) \in \mathbb{R}^d$. Stacking across nodes yields a 0-cochain $x \in C^0(\mathcal{G}; \mathcal{F}) \cong \mathbb{R}^{Nd \times C}$, where C denotes the number of feature channels.

(2) Sheaf Learner and Restriction Maps. A sheaf learner Ψ takes as input local edge neighbourhoods (and optionally edge attributes) and outputs per-incidence restriction maps $\mathcal{F}_{v \trianglelefteq e} : \mathbb{R}^d \rightarrow \mathbb{R}^d$. We support three families: *diagonal* maps $\mathcal{F}_{v \trianglelefteq e} = \text{diag}(t_{v \trianglelefteq e})$ with $O(|E|d)$ parameters, *bundle/orthogonal* maps $\mathcal{F}_{v \trianglelefteq e} \in O(d)$ acting as parallel transports, and *general* linear maps $\mathcal{F}_{v \trianglelefteq e} \in \text{GL}(d)$ with maximal expressivity. Optional scalar edge weights w_e further modulate the assembled operator.

(3) Laplacian Assembly and Spectral Scale. From $\{\mathcal{F}_{v \trianglelefteq e}\}$ we assemble the vertex sheaf Laplacian $L_{\mathcal{F}} = \delta_{\mathcal{F}}^\top \delta_{\mathcal{F}}$ (and, optionally, its degree-normalised variant $\Delta_{\mathcal{F}}$), stored in sparse (`idx, vals`) format. For the normalised Laplacian we use the standard spectral bound $\lambda_{\max} = 2$, while for the unnormalised case we either use a Gershgorin-type analytic bound or a short power iteration, as detailed in subsection A.3.

(4) PolyNSD + High-pass Skip and Residual Gate. Given L and λ_{\max} , we form \tilde{L} and evaluate a degree- K Chebyshev-PolyNSD filter $p_\theta(\tilde{L})$ via the recurrence (17). To compensate for the intrinsic low-pass bias of diffusion we add the high-pass skip $h_{\text{hp}} = x - \lambda_{\max}^{-1} Lx$, scaled by a learned scalar α_{hp} , and define the pre-nonlinearity $z = p_\theta(\tilde{L})x + \alpha_{\text{hp}} h_{\text{hp}}$. A 1-Lipschitz nonlinearity ϕ and a diagonal residual gate ε then produce the update $x^+ = (I + \tanh \varepsilon)x - \phi(z)$. Its spectral form and global Lipschitz bound are analysed in subsection A.4.

(5) Readout and depth. A linear readout head maps the final stalk features to logits or regression targets. Multiple PolyNSD blocks can be stacked. In practice, we recompute restriction maps at each depth, but keep the within-block recurrence cheap by reusing the same sparse Laplacian and storing only two work buffers (T_{k-1}, T_k) , so the extra memory overhead is $O(NdC)$ and independent of K .

A.3 Estimating the Spectral Scale: Analytic Bound and Power Iteration

Chebyshev rescaling requires an upper spectral bound λ_{\max} for the chosen Laplacian L , so that the affine map $\tilde{L} = \frac{2}{\lambda_{\max}} L - I$ has spectrum in $[-1, 1]$. In PolyNSD we use two complementary strategies to obtain such a bound: (i) *closed-form spectral enclosures* for normalised sheaf Laplacians and for unnormalised Laplacians via Gershgorin-type arguments, and (ii) a *short power iteration* to refine (or replace) the analytic bound on unnormalised operators.

Normalised Sheaf Laplacian. For the degree-normalised sheaf Laplacian $\Delta_{\mathcal{F}} = D^{-1/2} L_{\mathcal{F}} D^{-1/2}$, the situation is directly analogous to the scalar graph case: all eigenvalues lie in $[0, 2]$.

Proposition 3 (Spectral Enclosure for the Normalised Sheaf Laplacian). *Let $\Delta_{\mathcal{F}}$ be the degree-normalised (vertex) sheaf Laplacian on a finite sheaf \mathcal{F} over $G = (V, E)$. Then $\Delta_{\mathcal{F}}$ is symmetric*

positive semidefinite and its spectrum satisfies

$$\sigma(\Delta_{\mathcal{F}}) \subset [0, 2]. \quad (19)$$

Proof. Symmetry and positive semidefiniteness follow from the usual sheaf Laplacian construction: $L_{\mathcal{F}}$ is symmetric PSD as a discrete Hodge Laplacian and $D^{-1/2}$ is symmetric and invertible, so $\Delta_{\mathcal{F}} = D^{-1/2} L_{\mathcal{F}} D^{-1/2}$ is symmetric and PSD as well.

For the upper bound, we use a Rayleigh quotient argument. For any nonzero $x \in C^0(\mathcal{G}; \mathcal{F})$, the Rayleigh quotient of $\Delta_{\mathcal{F}}$ is:

$$R(x) := \frac{\langle x, \Delta_{\mathcal{F}} x \rangle}{\|x\|_2^2} = \frac{\langle D^{-1/2} x, L_{\mathcal{F}} D^{-1/2} x \rangle}{\|x\|_2^2} \quad (20)$$

Writing $y = D^{-1/2} x$ we obtain:

$$R(x) = \frac{\langle y, L_{\mathcal{F}} y \rangle}{\langle D^{1/2} y, D^{1/2} y \rangle} \quad (21)$$

The numerator has the usual sheaf Dirichlet form representation:

$$\langle y, L_{\mathcal{F}} y \rangle = \frac{1}{2} \sum_{(u,v) \in E} \|T_{vu} y_u - y_v\|_{\mathcal{F}_v}^2 \geq 0 \quad (22)$$

while the denominator is a weighted norm $\langle D^{1/2} y, D^{1/2} y \rangle = \sum_v \langle D_v y_v, y_v \rangle$. The same computation as in the scalar normalised Laplacian case shows that:

$$0 \leq R(x) \leq 2 \quad \forall x \neq 0 \quad (23)$$

because the local contributions at each edge are controlled by the incident degrees. Since the eigenvalues of a symmetric matrix are exactly the extremal values of its Rayleigh quotient, this implies $\sigma(\Delta_{\mathcal{F}}) \subset [0, 2]$. \square

In practice, for normalised sheaf Laplacians we simply set $\lambda_{\max} = 2$, which guarantees $\sigma(\tilde{L}) \subset [-1, 1]$ without any numerical estimation.

Unnormalised Sheaf Laplacian: Analytic Bound via Gershgorin Discs. When working with an unnormalised sheaf Laplacian $L = L_{\mathcal{F}} = D - A$, we can obtain a cheap but safe upper bound on the spectral radius by combining Gershgorin's theorem with the structural constraints on D and A .

Proposition 4 (Gershgorin-Type Bound for Unnormalised Sheaf Laplacians). *Let $L_{\mathcal{F}} = D - A$ be an unnormalised sheaf Laplacian, with $D = \text{blkdiag}(D_v) \succeq 0$ block-diagonal over vertices and A supported on off-diagonal edge blocks (so that the diagonal of A is zero). Assume that for each vertex v the diagonal block D_v dominates the outgoing sheaf couplings in the sense that:*

$$\sum_{u \neq v} \|A_{vu}\|_2 \leq \|D_v\|_2 \quad (24)$$

Then $L_{\mathcal{F}}$ is symmetric PSD and its largest eigenvalue satisfies:

$$\lambda_{\max}(L_{\mathcal{F}}) \leq 2 \max_{v \in V} \|D_v\|_2 \quad (25)$$

In the scalar trivial-sheaf case ($\dim \mathcal{F}_v = 1$, $D_v = \deg(v)$) this reduces to the familiar bound $\lambda_{\max} \leq 2 \max_v \deg(v)$.

Proof. Positive semidefiniteness follows from the usual sheaf Laplacian construction (as for $L_{\mathcal{F}}$ above), so all eigenvalues are real and nonnegative.

We now apply Gershgorin's disc theorem to the full matrix $L_{\mathcal{F}}$, viewed as an $N \times N$ real symmetric matrix with $N = \sum_v \dim \mathcal{F}_v$. Let L_{ij} denote its scalar entries, and let λ be any eigenvalue.

Gershgorin's theorem guarantees that there exists a row index i such that:

$$|L_{ii} - \lambda| \leq \sum_{j \neq i} |L_{ij}| \quad (26)$$

The row index i belongs to some vertex fibre \mathcal{F}_v , so the diagonal entry L_{ii} is one of the diagonal entries of the block D_v . Hence $|L_{ii}| \leq \|D_v\|_2$. On the other hand, the off-diagonal entries L_{ij} in that row come from the off-diagonal sheaf blocks $\{-A_{vu}\}_{u \neq v}$, so by the triangle inequality and the definition of the operator norm, we will have:

$$\sum_{j \neq i} |L_{ij}| \leq \sum_{u \neq v} \|A_{vu}\|_2 \leq \|D_v\|_2 \quad (27)$$

by the domination assumption (24). Therefore the corresponding Gershgorin disc for row i is contained in the real interval:

$$[L_{ii} - R_i, L_{ii} + R_i] \subseteq [-\|D_v\|_2, 2\|D_v\|_2], \quad R_i := \sum_{j \neq i} |L_{ij}| \quad (28)$$

Since $L_{\mathcal{F}}$ is symmetric, all its eigenvalues are real and must lie in the union of these intervals over all rows i , and hence over all vertices v . Intersecting with $[0, \infty)$ (because $L_{\mathcal{F}}$ is PSD), we obtain:

$$0 \leq \lambda \leq 2 \max_{v \in V} \|D_v\|_2 \quad (29)$$

for every eigenvalue λ of $L_{\mathcal{F}}$, which proves the claim. In the scalar case, $D_v = \deg(v)$ and the off-diagonal entries in row v are -1 along each incident edge, so $R_i = \deg(v)$ and the usual scalar Gershgorin discs $[0, 2 \deg(v)]$ are recovered. \square

In PolyNSD, the Gershgorin-type bound of Proposition 4 provides a very fast, purely local estimate of λ_{\max} that is guaranteed to be safe for Chebyshev rescaling, even though it can be somewhat conservative.

Unnormalised Sheaf Laplacian: Power Iteration. To obtain a tighter estimate of the largest eigenvalue of $L_{\mathcal{F}}$ we can refine (or replace) the analytic bound via a short power iteration. For a symmetric matrix L with eigenvalues $\lambda_1 \geq \lambda_2 \geq \dots \geq 0$, the classical power method repeatedly applies L and normalises:

$$x^{(t+1)} \leftarrow \frac{Lx^{(t)}}{\|Lx^{(t)}\|_2}, \quad \|x^{(0)}\|_2 = 1, \quad (30)$$

and uses the Rayleigh quotient:

$$\hat{\lambda}^{(t)} := \frac{\langle x^{(t)}, Lx^{(t)} \rangle}{\|x^{(t)}\|_2^2} = \langle x^{(t)}, Lx^{(t)} \rangle \quad (31)$$

as an approximation to λ_1 .

