Sheaf Diffusion Goes Nonlinear: Enhancing GNNs with Adaptive Sheaf Laplacians

Olga Zaghen [1 2] **Antonio Longa** [3] **Steve Azzolin** [3] **Lev Telyatnikov** [4] **Andrea Passerini** [3] **Pietro Liò** [5]

Editors: *S. Vadgama, E.J. Bekkers, A. Pouplin, S.O. Kaba, H. Lawrence, R. Walters, T. Emerson, H. Kvinge, J.M. Tomczak, S. Jegelka*

Abstract

ZaghenSheaf Neural Networks (SNNs) have recently been introduced to enhance Graph Neural Networks (GNNs) in their capability to learn from graphs. Previous studies either focus on linear sheaf Laplacians or hand-crafted nonlinear sheaf Laplacians. The former are not always expressive enough in modeling complex interactions between nodes, such as antagonistic dynamics and bounded confidence dynamics, while the latter use a fixed nonlinear function that is not adapted to the data at hand. To enhance the capability of SNNs to capture complex node-to-node interactions while adapting to different scenarios, we propose a Nonlinear Sheaf Diffusion (NLSD) model, which incorporates nonlinearity into the Laplacian of SNNs through a general function learned from data. Our model is validated on a synthetic community detection dataset, where it outperforms linear SNNs and common GNN baselines in a node classification task, showcasing its ability to leverage complex network dynamics.[6]

1. Introduction

Graph Neural Networks (GNNs) are powerful tools for analyzing and learning from graph-structured data (Sperduti, 1993; Gori et al., 2005; Scarselli et al., 2008; Micheli, 2009; Bruna et al., 2013; Kipf & Welling, 2016). Recently, Sheaf Neural Networks (SNNs) have been introduced as an enhancement of standard GNNs, improving the expressiveness of node features and information propagation (Hansen & Gebhart, 2020; Bodnar et al., 2022; Barbero et al., 2022; Suk et al., 2022). In essence, sheaves are topological objects that, when used to define SNNs, consist of a lifting map on the input graph that projects node features into a higher-dimensional, more expressive feature space[7]. Sheaf diffusion – which corresponds to the heat diffusion process on sheaves – enables a "inter-dimensional" message passing modelling complex interactions between nodes, and it enhances the linear separation power of node features in the limit of the diffusion process (Bodnar et al., 2022; Duta et al., 2024). Consequently, SNNs can be viewed as higher-dimensional and more general versions of message passing GNNs.

The original definition of SNN models involves using the *linear* sheaf Laplacian (Hansen & Ghrist, 2019; Bodnar et al., 2022), generalizing the graph Laplacian, a linear operator acting on the space of node features. Recently, Hansen & Ghrist (2021) theoretically examined the potential of a *nonlinear* sheaf Laplacian in the context of opinion dynamics. They emphasized its ability to model complex opinion-spreading scenarios, such as bounded confidence and antagonistic dynamics. Additionally, Duta et al. (2024) defined a nonlinear Sheaf (Hypergraph) Neural Network by incorporating a predefined nonlinear function into the Laplacian that allows deriving a sparse graph structure from the original hypergraph data. Previous work, however, has considered only specific and fixed nonlinear functions according to the specific behavior they wanted to model, which requires substantial expert knowledge and lacks flexibility.

In this work, we introduce a *general*, adaptive and data-driven nonlinear function into the sheaf Laplacian to better adapt to the task at hand and model complex interactions between nodes. We propose a general normaliza-

[1]AMLab, University of Amsterdam, Netherlands [2]Work done while at University of Trento and University of Cambridge [3]University of Trento, Italy [4]Sapienza University of Rome, Italy [5]University of Cambridge, United Kingdom. Correspondence to: Olga Zaghen <o.zaghen@uva.nl>.

Proceedings of the Geometry-grounded Representation Learning and Generative Modeling Workshop (GRaM) at the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 251, 2024. Copyright 2024 by the author(s).

[6]Our code is available at `https://github.com/olgatticus/NLSD`.