A.4 High-pass Skip and Residual Gating

We conclude the analysis of PolyNSD by formalising the spectral effect and stability properties of the *high-pass correction* and *gated residual* used in the main text, and by explaining why they are structurally useful in the design of our PolyNSD layers. Recall that a single PolyNSD layer applies, before the pointwise nonlinearity ϕ , the linear transformation:

$$h_{\text{hp}} := x - \frac{1}{\lambda_{\max}} Lx \quad (32)$$

$$z := p(\tilde{L})x + \alpha_{\text{hp}} h_{\text{hp}} \quad (33)$$

$$x^+ := (I + \tanh \varepsilon)x - \phi(z) \quad (34)$$

where $\tilde{L} = \frac{2}{\lambda_{\max}} L - I$ is the spectrally rescaled Laplacian, p is a polynomial filter, $\alpha_{\text{hp}} \in \mathbb{R}$ is a learnable scalar, ε is a learnable diagonal parameter (so $I + \tanh \varepsilon$ is a diagonal residual gate), and ϕ is a component-wise nonlinearity with Lipschitz constant $\text{Lip}(\phi) \leq 1$.

The term $p(\tilde{L})x$ plays the role of a learned, spectrally controlled diffusion-like filter, whose response is bounded and well approximated. The additional term $h_{\text{hp}} = x - \frac{1}{\lambda_{\text{max}}}Lx$ is a simple linear spectral correction: it is affine in L , cheap to compute, and has a closed-form frequency response that can be combined with p to compensate for excessive low-pass bias. The residual gate $(I + \tanh \varepsilon)$ then allows us to tune the deviation from the identity map while keeping a global Lipschitz control.

A.5 PolyNSD vs Neural Sheaf Diffusion: Operator Class

In this subsection we make precise the claim that Chebyshev-PolyNSD with degree $K=1$ recovers the same diffusion operator class as Neural Sheaf Diffusion (NSD) Bodnar et al. [2022], while for $K>1$ it strictly generalises it to higher-order polynomials in the sheaf Laplacian. In order to do that, we focus on the *diffusion core*, i.e. the linear operator that acts on sheaf 0-cochains between pointwise nonlinearities and feature-mixing MLPs.

Canonical NSD Diffusion Core. In Neural Sheaf Diffusion, once a sheaf Laplacian L has been assembled from learned restriction maps, the diffusion step acting on a 0-cochain $x \in C^0(\mathcal{G}; \mathcal{F})$ can be written abstractly as:

$$x^+ = Ax - BLx, \quad (35)$$

where A and B are (typically diagonal) feature-wise scaling operators, encoding the step size and residual weighting. For the purposes of spectral analysis, the key point is that, *for fixed L* , NSD’s diffusion core always lies in the two-dimensional linear span $\{I, L\}$: it is a first-order polynomial $aI + bL$, with a, b determined by the learnable scalings in Equation 35 (possibly different per feature channel).

We now show that Chebyshev-PolyNSD with $K=1$ recovers exactly this operator class, and that allowing $K>1$ extends it to higher-order polynomials in L with explicit K -hop locality.

A.5.1 Chebyshev PolyNSD with $K=1$ Recovers NSD

Remember that the layer update used in PolyNSD is:

$$h_{\text{hp}} := x - \frac{1}{\lambda_{\text{max}}}Lx, \quad (36)$$

$$z := p_{\theta}(\tilde{L})x + \alpha_{\text{hp}}h_{\text{hp}}, \quad (37)$$

$$x^+ := (I + \tanh \varepsilon)x - \phi(z), \quad (38)$$

where $p_{\theta}(\tilde{L})$ is a degree- K polynomial in \tilde{L} (e.g. a convex Chebyshev mixture), $\alpha_{\text{hp}} \in \mathbb{R}$ is a scalar high-pass weight, ε is a diagonal residual gate, and ϕ is a 1-Lipschitz nonlinearity. For the operator-class comparison we temporarily set ϕ to the identity and focus on the linear map $x \mapsto x^+$.

We first specialise the PolyNSD diffusion core (38) to a degree-1 Chebyshev parameterisation and compute the resulting linear operator explicitly.

Degree-1 Chebyshev Parameterisation. For $K=1$, the Chebyshev recurrence yields:

$$T_0(\tilde{L}) = I, \quad T_1(\tilde{L}) = \tilde{L} = \frac{2}{\lambda_{\text{max}}}L - I \quad (39)$$

Let $\theta = (\theta_0, \theta_1) \in \Delta^1$ be the (convex) coefficients of the Chebyshev mixture, e.g. $\theta = \text{softmax}(\eta)$ for some logits $\eta \in \mathbb{R}^2$. The degree-1 filter is then:

$$p_{\theta}(\tilde{L}) = \theta_0 T_0(\tilde{L}) + \theta_1 T_1(\tilde{L}) = \theta_0 I + \theta_1 \left(\frac{2}{\lambda_{\text{max}}}L - I \right) \quad (40)$$

Acting on an input x , this gives:

$$x_{\text{cheb}} := p_{\theta}(\tilde{L})x = (\theta_0 - \theta_1)x + \theta_1 \frac{2}{\lambda_{\text{max}}}Lx \quad (41)$$

The high-pass correction is:

$$h_{\text{hp}} = x - \frac{1}{\lambda_{\text{max}}}Lx \quad (42)$$

and the pre-nonlinearity activation becomes:

$$z = x_{\text{cheb}} + \alpha_{\text{hp}}h_{\text{hp}} = (\theta_0 - \theta_1 + \alpha_{\text{hp}})x + \left(\theta_1 \frac{2}{\lambda_{\text{max}}} - \alpha_{\text{hp}} \frac{1}{\lambda_{\text{max}}} \right)Lx \quad (43)$$

Finally, setting ϕ to the identity¹ in (38), we obtain:

$$x^+ = (I + \tanh \varepsilon)x - z \quad (44)$$

Proposition 5 (Chebyshev-PolyNSD with $K=1$ Induces a First-Order Polynomial in L). *Under the setup above with $K=1$ and $\phi = \text{id}$, the PolyNSD update (44) can be written as:*

$$x^+ = ax + bLx \quad (45)$$

where the coefficients a, b (per feature channel) are given by:

$$a = (1 + \tanh \varepsilon) - (\theta_0 - \theta_1 + \alpha_{\text{hp}}), \quad b = -\left(\theta_1 \frac{2}{\lambda_{\text{max}}} - \alpha_{\text{hp}} \frac{1}{\lambda_{\text{max}}}\right) \quad (46)$$

In particular, for any fixed L and λ_{max} , the degree-1 Chebyshev-PolyNSD core spans exactly the same operator class $\{aI + bL\}$ as NSD's diffusion core (35).

Proof. Substituting the expression for z in (43) into (44) yields:

$$x^+ = (I + \tanh \varepsilon)x - \left[(\theta_0 - \theta_1 + \alpha_{\text{hp}})x + \left(\theta_1 \frac{2}{\lambda_{\text{max}}} - \alpha_{\text{hp}} \frac{1}{\lambda_{\text{max}}}\right)Lx\right] \quad (47)$$

$$= \left[(1 + \tanh \varepsilon) - (\theta_0 - \theta_1 + \alpha_{\text{hp}})\right]x - \left[\theta_1 \frac{2}{\lambda_{\text{max}}} - \alpha_{\text{hp}} \frac{1}{\lambda_{\text{max}}}\right]Lx \quad (48)$$

Defining a, b as in (46) gives the claimed form (45). Since $a, b \in \mathbb{R}$ (or feature-wise scalars when ε is diagonal), this coincides with the NSD form (35), i.e. a first-order polynomial $aI + bL$. \square

Exact Matching of a Given NSD Step. Given a target NSD diffusion step:

$$x_{\text{NSD}}^+ = Ax - BLx \quad (49)$$

we can always find PolyNSD parameters $(\theta_0, \theta_1, \alpha_{\text{hp}}, \varepsilon)$ realising the same operator, up to per-channel scaling, provided $\lambda_{\text{max}} > 0$. For example, in the degree-normalised setting where $\lambda_{\text{max}} = 2$ and we ignore the high-pass term ($\alpha_{\text{hp}} = 0$), choosing:

$$\theta_1 = \frac{B}{2}, \quad \theta_0 = 1 - \theta_1 \quad (50)$$

gives $b = -B$ in (46), and then we can enforce $a = A$ by tuning ε , i.e. setting the residual gate so that:

$$1 + \tanh \varepsilon = A + (\theta_0 - \theta_1) \quad (51)$$

Thus, after fixing λ_{max} , the map from PolyNSD parameters to the pair (a, b) is surjective onto the NSD operator class $\{aI + bL\}$.

A.5.2 Strict Generalisation for $K > 1$

We now show that allowing higher polynomial degrees $K > 1$ extends the diffusion operator class from $\{aI + bL\}$ to all polynomials in L of degree at most K . Under mild spectral assumptions on L , this generalises NSD strictly, meaning that there exist PolyNSD operators that cannot be represented by any single NSD diffusion step.

PolyNSD Operator Class for General K . For a degree- K Chebyshev-PolyNSD core, the filter p_θ can be written as:

$$p_\theta(\tilde{L}) = \sum_{k=0}^K \theta_k T_k(\tilde{L}), \quad \theta \in \Delta^K. \quad (52)$$

Each $T_k(\tilde{L})$ is itself a degree- k polynomial in L (since \tilde{L} is affine in L), so $p_\theta(\tilde{L})$ is a polynomial in L of degree at most K . The same calculation as in the $K=1$ case shows that the full linear core:

$$x \mapsto (I + \tanh \varepsilon)x - (p_\theta(\tilde{L})x + \alpha_{\text{hp}}h_{\text{hp}}) \quad (53)$$

remains a polynomial in L of degree at most K . More precisely, we have:

¹In practice ϕ is a 1-Lipschitz nonlinearity; here we isolate the linear diffusion core. The nonlinearity can be placed before or after this core in both NSD and PolyNSD, and does not change the operator class of the linear part.