[7]Sheaves are general topological objects. In this informal definition, we only aim to provide a high-level idea of the application of sheaves in the context of graph learning.

tion scheme for the nonlinear sheaf Laplacian, reflecting the standard symmetric normalization used for the graph Laplacian which is commonly performed to ensure the stability of diffusion dynamics. By incorporating this enhancement, we design a *Nonlinear Sheaf Diffusion* (**NLSD**) model. We present our model as a generalization of previously defined SNNs and experimentally investigate scenarios where NLSD outperforms linear SNNs. Notably, our model demonstrates superior capabilities in a synthetic community detection problem, while experiments on real-world data show that NLSD generally performs on par with linear SNN models.

Finally, in Section 6, we contextualize current SNN models within the emerging Topological Deep Learning (TDL) framework (Hajij et al., 2022; Bodnar, 2022; Papamarkou et al., 2024), which explores methods across various topological domains, including simplicial (Schaub et al., 2022; Yang & Isufi, 2023), cellular (Hajij et al., 2020; Giusti et al., 2023), combinatorial complexes (Hajij et al., 2022), and hypergraphs (Feng et al., 2019; Telyatnikov et al., 2023). We provide a discussion on how sheaves can enhance the diffusion process in the realm of TDL.

Our main contributions can be summarized as follows:

1. We introduce a general and data-dependent nonlinearity into the Laplacian of SNNs, resulting in nonlinear diffusion that enhances message propagation.
2. We propose a novel symmetric normalization scheme for the nonlinear Laplacian as a generalization of the linear case.
3. The resulting NLSD model is shown to outperform the linear case in a synthetic community detection task.

2. Related Work

2.1. Sheaf Neural Networks

The first SNN model was designed by Hansen & Gebhart (2020) to generalize Graph Convolutional Networks (GCNs) (Kipf & Welling, 2016), enabling the modeling of more complex relationships between nodes. In a subsequent work, Bodnar et al. (2022) proposed the Neural Sheaf Diffusion (NSD) model, a more scalable SNN architecture learning the sheaf structure instead of hand-designing it during preprocessing. The design of NSD is also justified by the separation capability of sheaf diffusion: they show, both theoretically and experimentally, that it is well-suited to address common issues in GNN tasks, including oversmoothing and handling heterophilic data (Wu et al., 2022; Rusch et al., 2023; Bodnar et al., 2022).

More recent SNN models primarily build upon the NSD framework and involve introducing attention mechanisms (Barbero et al., 2022), exploring different underlying PDEs for the diffusion process (Suk et al., 2022), and extending SNNs to hypergraph data (Duta et al., 2024).

2.2. Nonlinear sheaf Laplacians

In their sheaf-based model of opinion dynamics, Hansen & Ghrist (2021) demonstrate that the nonlinear Laplacian possesses several key properties that make it more expressive for modeling specific opinion-spreading scenarios, such as bounded confidence and antagonistic dynamics.

In the context of SNNs, a nonlinear Laplacian has been used in the design of SheafHyperGCN (Duta et al., 2024), a SNN model for hypergraph data. In this work, the nonlinearity is defined as an *argmax* function that, for each node v and hyperedge e where $v \in e$, allows for selecting a single neighbor $u \in e$ to define a simple edge (v, u). This operation allows for defining a graph from the initial hypergraph structure, on which sheaf diffusion is then executed.

3. Background

This section aims to provide the necessary context for the introduction of our method. We review the definitions related to sheaves on graphs (for a comprehensive discussion, see Hansen & Ghrist (2019; 2021)) and provide a description of existing SNN models pertinent to our work.

3.1. Cellular sheaves

Sheaves are general mathematical objects that can be defined on any topological space. In what follows, we focus on *cellular sheaves* (Curry, 2014; Shepard, 1985), which are sheaves defined on graphs.

Definition 3.1 (Cellular sheaf)**.** Let $G = (V, E)$ be an undirected graph. A cellular sheaf $(G, \mathcal{F})$ of vector spaces is composed by:

1. An assignment of a vector space $\mathcal{F}(v)$ for each $v \in V$,
2. An assignment of a vector space $\mathcal{F}(e)$ for each $e \in E$,
3. A linear map $\mathcal{F}_{v \trianglelefteq e} : \mathcal{F}(v) \to \mathcal{F}(e)$ whenever v is adjacent to the edge e.