Depth vs Polynomial Degree and Computational Cost. Let’s now also compare PolyNSD and NSD from a *depth* and *complexity* perspective. For fixed L and linear activations, stacking T NSD diffusion cores of the form $a_t I + b_t L$ yields:

$$x^{(T)} = \left(\prod_{t=1}^T (a_t I + b_t L) \right) x^{(0)} \quad (54)$$

which is a polynomial in L of degree at most T . Conversely, any polynomial in L with real roots can be factorised (over \mathbb{C}) as a product of linear factors, which can in principle be mapped to NSD-style layers. In practice, however, this requires T layers, each with its own sheaf prediction and Laplacian assembly.

By contrast, a single degree- K PolyNSD layer computes $p_\theta(\tilde{L})x$ via a three-term recurrence using K sparse matrix–vector products with \tilde{L} , reusing the same sheaf Laplacian within the block. The resulting complexity is:

$$\mathcal{O}(K \text{nnz}(L) C),$$

for feature dimension C , which matches the asymptotic cost of stacking K NSD layers with *fixed* sheaf structure, *but avoids repeated edge-wise sheaf prediction and Laplacian construction*. This shows that PolyNSD provides a spectrally controlled, computationally efficient generalisation of NSD: degree $K=1$ recovers NSD’s diffusion core, while $K>1$ trades depth for polynomial degree, enlarging the operator class from first-order to higher-order polynomials in the sheaf Laplacian with explicit K -hop locality.

B Extensive and Additional Experiments

This appendix collects the full experimental details for the results reported in section 5. We first describe the datasets used in our evaluation, then provide a consolidated view of model variants, hyperparameters, and training protocols for PolyNSD and all baselines. Here, we provide extended quantitative results that complement the main trends reported in section 5. We first analyze the impact of stalk dimension on PolyNSD performance, then reproduce the full heterophily leaderboard and the complete depth-sweep tables used for the oversmoothing study. We also provide additional ablation studies and detail the synthetic data generation process, and results used for the stress tests in section 5.

B.1 Datasets

In this subsection, we provide further details concerning the real-world and synthetic benchmarks.

Real-World Node-Classification Benchmarks. For real-world experiments we use the standard node-classification benchmarks commonly adopted in the sheaf-learning and heterophily literature: TEXAS, WISCONSIN, FILM, SQUIRREL, CHAMELEON, CORNELL, CITESEER, PUBMED, and CORA. Below we briefly recall their construction.

WebKB (Cornell, Texas, Wisconsin). Nodes correspond to webpages from the computer science departments of the respective universities, and edges are hyperlinks between pages. Node features are bag-of-words representations of page content. The node labels distinguish different page types (e.g., student, project, course, staff, faculty).

- **WikipediaNetwork (Chameleon, Squirrel).** Nodes are Wikipedia pages and edges denote hyperlinks. Node features are derived from page content (e.g., informative nouns or bag-of-words). Labels denote page categories/roles, leading to notoriously heterophilous graphs.
- **Film / Actor.** In the FILM/ACTOR graph, nodes correspond to actors and edges connect actors co-occurring on the same Wikipedia page or film. Node features are keyword-based descriptors; the task is multi-class node classification.
- **Citation graphs (Cora, Citeseer, PubMed).** Nodes represent scientific articles, edges encode citation relations. Node features are bag-of-words or standard pre-computed attributes. The task is to classify each article into a subject category.

Table 2 reports the basic statistics used throughout our experiments: homophily level h (fraction of edges whose endpoints share the same label), number of nodes, number of edges, and number of classes. These match the statistics and fixed-split protocol used in the sheaf literature: for each dataset we adopt the per-class 48%/32%/20% train/validation/test protocol and average results over the same 10 fixed splits as in prior work.

Table 2: Dataset statistics used in the experiments: homophily level h , number of nodes, edges and classes.

Dataset	Homophily h	#Nodes	#Edges	#Classes
Texas	0.11	183	295	5
Wisconsin	0.21	251	466	5
Film	0.22	7,600	26,752	5
Squirrel	0.22	5,201	198,493	5
Chameleon	0.23	2,277	31,421	5
Cornell	0.30	183	280	5
Citeseer	0.74	3,327	4,676	7
Pubmed	0.80	18,717	44,327	3
Cora	0.81	2,708	5,278	6

B.1.1 Synthetic Benchmarks

For the controlled stress tests, we adopt the synthetic benchmark of Caralt et al. [2024], which explicitly decouples (i) feature complexity, (ii) graph connectivity, and (iii) label structure.

Each graph instance is specified by: N (number of nodes), K (even base degree of a regular ring lattice before rewiring), n_c (number of balanced classes), $het \in [0, 1]$ (heterophily coefficient controlling inter-/intra-class edges) and $\sigma \geq 0$ (feature noise level). We briefly recall the feature and graph generation steps below.

Feature Generation: non-linear Class Manifolds with Shared Mean. We construct class-specific, non-linearly separable features while enforcing a common mean across classes. This makes naive averaging ineffective while preserving class structure under non-trivial aggregation. Let n_{data} be the feature dimension, n_h an auxiliary latent dimension, and n_c the number of classes.

1. Draw class prototypes $v_k \sim \mathcal{U}([0, 1]^{n_{\text{data}}})$ for $k = 1, \dots, n_c$.
2. Define a fixed non-linear map $f : \mathbb{R}^{n_h} \rightarrow \mathbb{R}^{n_{\text{data}}}$ as the sine–cosine embedding of the $(n_h - 1)$ -sphere. We sample angles $\theta_0, \dots, \theta_{n_h-3} \sim \mathcal{U}([0, \pi])$ and $\theta_{n_h-2} \sim \mathcal{U}([0, 2\pi])$, construct $s \in \mathbb{S}^{n_h-1}$ via standard hyperspherical coordinates, and set $z = f(\theta)$ after truncating/tiling to length n_{data} .
3. For a node of class k , we sample θ as above and set the raw feature:

$$x_{\text{raw}} = v_k \odot f(\theta) \quad (55)$$

where \odot denotes element-wise multiplication, yielding an ellipsoidal shell per class.

4. We centre features to enforce a shared expected value across classes:

$$\mu = \frac{1}{N} \sum_{i=1}^N x_{\text{raw},i}, \quad \tilde{x}_i = x_{\text{raw},i} - \mu \quad (56)$$

5. Finally, we add i.i.d. Gaussian noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I)$ and optionally rescale:

$$x_i = \tilde{x}_i + \epsilon_i \quad (57)$$

Graph Generation: Ring–rewire with Class-aware Mixing. Connectivity is generated independently of features and controlled by the heterophily index het .

1. Assign classes almost uniformly so that $|C_k| \approx N/n_c$.
2. Build a regular ring lattice of degree K :

$$E = \{(i, j) : 0 < \min(|i-j|, N-|i-j|) \leq K/2\} \quad (58)$$

3. Define the class–transition matrix:

$$R^c = (1 - \text{het}) I_{n_c} + \frac{\text{het}}{n_c - 1} (\mathbf{1}\mathbf{1}^\top - I_{n_c}) \quad (59)$$

so that $\Pr(\text{same class}) = 1 - \text{het}$ and each different class has probability $\text{het}/(n_c - 1)$.

4. For each node i and each of its $K/2$ “rightmost” edges, rewire with probability p :

- (a) Sample a target class $c' \sim \text{Categorical}((R^c)_{c_i,:})$, where c_i is the class of node i .
- (b) Choose j uniformly among nodes in class c' that are not i and not already adjacent to i , and replace the endpoint with j .

We then de-duplicate multi-edges and remove self-loops. The average degree remains K , while the expected fraction of inter-class edges equals het .

We consider two synthetic regimes (RISNN and DIFF) that differ only in feature dimensionality and a few graph knobs, following Caralt et al. [2024].

Table 3: Hyper-parameter Search Space used across Experiments.

Hyper-parameter	Searchable values / notes
Hidden channels	$\{8, 16, 32\}$ (WebKB) and $\{8, 16, 32, 64\}$ (others)
Stalk width d	$\{1, 2, \dots, 5\}$
Layers	$\{1, 2, \dots, 8\}$
Poly degree K	$\{2, 3, 4, 5, 8, 12, 16\}$ (for PolySD / Chebyshev)
Learning rate	0.02 (WebKB) and 0.01 (others)
Activations	ELU
Weight decay (model)	Log-uniform (exponent range e.g. $[-4.5, 11.0]$)
Sheaf decay	Log-uniform (exponent range e.g. $[-4.5, 11.0]$)
Input dropout	categorical $\{0.0, 0.1, \dots, 0.9\}$
Layer dropout	Uniform $[0.0, 0.9]$
Patience (epochs)	100 (Wiki) and 200 (others)
Max training epochs	1000 (Wiki) and 500 (others)
Optimiser	Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$)
Misc model flags	left/right weights, normalised, use_act, edge_weights, etc.

Table 4: Synthetic Experiments: Common Training Hyper-parameters.

Hyper-parameter	Value / note
Models	DiagSheafChebyshev, BundleSheafChebyshev, GeneralSheafChebyshev
Optimiser	Adam (learning rate 0.01)
Weight decay	5×10^{-4}
Sheaf decay	5×10^{-4}
Max epochs / Patience	1500 / 200
Seed	43
Device	single NVIDIA A100–SXM4–80GB
Dataset flag	dataset = synthetic_exp, ellipsoids = false, edge_noise = 0.0

Table 5: Synthetic Experiments: Panel-specific Grids.

Family (Fig)	Column	#Classes	#Feats	#Nodes	Degree	Feat noise	Heterophily (het)
Heterophily	Diff	$\{2, 3, 4, 5\}$	4	$\{400, 600, 800, 1000\}$	4	0.0	$\{0.0, 0.25, 0.50, 0.75, 1.0\}$
Heterophily	RISNN	$\{2, 3, 4, 5\}$	15	$\{400, 600, 800, 1000\}$	4	0.0	$\{0.0, 0.25, 0.50, 0.75, 1.0\}$
Feature Noise	Diff	2	3	400	2	$\{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$	1.0
Feature Noise	RISNN	2	15	500	5	$\{0.0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.30\}$	1.0
Amount of Data	Diff	3	3	$\{100, 500, 1000\}$	$\{2, 6, 10\}$	0.0	0.9
Amount of Data	RISNN	3	15	$\{100, 500, 1000\}$	$\{2, 6, 10\}$	0.0	0.9

B.2 Hyper-Parameters

This section collects the full hyper-parameter search space used in the experiments in section 5. The final reported test-set accuracy for each run is chosen from the checkpoint that achieved the highest validation accuracy. All experiments were run on a single *NVIDIA A100–SXM4–80GB* GPU.