The vector spaces $\mathcal{F}(v)$ and $\mathcal{F}(e)$ are referred to as $stalks$, while the linear maps $\mathcal{F}_{v \trianglelefteq e}$ are the *restriction maps*. For our use case, elements of a vertex stalk $\mathcal{F}(v)$ correspond to node-wise feature vectors $\mathbf{x}_v$, while the edge stalks $\mathcal{F}(e)$ only serve as auxiliary spaces for mixing node features.

Given a sheaf $(G; \mathcal{F})$, the space of 0-cochains $C^0(G; \mathcal{F})$ is the direct sum over the vertex stalks $C^0(G; \mathcal{F}) := \oplus_{v \in V} \mathcal{F}(v)$. The space of 1-cochains $C^1(G; \mathcal{F})$ is instead the direct sum over the edge stalks $C^1(G; \mathcal{F}) :=$

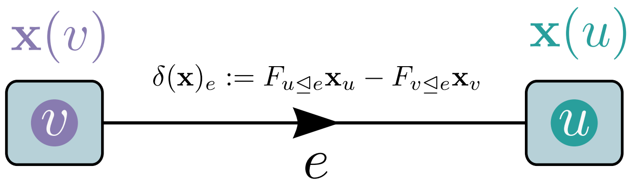
$\mathbf{x}(v)$
$\mathbf{x}(u)$
$\delta(\mathbf{x})_e := F_{u\trianglelefteq e}\mathbf{x}_u - F_{v\trianglelefteq e}\mathbf{x}_v$
v
u
e

Figure 1. The coboundary operator acts on the space of 0-cochains by *projecting* information from node stalks into edge stalks.

$\oplus_{e\in E}\mathcal{F}(e)$. Put in simple terms, these spaces gather all the stalks into a vector space, and specifically the space of 0-cochains consists of all possible collections of feature vectors $\mathbf{x} = (\mathbf{x}_v)_{v\in V}$.

The following definitions lead to the introduction of the *sheaf Laplacian*, which is the focal concept of our study.

Definition 3.2 (Coboundary)**.** Once defined a specific orientation for each edge $e = v \to u \in E$ of the graph G, the coboundary map $\delta : C^0(G;\mathcal{F}) \to C^1(G;\mathcal{F})$ is defined as $\delta(\mathbf{x})_e := F_{u\trianglelefteq e}\mathbf{x}_u - F_{v\trianglelefteq e}\mathbf{x}_v$ (Figure 1).

Remark 3.3. In what follows, when referring to the sheaf Laplacian without specifying "linear" or "nonlinear", we will be referring to the linear one, which is the version commonly studied and used by the GNN community.

Definition 3.4 (Sheaf Laplacian)**.** The sheaf Laplacian is a linear map $L_{\mathcal{F}} : C^0(G;\mathcal{F}) \to C^0(G;\mathcal{F})$ defined as $L_{\mathcal{F}} := \delta^T \circ \delta$. The node-wise expression of the Laplacian is as follows:

$$L_{\mathcal{F}}(\mathbf{x})_v = \sum_{u,v\trianglelefteq e} \mathcal{F}^T_{v\trianglelefteq e}(\mathcal{F}_{v\trianglelefteq e}\mathbf{x}_v - \mathcal{F}_{u\trianglelefteq e}\mathbf{x}_u). \tag{1}$$

The sheaf Laplacian is a positive semidefinite block matrix, of which the diagonal blocks are $L_{\mathcal{F}_{v,v}} = \sum_{v\trianglelefteq e}\mathcal{F}^T_{v\trianglelefteq e}\mathcal{F}_{v\trianglelefteq e}$ while the off-diagonal blocks are $L_{\mathcal{F}_{v,u}} = -\mathcal{F}^T_{v\trianglelefteq e}\mathcal{F}_{u\trianglelefteq e}$.