PolyNSD - Real Datasets. Table 3 lists the hyper-parameters and their searchable values / ranges used in the experiments in the Real Datasets settings. These reflects the same choices of Bodnar et al. [2022].

PolyNSD — Synthetic Datasets. For the synthetic study we adopt a *grid* search over three Chebyshev-filtered sheaf diffusion variants, in order to make the plots. Unless otherwise stated, training uses Adam ($\text{lr} = 0.01$), weight decay $= 5 \times 10^{-4}$, sheaf decay $= 5 \times 10^{-4}$, max 1500 epochs with early stopping (patience = 200), and seed = 43. We evaluate three families of settings mirroring the experiments in Caralt et al. [2024]: *Heterophily*, *Feature Noise*, and *Amount of Data*, each family is instantiated in two columns (DIFF on the right and R1SNN on the left), which differ only in feature dimensionality and a few graph knobs. Table 4 summarises the common training setup and Table 5 the panel-specific grids.

B.3 Stalk Dimension vs. Accuracy

In this subsection here we detail the results obtained in section 5 about the stalk dimensionality impact on the accuracy. We begin therefore, this experiment to isolate the effect of stalk dimension on real-world performance. By recalling that the stalk dimension d controls the size of the local sheaf fibres and therefore the dimensionality of the features transported along edges, we have that in prior sheaf models, relatively large stalks, with $d \approx 4$, were often used by default and occasionally increased further in search of improved expressivity.

To quantify how much PolyNSD depends on stalk size, we sweep $d \in \{2, 3, 4, 5\}$ on a representative subset of datasets spanning different homophily regimes: CORA, PUBMED (homophilous citation graphs) and TEXAS, FILM (strongly heterophilous graphs). For each dataset, we run all three PolyNSD variants (Diagonal, Bundle, and General transports). Within each dataset-variant pair, we fix all architectural hyperparameters to the default PolyNSD configuration (depth $L = 2$, hidden width 32, Chebyshev parameterisation, same regularisation and optimiser settings) and vary only the stalk dimension d , training on the standard 10 fixed splits. We then report mean \pm std test accuracy across the 10 runs.

Findings. The results are summarised in Figure 3. On PUBMED and FILM, accuracies are essentially flat within error bars as d varies, indicating that increasing stalk dimension beyond $d = 2$ brings no systematic benefit. On CORA and TEXAS we observe mild, non-monotonic trends: performance can slightly increase when moving from $d = 2$ to $d = 3$, but often degrades again for $d \geq 4$. The strongest configurations on all four datasets typically occur at *small stalk dimensions* ($d \in \{2, 3\}$). Larger stalks do not lead to clear improvements and can even hurt performance through over-parameterisation and increased optimisation difficulty. This contrasts with earlier sheaf architectures, where $d = 4$ emerged as a de facto default and larger stalks were sometimes required to approach state-of-the-art performance.

In PolyNSD, polynomial diffusion on the sheaf Laplacian compensates for the reduced stalk dimensionality: expressive spectral filters and explicit parallel transport allows us to achieve near state-of-the-art accuracy with compact stalks, motivating the choice of low d in the main experiments.

B.4 Depth Robustness and Oversmoothing

In this subsection, we will better investigate how PolyNSD behaves as depth increases, compared to classical GNNs and other sheaf models, providing the full numeric results of the depth sweeps. We consider four representative datasets: CORA, CITESEER (homophilous citation graphs), and CORNELL, CHAMELEON (heterophilous benchmarks). For each model we sweep the number of layers $L \in \{2, 4, 8, 16, 32\}$, keeping all other hyperparameters fixed to the default configuration used in the main experiments for that model family. The sheaf transports (when present), stalk dimension, and hidden width are kept constant across depths. We mark runs that exceed the available GPU memory as OOM (out of memory) and runs that diverge numerically (exploding gradients, NaNs in activations, etc.) as INS (numerical instability). For each model-depth pair, we report test accuracy mean \pm std over the 10 fixed splits, and we additionally highlight, for each model, the depth at which its best test accuracy is achieved. The complete results are given in Table 6. It makes explicit the qualitative patterns summarised in the main text: GCN and Geom-GCN deteriorate quickly with depth (often approaching random performance on heterophilous graphs), GAT runs into numerical instability at high depth, GCNII and SAN/ANS provide strong deep baselines, and PolyNSD variants remain competitive and stable up to $L = 32$ layers, especially on the heterophilous benchmarks.

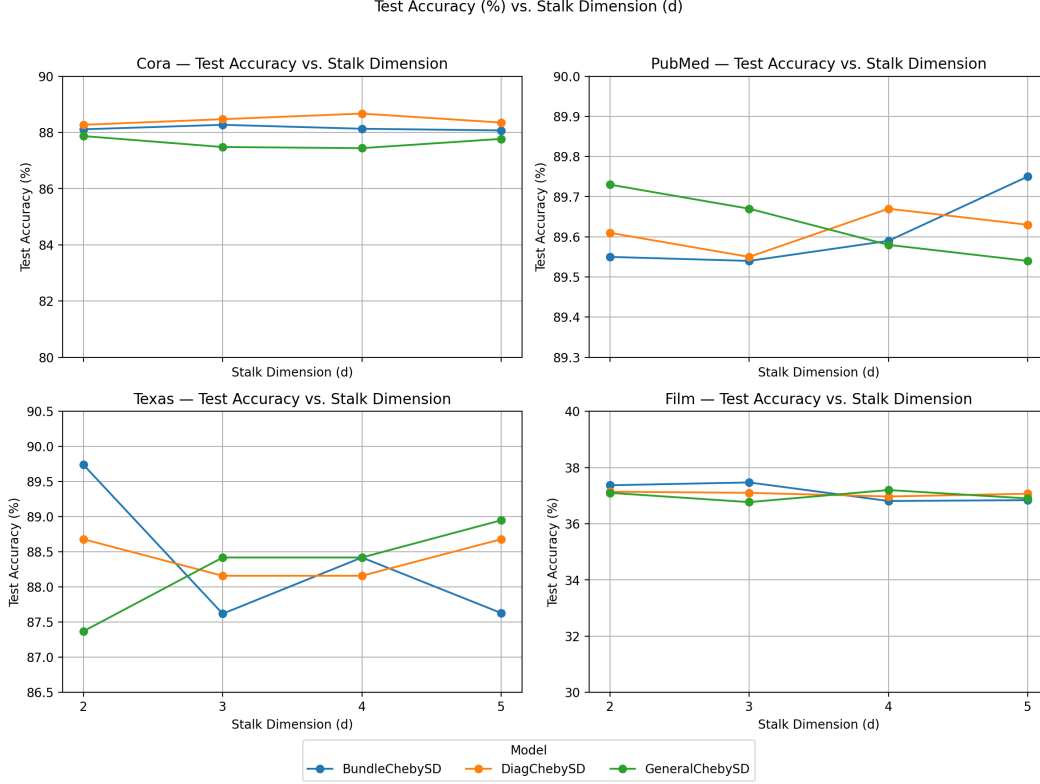


Figure 3: **Test accuracy vs. stalk dimension** $d \in \{2, 3, 4, 5\}$. We sweep d on four real-world datasets (CORA, PUBMED, TEXAS, FILM) for Diagonal, Bundle, and General PolyNSD. All other hyper-parameters are kept fixed to the default configuration used in the main real-world experiments. Error bars show mean \pm std over the 10 fixed data splits.

B.5 Chebyshev Order K weep

This section expands on the polynomial-filter ablations mentioned in section 5. More in particular, we investigate how the polynomial order K of PolyNSD affects performance and how this interacts with the choice of spectral scaling λ_{\max} . The goal is to understand when higher-order filters are beneficial and to what extent they strictly improve over the $K=1$ (NSD-equivalent) case.

For these ablations we use a controlled configuration shared across all datasets: stalk dimension $d = 4$, number of diffusion layers $L = 2$, hidden channels 16, same optimiser, weight decay, and sheaf decay as in the main experiments. The Chebyshev polynomial basis are with the high-pass correction and gated residual structure of subsection 4.2. Within each dataset and PolyNSD variant, we sweep the polynomial order $K \in \{1, 2, 4, 8, 12, 16\}$ and the spectral scaling strategy: *analytic* vs *iterative* estimate of λ_{\max} , as described in subsection 4.1. All runs share the same training splits and are trained until early stopping based on validation accuracy. We report mean \pm std test accuracy over the 10 splits.

Order K . The results in Figure 4 shows test accuracy as a function of K for the nine real-world benchmarks, with one panel per dataset. Each panel overlays the six configurations obtained by crossing the three transport classes (Diagonal, Bundle, General) with the two choices of λ_{\max} estimation (analytic vs iterative). Across datasets, the curves confirm the trends summarised already in section 5: (i) On homophilous graphs such as CORA, CITESEER and PUBMED, moderate orders ($K \approx 4-8$) are often optimal. Increasing K beyond this range yields diminishing returns. (ii) On heterophilous graphs such as SQUIRREL and CHAMELEON, the optimum shifts towards larger values ($K \in \{8, 16\}$), consistent with the need for longer-range, multi-frequency propagation to stabilise learning under label inconsistency. (iii) For all three transport classes and across all datasets, the best

Table 6: Accuracy \pm stdev for real-world datasets used to study the oversmoothing effect. The “best” result corresponds to the number of layers with the highest accuracy. “OOM” denotes out-of-memory and “INS” numerical instability. The top three models for each dataset and layer are coloured by **Red**, **Second** and **Third**, respectively.

Layers	2	4	8	16	32	Best
Cora ($h=0.81$)						
DiagChebySD	88.67\pm1.29	88.23\pm1.36	88.11\pm1.08	87.73\pm1.64	87.22 \pm 1.64	2
BundleChebySD	87.95\pm1.31	87.75 \pm 1.49	87.87\pm1.90	88.01\pm1.46	87.36 \pm 1.49	16
GeneralChebySD	87.44 \pm 1.09	87.77\pm1.24	87.48 \pm 1.25	86.92 \pm 1.18	86.90 \pm 1.48	4
SAN	86.90 \pm 1.31	86.84 \pm 0.97	86.68 \pm 1.13	86.54 \pm 0.89	86.62 \pm 1.39	2
ANSD	86.98 \pm 1.07	87.08 \pm 1.26	86.80 \pm 1.15	86.84 \pm 1.24	86.56\pm0.75	4
GGCN	87.00 \pm 1.15	87.48 \pm 1.32	87.63 \pm 1.33	87.51\pm1.19	87.95\pm1.05	32
GPRGNN	87.93\pm1.11	87.95\pm1.18	87.87\pm1.41	87.26 \pm 1.51	87.18 \pm 1.29	4
H2GCN	87.87 \pm 1.20	86.10 \pm 1.51	86.18 \pm 2.10	OOM	OOM	2
GCNII	85.35 \pm 1.56	85.35 \pm 1.48	86.38 \pm 0.98	87.12 \pm 1.11	87.95\pm1.23	64
PairNorm	85.79 \pm 1.01	85.07 \pm 0.91	84.65 \pm 1.09	82.21 \pm 2.84	60.32 \pm 8.28	2
Geom-GCN	85.35 \pm 1.57	21.01 \pm 2.61	13.98 \pm 1.48	13.98 \pm 1.48	13.98 \pm 1.48	2
GCN	86.98 \pm 1.27	83.24 \pm 1.56	31.03 \pm 3.08	31.05 \pm 2.36	30.76 \pm 3.43	2
GAT	87.30 \pm 1.10	86.50 \pm 1.20	84.97 \pm 1.24	INS	INS	2
Citeseer ($h=0.74$)						
DiagChebySD	77.19\pm1.25	77.12\pm1.61	76.54 \pm 1.93	76.71 \pm 2.50	76.34 \pm 1.24	2
BundleChebySD	76.98 \pm 1.76	77.57\pm1.55	76.97\pm1.90	76.87\pm1.51	77.26\pm1.99	4
GeneralChebySD	77.82\pm1.68	76.75 \pm 1.59	76.62 \pm 1.04	75.87 \pm 1.69	75.11 \pm 1.86	2
SAN	76.27 \pm 1.76	76.30 \pm 1.80	76.62 \pm 1.70	76.18 \pm 1.47	76.07 \pm 2.18	8
ANSD	76.99 \pm 1.74	76.86 \pm 1.71	76.61 \pm 1.51	76.69 \pm 1.56	76.22 \pm 1.47	2
GGCN	76.83 \pm 1.82	76.77 \pm 1.48	76.91\pm1.56	76.88\pm1.56	76.97\pm1.52	10
GPRGNN	77.13 \pm 1.67	77.05\pm1.43	77.09\pm1.62	76.00 \pm 1.64	74.97 \pm 1.47	2
H2GCN	76.90 \pm 1.80	76.09 \pm 1.54	74.10 \pm 1.83	OOM	OOM	1
GCNII	75.42 \pm 1.78	75.29 \pm 1.90	76.00 \pm 1.66	76.96\pm1.38	77.33\pm1.48	32
PairNorm	73.59 \pm 1.47	72.62 \pm 1.97	72.32 \pm 1.58	59.71 \pm 15.97	27.21 \pm 10.95	2
Geom-GCN	78.02\pm1.15	23.01 \pm 1.95	7.23 \pm 0.87	7.23 \pm 0.87	7.23 \pm 0.87	2
GCN	76.50 \pm 1.36	64.33 \pm 8.27	24.18 \pm 1.71	23.07 \pm 2.95	25.3 \pm 1.77	2
GAT	76.55 \pm 1.23	75.33 \pm 1.39	66.57 \pm 5.08	INS	INS	2
Cornell ($h=0.3$)						
DiagChebySD	85.40\pm5.16	85.13\pm5.57	85.68\pm7.36	84.05 \pm 8.41	81.62 \pm 6.92	8
BundleChebySD	85.40\pm7.95	84.86\pm5.82	84.59 \pm 7.05	85.13\pm8.31	84.59\pm7.93	2
GeneralChebySD	85.13\pm6.19	84.86\pm6.42	84.32 \pm 6.49	81.62 \pm 7.53	81.35 \pm 6.22	2
SAN	82.70 \pm 6.64	84.59 \pm 4.69	85.68\pm4.53	84.32\pm6.82	83.51\pm7.20	8
ANSD	84.86 \pm 6.07	84.32 \pm 5.10	84.86 \pm 5.95	84.59\pm6.51	83.24 \pm 3.97	8
GGCN	83.78 \pm 6.73	83.78 \pm 6.16	84.86\pm5.69	83.78 \pm 6.73	83.78\pm6.51	6
GPRGNN	76.76 \pm 8.22	77.57 \pm 7.46	80.27 \pm 8.11	78.38 \pm 6.04	74.59 \pm 7.66	8
H2GCN	81.89 \pm 5.98	82.70 \pm 6.27	80.27 \pm 6.63	OOM	OOM	1
GCNII	67.57 \pm 11.34	64.59 \pm 9.63	73.24 \pm 5.91	77.84 \pm 3.97	75.41 \pm 5.47	16
PairNorm	50.27 \pm 7.17	53.51 \pm 8.00	58.38 \pm 5.01	58.38 \pm 3.01	58.92 \pm 3.15	32
Geom-GCN	60.54 \pm 3.67	23.78 \pm 11.64	12.97 \pm 2.91	12.97 \pm 2.91	12.97 \pm 2.91	2
GCN	60.54 \pm 5.30	59.19 \pm 3.30	58.92 \pm 3.15	58.92 \pm 3.15	58.92 \pm 3.15	2
GAT	61.89 \pm 5.05	58.38 \pm 4.05	58.38 \pm 3.86	INS	INS	2
Chameleon ($h=0.23$)						
DiagChebySD	69.50\pm1.81	70.39\pm1.66	70.04\pm2.87	68.99\pm4.13	66.78 \pm 3.99	4
BundleChebySD	67.13\pm1.63	67.13\pm3.59	66.03 \pm 1.70	66.51 \pm 2.11	66.47 \pm 2.79	2
GeneralChebySD	63.29 \pm 2.47	65.22 \pm 1.72	58.55 \pm 3.43	61.80 \pm 4.64	65.17 \pm 5.19	4
SAN	65.88 \pm 2.10	65.99 \pm 1.28	68.16\pm2.18	68.62\pm2.81	67.61\pm2.80	16
ANSD	65.35 \pm 1.26	67.11\pm1.88	66.69 \pm 2.30	67.39 \pm 1.84	66.91\pm1.61	16
GGCN	70.77\pm1.42	69.58 \pm 2.68	70.33\pm1.70	70.44\pm1.82	70.29\pm1.62	5
GPRGNN	46.58 \pm 1.77	45.72 \pm 3.45	41.16 \pm 5.79	39.58 \pm 7.85	35.42 \pm 8.52	2
H2GCN	59.06 \pm 1.85	60.11 \pm 2.15	OOM	OOM	OOM	4
GCNII	61.07 \pm 4.10	63.86 \pm 3.04	62.89 \pm 1.18	60.20 \pm 2.10	56.97 \pm 1.81	4
PairNorm	62.74 \pm 2.82	59.01 \pm 2.80	54.12 \pm 2.24	46.38 \pm 2.23	46.78 \pm 2.26	2
Geom-GCN	60.00 \pm 2.81	19.17 \pm 1.66	19.58 \pm 1.73	19.58 \pm 1.73	19.58 \pm 1.73	2
GCN	64.82 \pm 2.24	53.11 \pm 4.44	35.15 \pm 3.14	35.39 \pm 3.23	35.20 \pm 3.25	2
GAT	60.26 \pm 2.50	48.71 \pm 2.96	35.09 \pm 3.55	INS	INS	2

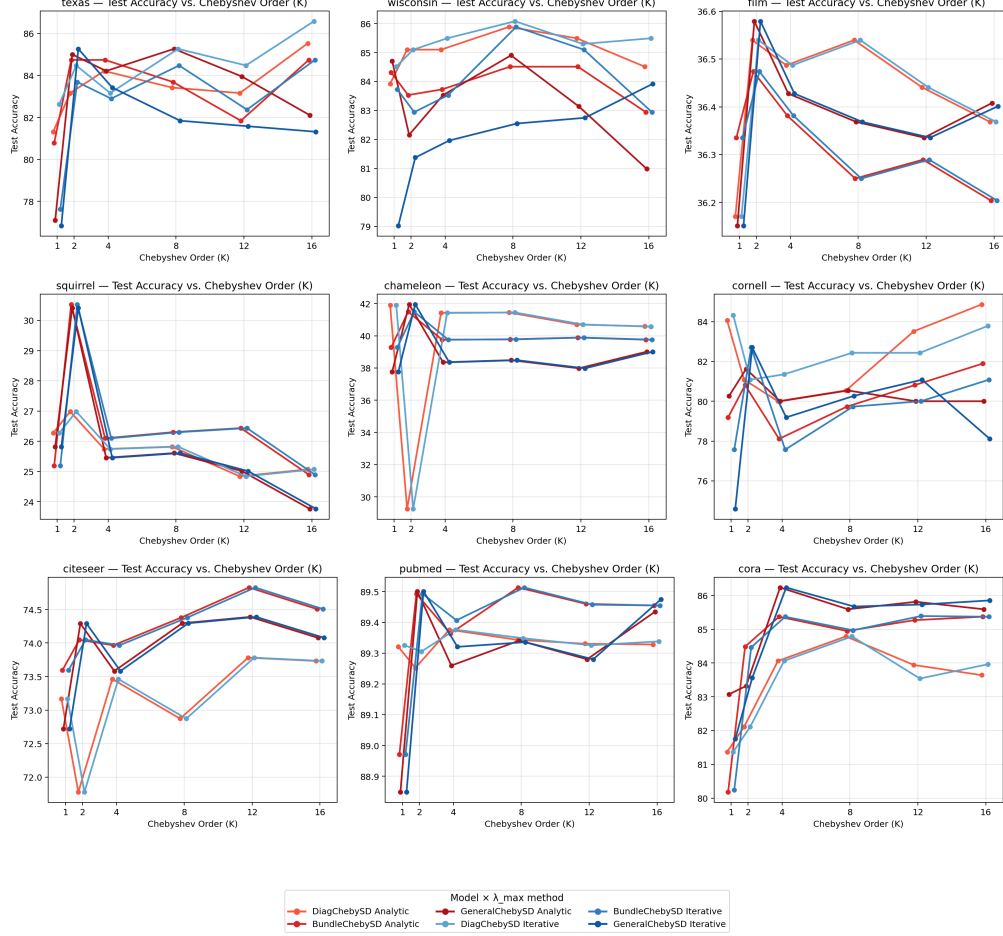


Figure 4: **Chebyshev order K sweep.** Test accuracy vs. K for the nine real-world benchmarks. Each panel corresponds to one dataset and overlays the six configurations given by the three PolyNSD variants crossed with analytic vs. iterative estimates of λ_{\max} . Error bars denote mean \pm std over the 10 fixed splits.

configuration always satisfies $K > 1$, meaning that higher-order polynomial filters strictly improve over the NSD-like $K = 1$ baseline at fixed depth and width.