Definition 3.5 (Normalized sheaf Laplacian)**.** Given a sheaf Laplacian as in 3.4, the corresponding normalized sheaf Laplacian $\Delta_{\mathcal{F}}$ is defined as $\Delta_{\mathcal{F}} = D^{-\frac{1}{2}}L_{\mathcal{F}}D^{-\frac{1}{2}}$ where D is the block-diagonal of $L_{\mathcal{F}}$.

The sheaf Laplacian can be interpreted as a generalization of the well-known graph Laplacian on G: if we define a trivial sheaf where each stalk is isomorphic to $\mathbb{R}$ and the restriction maps are the identity map over $\mathbb{R}$, we recover the standard $n \times n$ graph Laplacian.

For convenience, we set the dimension of all node and edge stalks to a constant value d. Each restriction map will then have dimensions $d \times d$, and the sheaf Laplacian matrix will have dimensions $nd \times nd$. For what concerns node features, along with the dimensionality of the stalks, we can also allow for $f > 1$ channels. In this setting, node features are represented by a single matrix $\mathbf{X} \in \mathbb{R}^{(nd)\times f}$, with columns being vectors in $C^0(G;\mathcal{F})$.

At this point we formally introduce *sheaf diffusion*, that is a heat diffusion process on sheaves.

Definition 3.6 (Sheaf diffusion)**.** Sheaf diffusion is a process on $(G,\mathcal{F})$ governed by the following partial differential equation:

$$\mathbf{X}(0) = \mathbf{X}, \quad \dot{\mathbf{X}}(t) = -\Delta_{\mathcal{F}}\mathbf{X}(t). \tag{2}$$

3.2. Sheaf Neural Networks

In what follows, we provide an overview of the main SNN architectures previously proposed in the literature, highlighting their properties and differences.

Sheaf Convolutional Network. The continuous diffusion process expressed in Equation 2 can be discretized via the explicit Euler scheme with unit step-size:

$$\mathbf{X}(t+1) = \mathbf{X}(t) - \Delta_{\mathcal{F}}\mathbf{X}(t) = (\mathbf{I}_{nd} - \Delta_{\mathcal{F}})\mathbf{X}(t). \tag{3}$$

Starting from Equation 3, Hansen & Gebhart (2020) defined their SNN update function as:

$$\mathbf{X}_{t+1} = \sigma((\mathbf{I}_{nd} - \Delta_{\mathcal{F}})(\mathbf{I}_n \otimes \mathbf{W}_1)\mathbf{X}_t\mathbf{W}_2). \tag{4}$$

The Kronecker product $\otimes$ defines a block matrix $\mathbf{I}_n \otimes \mathbf{W}_1 \in \mathbb{R}^{nd\times nd}$ with $\mathbf{W}_1 \in \mathbb{R}^{d\times d}$ in the diagonal blocks and the zero matrix $\mathbf{0}_n \in \mathbb{R}^{d\times d}$ elsewhere. Assuming $\mathbf{X} \in \mathbb{R}^{nd\times f_1}$, the weight matrix $\mathbf{W}_1$ multiplies independently stalk features of all nodes in all channels, while $\mathbf{W}_2 \in \mathbb{R}^{f_1\times f_2}$ instead alters the number of feature channels in a layer from a quantity f_1 to f_2, as in GCNs. This expression also equips the discretized diffusion equation with a nonlinearity σ.

When $\Delta_{\mathcal{F}}$ is the standard normalized graph Laplacian and $\mathbf{W_1}$ is set to be a scalar, Equation 4 coincides with the GCN model proposed by Kipf & Welling (2016). Since this model can be seen as a generalization of GCN, it is generally referred to as a Sheaf Convolutional Network (SCN).

When implementing the model, many questions arise, such as how to properly define the sheaf itself starting from graph-structured input data. In the SCN model, Hansen & Gebhart (2020) use a *hand-crafted* sheaf with $d = 1$, constructed with the assumption of fully understanding the data-generating process in a synthetic setting. Bodnar et al. (2022), on the other hand, design a more versatile approach that allows the sheaf to be learned end-to-end directly from the graph data, enabling the selection of the appropriate geometry for solving the specific task and making the model applicable to any real-world graph dataset. We now proceed to define this SNN model.