Numeric Results. To provide a more compact view of the effect of K in the context of the accuracy–efficiency trade-offs, Table 7 reports PolyNSD performance on three datasets with distinct homophily levels: PUBMED (high homophily), CHAMELEON, and SQUIRREL (heterophilous). We fix the depth and width and sweep $K \in \{2, 4, 6, 8, 16\}$, reporting for each entry both the test accuracy and the corresponding parameter count. On all three datasets, the best configuration occurs at $K > 1$, and increasing K allows PolyNSD to emulate a wide range of effective propagation depths without changing the number of layers. Here, higher-order sheaf polynomials are shown to strictly enlarge the class of linear operators that can be realised at fixed depth.

B.6 PolyNSD vs NSD: Detailed Accuracy–Efficiency Analysis

This subsection expands the empirical comparison between PolyNSD and NSD summarised in section 5. Our goal is to quantify how much accuracy can be retained or gained when replacing NSD layers with PolyNSD layers at comparable or reduced parameter counts. We focus on three large benchmarks that cover both homophilous and heterophilous regimes: PUBMED (high homophily), CHAMELEON and SQUIRREL (low homophily). For each dataset and for each transport class

Table 7: **PolySD (Chebyshev) K -sweep.** The three PolySD variants are held fixed with $Layers=2$, $StalkDim=4$ and $Hidden=16$, and columns sweep $K \in \{2, 4, 8, 16\}$ are reported, with the associated number of parameters. Top-1 per dataset and model (per-row) are coloured as **First**.

K	2	4	6	8	16	Best
<i>PubMed</i> ($h=0.80$, $\#N = 18,707$, $\#E = 44,327$, $\#C = 3$)						
DiagChebySD	87.37 \pm 0.45	87.95 \pm 0.60	88.01 \pm 0.39	88.18\pm0.40	87.65 \pm 0.42	8
<i>#params</i>	48,659	48,661	48,663	48,665	48,673	48,665
BundleChebySD	87.86 \pm 0.47	87.91\pm0.32	87.52 \pm 0.45	87.63 \pm 0.37	87.51 \pm 0.74	4
<i>#params</i>	49,619	49,621	49,623	49,625	49,633	49,621
GeneralChebySD	87.74 \pm 0.49	87.80 \pm 0.25	87.62 \pm 0.43	87.87\pm0.49	87.82 \pm 0.49	8
<i>#params</i>	52,499	52,501	52,503	52,505	52,513	52,505
<i>Chameleon</i> ($h=0.23$, $\#N = 2,277$, $\#E = 31,421$, $\#C = 5$)						
DiagChebySD	36.27 \pm 2.48	61.21 \pm 10.09	59.36 \pm 10.55	68.55 \pm 2.32	70.57\pm1.32	16
<i>#params</i>	194,821	194,823	194,825	194,827	194,835	194,835
BundleChebySD	59.85 \pm 7.93	61.97 \pm 7.18	66.18 \pm 2.34	66.64\pm2.35	66.58 \pm 9.92	8
<i>#params</i>	195,781	195,783	195,785	195,787	195,795	195,787
GeneralChebySD	62.39 \pm 3.16	66.05 \pm 1.65	65.94 \pm 1.64	62.08 \pm 3.27	67.39\pm2.50	16
<i>#params</i>	198,661	198,663	198,665	198,667	198,675	198,675
<i>Squirrel</i> ($h=0.22$, $\#N = 5,201$, $\#E = 198,493$, $\#C = 5$)						
DiagChebySD	35.34 \pm 2.76	46.65 \pm 1.76	43.92 \pm 2.83	47.18 \pm 1.18	47.72\pm1.20	16
<i>#params</i>	174,661	174,663	174,665	174,667	174,675	174,675
BundleChebySD	44.12 \pm 1.65	43.58 \pm 2.26	41.69 \pm 5.56	41.29 \pm 2.55	44.48\pm3.22	16
<i>#params</i>	176,901	174,983	176,903	176,905	176,913	176,913
GeneralChebySD	41.66 \pm 1.42	40.27 \pm 2.69	42.02\pm2.50	40.70 \pm 2.01	41.82 \pm 3.24	6
<i>#params</i>	179,781	179,783	179,785	179,787	179,795	179,785

(Diagonal, Bundle, General), we consider three complementary sweeps: (i) *Depth sweep (NSD)*, where hidden width and stalk dimension are fixed, while the number of NSD layers L is varied, and (ii) *Width sweep (NSD)*, where depth and stalk dimension are fixed, while the hidden width (number of channels) is varied. For each configuration we record the test accuracy (mean \pm std over the 10 fixed splits) and the total number of trainable parameters. This allows us to draw iso-accuracy and iso-parameter comparisons between PolyNSD and NSD.

B.6.1 PolyNSD VS NSD: Depth Sweep

In the first comparison we fixed PolyNSD depth to $L = 2$ and vary its polynomial order K , while sweeping NSD depth $L \in \{2, 4, 8, 16, 32\}$ at fixed hidden width 16 and stalk dimension 4. Once chosen the best setting from Table 7, we compare it against the NSD depth sweep. Table 8 summarises the results by reporting, for each dataset and transport class: (i) the best NSD configuration over the depth sweep, (ii) the best PolyNSD configuration over the K sweep at depth $L = 2$ and (iii) the accuracy difference and parameter ratio between PolyNSD and the best NSD configuration.

The table supports the following conclusions: (i) On PUBMED (homophilic setting), PolyNSD matches or slightly improves over the best NSD configuration with comparable or smaller parameter counts. (ii) On CHAMELEON and SQUIRREL (heterophilic settings), PolyNSD yields substantial gains (up to +6%– +13% accuracy) over NSD at similar or smaller parameter budgets, particularly in the diagonal and bundle variants. This highlights the advantage of higher-order spectral control in heterophilous regimes. (iv) If we were to compare PolyNSD and NSD with the same power, we would need to look at the very first column of the layers table. From this, we can notice further improvements: with the same computational power, PolyNSD is extremely stronger than NSD, reaching peaks of +20% in accuracy in heterophilic settings.

B.6.2 PolyNSD VS NSD: Width Sweep

The previous analysis keeps the NSD width fixed while varying depth. We now consider the complementary scenario: we fix NSD depth to $L = 2$ and sweep the hidden width $Hidden \in \{16, 32, 64, 128, 256\}$, keeping stalk dimension 4 constant. Table 9 reports the corresponding test accuracies and parameter counts, together with the relative improvement of PolyNSD over the best NSD configuration across the width sweep.

Table 8: **PolySD vs NSD: Layers Sweep** The three PolySD variants are held fixed with *HiddenDim*=16, *StalkDim*=4 and the sweep over the layers $L \in \{2, 4, 8, 16, 32\}$ is reported, with the associated number of parameters. “OOM” stands for out of memory, whilst “N/A” means that the parameter count is not available. We also report the best results, for each dataset, for the three PolySD variant, with the associated parameter count. The last column contains the summary of improvements of the PolySD model variant w.r.t. the associated best NSD model. Top-1 per dataset and model (per-row) are coloured as **First**. The “improvement” column then shows the PolySD **Improvement** or **Deterioration** w.r.t. the NSD Best setting.

Layers	2	4	8	16	32	Best	PolySD Improvement
<i>PubMed</i> ($h=0.80$, $\#N = 18,707$, $\#E = 44,327$, $\#C = 3$)							
DiagChebySD: 88.18 ± 0.40 (K=8), #params=48,665							
BundleChebySD: 87.91 ± 0.32 (K=4), #params=49,621							
GeneralChebySD: 87.87 ± 0.49 (K=8), #params=52,505							
Diag-NSD	87.82 ± 0.55	87.92 ± 0.51	87.92 ± 0.52	65.92 ± 20.39	39.49 ± 1.60	4	+0.26%
#params	48,655	50,507	54,211	61,619	76,435	50,507	-3.65% (-1,842)
Bundle-NSD	87.70 ± 0.56	87.85 ± 0.42	87.94 ± 0.36	87.63 ± 0.47	37.03 ± 4.72	8	-0.03%
#params	49,615	52,427	58,051	69,299	91,795	58,051	-14.52% (-8,430)
General-NSD	87.48 ± 0.64	87.62 ± 0.36	87.72 ± 0.68	39.94 ± 1.03	39.33 ± 2.25	8	+0.15%
#params	52,495	58,187	69,571	92,339	137,875	69,571	-24.53% (-17,066)
<i>Chameleon</i> ($h=0.23$, $\#N = 2,277$, $\#E = 31,421$, $\#C = 5$)							
DiagChebySD: 70.57 ± 1.32 (K=16), #params=194,835							
BundleChebySD: 66.64 ± 2.35 (K=8), #params=195,787							
GeneralChebySD: 67.39 ± 2.50 (K=16), #params=198,675							
Diag-NSD	64.43 ± 2.06	61.27 ± 5.14	57.34 ± 6.10	22.92 ± 1.42	22.89 ± 2.59	2	+6.15%
#params	194,817	196,669	200,373	207,781	222,597	194,817	+0.009% (+18)
Bundle-NSD	47.10 ± 10.14	54.08 ± 6.30	50.57 ± 3.53	24.91 ± 2.87	23.05 ± 2.46	4	+12.56%
#params	195,777	198,589	204,213	215,461	237,957	198,589	-1.41% (-2,802)
General-NSD	59.60 ± 4.53	58.18 ± 3.56	26.05 ± 3.41	23.79 ± 3.41	20.07 ± 3.74	2	+7.79%
#params	198,657	204,349	215,733	238,501	284,037	198,657	+0% (+0)
<i>Squirrel</i> ($h=0.22$, $\#N = 5,201$, $\#E = 198,493$, $\#C = 5$)							
DiagChebySD: 47.72 ± 1.20 (K=16), #params=174,675							
BundleChebySD: 44.48 ± 3.22 (K=16), #params=176,913							
GeneralChebySD: 42.02 ± 2.50 (K=6), #params=179,785							
Diag-NSD	41.98 ± 1.17	42.60 ± 1.83	42.05 ± 1.54	20.80 ± 1.33	OOM	4	+5.12%
#params	175,937	177,789	181,493	188,901	N/A	177,789	-1.75% (-3,114)
Bundle-NSD	42.46 ± 1.45	42.30 ± 1.83	37.99 ± 2.61	22.86 ± 3.03	OOM	2	+2.02%
#params	176,897	179,709	185,333	196,581	N/A	176,897	+0.009% (+16)
General-NSD	39.11 ± 1.96	39.96 ± 1.76	33.69 ± 1.32	OOM	OOM	4	+2.06%
#params	179,777	185,469	196,853	N/A	N/A	185,469	-3.06% (-5,684)

The trends we report are the following: (i) On PUBMED and CHAMELEON, PolyNSD matches or exceeds the best NSD performance while using fewer parameters dramatically (often only 1%–20% of the parameters of the widest NSD models). Increasing NSD width beyond a certain point yields only marginal gains at a substantial parameter cost. (ii) On SQUIRREL, NSD can sometimes surpass PolyNSD by combining wide layers with large depth, resulting in models with between 10^6 and 10^7 parameters. However, at equal or smaller parameter budgets, PolyNSD consistently attains higher accuracy, indicating a more favourable accuracy–efficiency frontier. Overall, these ablations support the interpretation of PolyNSD as a depth-efficient extension of NSD: by trading layers for spectral order, we can achieve comparable or better performance with shallower networks and significantly reduced parameter counts.

B.7 Synthetic Benchmarks: Heterophily, Scalability, and Noise

We conclude the appendix with the detailed synthetic experiments that underpin the synthetic stress tests discussed in section 5. All experiments use the synthetic data-generation procedure described in subsection B.1.1.

Table 9: **PolySD vs NSD: Hidden-Channels Sweep** The three PolySD variants are held fixed with $Layers=2$, $StalkDim=4$ and the sweep over the columns $HiddenDim \in \{16, 32, 64, 128, 256\}$ is reported, with the associated number of parameters. “OOM” stands for out of memory, whilst “N/A” means that the parameter count is not available. We also report the best results, for each dataset, for the three PolySD variant, with the associated parameter count. The last column contains the summary of improvements of the PolySD model variant w.r.t. the associated best NSD model. Top-1 per dataset and model (per-row) are coloured as **First**. The “improvement” column then shows the PolySD **Improvement** or **Deterioration** w.r.t. the NSD Best setting.

Hidden Channels	16	32	64	128	256	Best	PolySD Improvement
<i>PubMed</i> ($h=0.80$, $\#N = 18,707$, $\#E = 44,327$, $\#C = 3$)							
DiagChebySD: 88.18 ± 0.40 (K=8), #params=48,665							
BundleChebySD: 87.91 ± 0.32 (K=4), #params=49,621							
GeneralChebySD: 87.87 ± 0.49 (K=8), #params=52,505							
Diag-NSD	87.82 ± 0.55	87.86 ± 0.44	88.02 ± 0.53	88.05 ± 0.46	88.03 ± 0.50	128	+0.13%
#params	48,655	111,071	277,375	775,871	2,436,415	775,871	-93.73% (-727,206)
Bundle-NSD	87.70 ± 0.56	87.83 ± 0.51	87.84 ± 0.38	87.87 ± 0.56	87.92 ± 0.44	256	-0.01
#params	49,615	112,991	281,215	783,551	2,451,775	2,451,775	-97.98% (-2,402,154)
General-NSD	87.48 ± 0.64	87.69 ± 0.53	87.71 ± 0.66	87.64 ± 0.55	87.91 ± 0.37	256	-0.04%
#params	52,495	118,751	292,735	806,591	2,497,855	2,497,855	-97.90% (-2,445,350)
<i>Chameleon</i> ($h=0.23$, $\#N = 2,277$, $\#E = 31,421$, $\#C = 5$)							
DiagChebySD: 70.57 ± 1.32 (K=16), #params=194,835							
BundleChebySD: 66.64 ± 2.35 (K=8), #params=195,787							
GeneralChebySD: 67.39 ± 2.50 (K=16), #params=198,675							
Diag-NSD	64.43 ± 2.06	65.33 ± 1.78	65.66 ± 1.61	65.42 ± 1.09	64.78 ± 1.99	64	+4.91%
#params	194,817	403,393	862,017	1,945,153	4,774,977	862,017	-77.40% (-667,182)
Bundle-NSD	46.75 ± 10.70	61.91 ± 8.76	61.53 ± 7.50	63.68 ± 1.60	62.10 ± 2.11	128	+2.96%
#params	195,777	405,313	865,857	1,952,833	4,790,337	1,952,833	-89.97% (-1,757,046)
General-NSD	61.12 ± 2.27	64.28 ± 1.58	64.74 ± 1.46	65.04 ± 2.13	63.92 ± 1.75	128	+2.35%
#params	198,657	411,073	877,377	1,975,873	4,836,417	1,975,873	-89.94% (-1,777,198)
<i>Squirrel</i> ($h=0.22$, $\#N = 5,201$, $\#E = 198,493$, $\#C = 5$)							
DiagChebySD: 47.72 ± 1.20 (K=16), #params=174,675							
BundleChebySD: 44.48 ± 3.22 (K=16), #params=176,913							
GeneralChebySD: 42.02 ± 2.50 (K=6), #params=179,785							
Diag-NSD	41.98 ± 1.17	49.21 ± 1.87	49.35 ± 1.53	49.59 ± 1.10	49.51 ± 1.89	128	-1.87%
#params	175,937	365,633	786,497	1,794,113	4,472,897	1,794,113	-90.26% (-1,619,438)
Bundle-NSD	42.46 ± 1.45	47.48 ± 3.47	49.81 ± 1.47	48.87 ± 2.70	OOM	64	-5.33%
#params	176,897	367,553	790,337	1,801,793	N/A	790,337	-77.62% (-613,424)
General-NSD	39.11 ± 1.96	39.22 ± 1.97	42.58 ± 3.66	OOM	OOM	64	-0.56%
#params	179,777	373,313	801,857	N/A	N/A	801,857	-77.58% (-622,072)

B.7.1 Heterophily Sweeps

To probe performance under controlled heterophily, we vary the heterophily coefficient $het \in \{0, 0.25, 0.5, 0.75, 1.0\}$, while keeping all other graph and feature parameters fixed. We consider both synthetic regimes introduced in Caralt et al. [2024]: (i) RiSNN-style: higher-dimensional features and a configuration mirroring the RiSNN experiments, and (ii) DIFF-style: lower-dimensional features and a configuration mirroring the diffusion-style experiments. For each regime we evaluate multiple class counts and average results over 5 synthetic graph realisations per configuration. We compare PolyNSD variants against MLP, GCN, VanillaSheaf, simple sheaf baselines, and the RiSNN/JdSNN architectures.

Figure 5 summarises the results as a 4×2 grid (rows correspond to different numbers of classes, columns to the two synthetic regimes). Within each subplot, we plot test accuracy vs. het for all models. As heterophily increases, homophily-biased message passing (e.g., GCN) rapidly degrades towards MLP performance, confirming that edges become uninformative or adversarial. Sheaf-based models are more robust, and PolyNSD variants consistently occupy the top of the accuracy curves across all het values, especially in the bundle and general transport classes. This corroborates the real-world findings in section 5 under controlled conditions.

(Synthetic) Heterophily Experiment

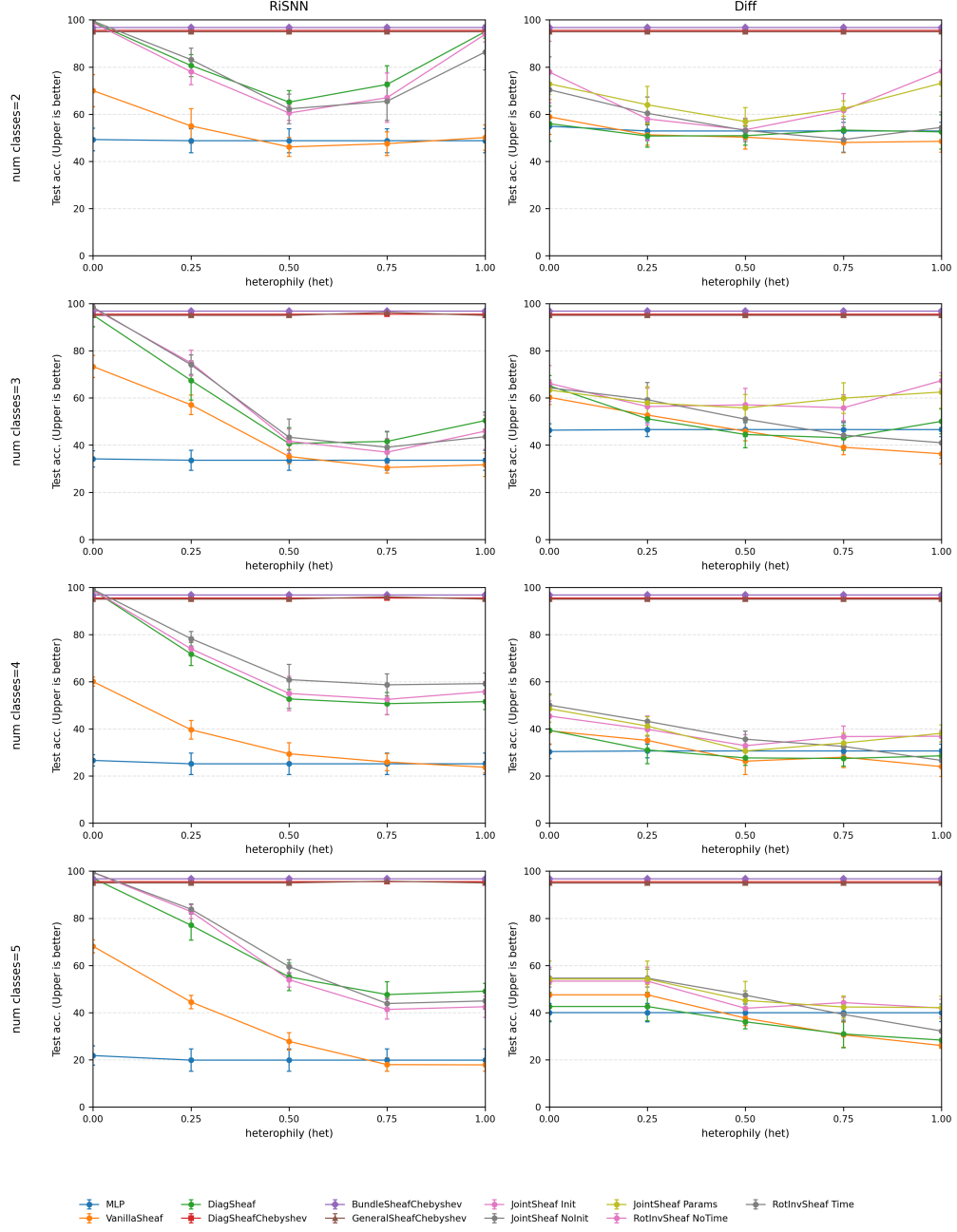


Figure 5: **Synthetic heterophily sweeps.** Each row corresponds to a different number of classes; columns distinguish the RiSNN and DIFF regimes. We sweep $het \in \{0, 0.25, 0.5, 0.75, 1.0\}$; error bars show mean \pm std over multiple random graph realisations.

B.8 Data Scalability

To study how PolyNSD scales with graph size and degree, we jointly vary the number of nodes $N \in \{100, 500, 1000\}$ and the base degree $K \in \{2, 6, 10\}$, while keeping heterophily fixed at a high value ($het = 0.9$). This setting acts as a stress test where edges are mostly cross-class and graphs become denser as K grows. For each (N, K) pair we generate a new synthetic graph in both the R1SNN and DIFF regimes and evaluate all models using the same training protocol as before.

Figure 6 depicts the results as a grid with rows indexed by N and columns by K . Each subplot reports test accuracy for PolyNSD variants and baselines. Across both regimes, PolyNSD variants maintain near-saturated performance (often close to 98%) across all scales, whereas baseline methods improve more slowly or plateau at lower accuracies as N and K increase. These results indicate that PolyNSD scales favourably with both the number of nodes and edge density, and that its polynomial filters remain effective even as graphs become larger and more connected.

B.9 Effect of Feature Noise

Finally, we examine robustness to feature corruption by injecting i.i.d. Gaussian noise into node features while keeping the underlying graphs maximally heterophilous ($het = 1$). This setting isolates the effect of covariate noise from that of connectivity. We consider two noise sweeps: for the DIFF-style setup, we vary `feat_noise` $\in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$, while for the R1SNN-style setup, we use finer-grained noise levels `feat_noise` $\in \{0.00, 0.05, 0.10, 0.15, 0.20, 0.25\}$. For each noise level we generate multiple synthetic instances and average test accuracy over these realisations.

Figure 7 summarises the results. As noise increases, GCN and other homophily-based message-passing models degrade rapidly, eventually approaching the performance of an MLP that ignores the graph. Sheaf-based models are more robust, and PolyNSD variants are consistently among the best-performing methods across all noise levels. The bundle and general transport classes, combined with spectral control, show the strongest robustness, retaining significant accuracy even at the highest noise levels. These findings support the interpretation of PolyNSD as a *structure-aware denoiser*: the sheaf transports align features in local fibres before comparison, while the spectral polynomial can attenuate high-frequency noise modes and preserve informative low- and mid-frequency components.

(Synthetic) Data Scalability Ablation Study

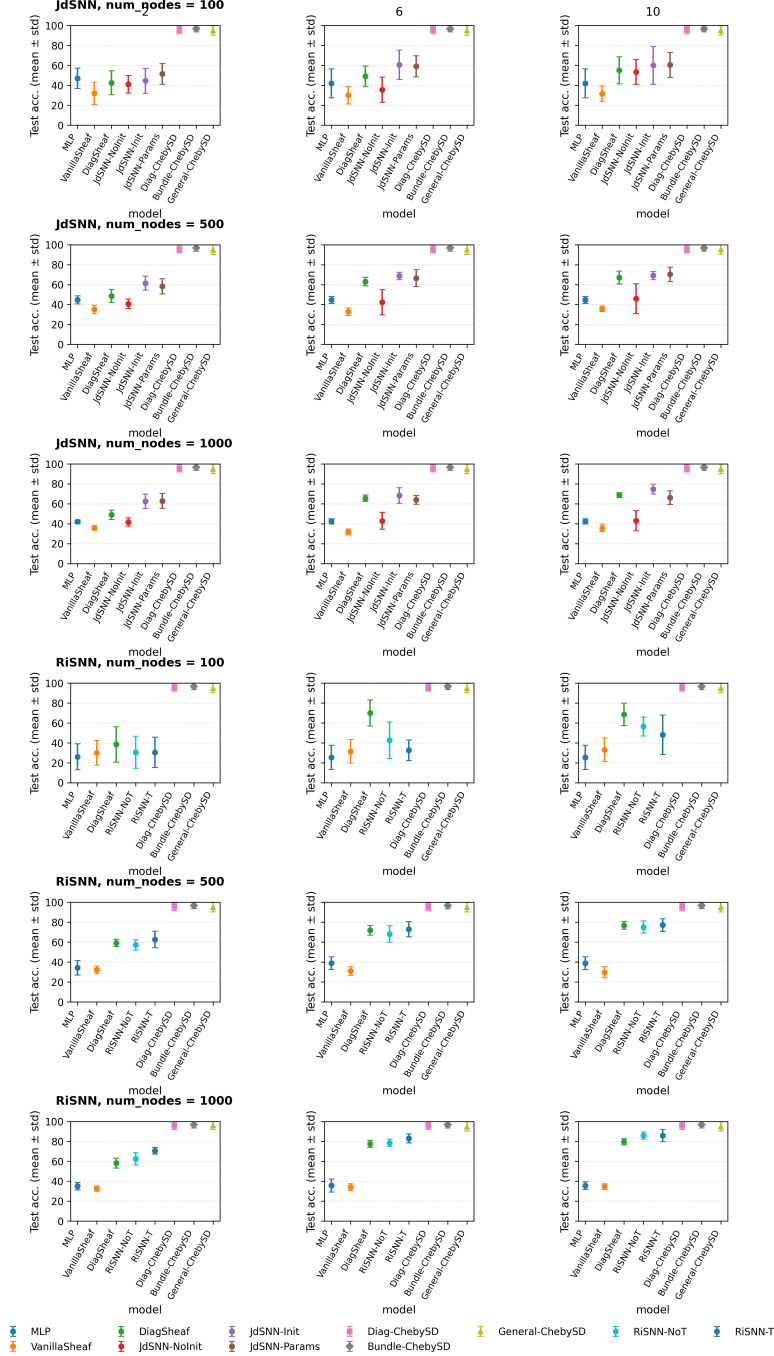


Figure 6: **Data scalability ablation.** Rows correspond to increasing number of nodes $N \in \{100, 500, 1000\}$; columns to degree $K \in \{2, 6, 10\}$ at fixed high heterophily $het = 0.9$. Top block: DIFF setup; bottom block: RISNN setup. PolyNSD maintains near-saturated performance across scales, while baselines plateau at lower accuracies.

(Synthetic) Noise Effect Ablation Study

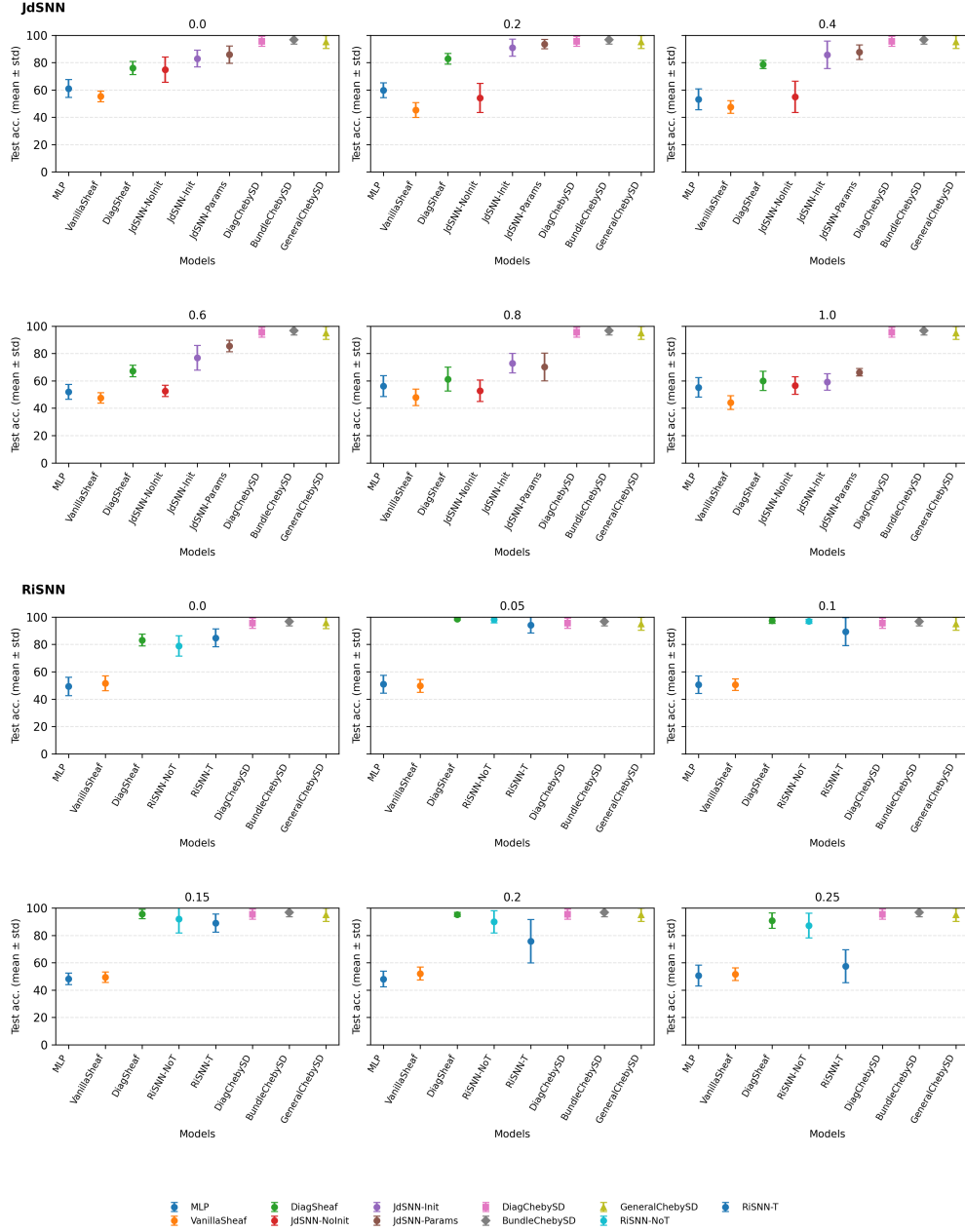


Figure 7: **Effect of feature noise on synthetic tasks.** Top two rows: DIFF-style setup with noise levels $\{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. Bottom two rows: RISNN-style setup with noise levels $\{0.00, 0.05, 0.10, 0.15, 0.20, 0.25\}$. PolyNSD variants remain among the most robust models, degrading more slowly and exhibiting smaller variance than baselines.