Neural Sheaf Diffusion. Bodnar et al. (2022) directly enrich Equation 2 with the two weight matrices $\mathbf{W}_1$ and $\mathbf{W}_2$

and the nonlinear function σ:

$$\dot{\mathbf{X}}(t) = -\sigma(\Delta_{\mathcal{F}(t)}(\mathbf{I}_n \otimes \mathbf{W}_1)\mathbf{X}(t)\mathbf{W}_2), \tag{5}$$

that gives rise to the following layer-wise update after performing Euler discretization in time:

$$\mathbf{X}_{t+1} = \mathbf{X}_t - \sigma(\Delta_{\mathcal{F}(t)}(\mathbf{I}_n \otimes \mathbf{W}_1^t)\mathbf{X}_t\mathbf{W}_2^t). \tag{6}$$

In these Equations, $\Delta_{\mathcal{F}(t)}$ is the Laplacian of a sheaf that evolves over time. In practice, Equation 6 is further enriched in expressiveness by learning an additional parameter $\epsilon \in [-1, 1]^d$ that allows the model to adjust the relative magnitude of the features in each stalk dimension:

$$\mathbf{X}_{t+1} = (1+\boldsymbol{\varepsilon})\,\mathbf{X}_t - \sigma\left(\Delta_{\mathcal{F}(t)}\left(\mathbf{I}_n \otimes \mathbf{W}_1^t\right)\mathbf{X}_t\mathbf{W}_2^t\right), \tag{7}$$

where $\boldsymbol{\varepsilon} \in [-1, 1]^{nd}$ is obtained by concatenating ϵ n times. Additionally to using a Multi-Layer Perceptron (MLP) to compute $\mathbf{X}(0)$ from the raw node features of the input graph, Bodnar et al. (2022) also came up with a method to perform sheaf learning *locally*. This is done by parametrizing the $d \times d$ matrices of restriction maps $\mathcal{F}_{v \trianglelefteq e}$ through a parametric matrix-valued function $\Psi : \mathbb{R}^{d\times 2} \to \mathbb{R}^{d\times d}$, $\mathcal{F}_{v \trianglelefteq e:=(v,u)} = \Psi(\mathbf{x}_v, \mathbf{x}_u)$, in which $\Psi(\mathbf{x}_v, \mathbf{x}_u) = \sigma(\mathbf{W}[\mathbf{x}_v || \mathbf{x}_u])$ where $\mathbf{W}$ is the same for all couples of neighboring nodes. In order to learn asymmetric restriction maps along every edge, the function Ψ should be non-symmetric. Furthermore, the authors show that if the capacity of Ψ is high enough, it is possible to learn any sheaf over a graph.

4. Nonlinear Sheaf Diffusion

In this section, we introduce a general and data-dependent nonlinearity in the Laplacian to enhance sheaf diffusion. We also present NLSD, our nonlinear sheaf diffusion model, along with defining a novel symmetric normalization scheme for the nonlinear Laplacian.

4.1. Nonlinear Laplacian

Let $(G, \mathcal{F})$ be a cellular sheaf on a graph $G = (V, E)$ and let $\delta : C^0(G; \mathcal{F}) \to C^1(G; \mathcal{F})$ denote the coboundary map, as in Section 3.1. We rely on the notation of Hansen & Ghrist (2021) for the following definitions.

Definition 4.1 (Nonlinear sheaf Laplacian)**.** Let $\phi_e : \mathcal{F}(e) \to \mathcal{F}(e)$ be a continuous and not-necessarily-linear map defined edge-wise for each $e \in G$, and $\Phi : C^1(G; \mathcal{F}) \to C^1(G; \mathcal{F})$ the combination of all such maps. The corresponding nonlinear sheaf Laplacian is defined as $L^{\Phi}_{\mathcal{F}} = \delta^T \circ \Phi \circ \delta$. Given $\mathbf{x} \in C^0(G; \mathcal{F})$, since the nonlinear map Φ is applied edge-wise, $L^{\Phi}_{\mathcal{F}}(\mathbf{x})_v$ can still be computed locally in the network as

$$L^{\Phi}_{\mathcal{F}}(\mathbf{x})_v = \sum_{u,v \trianglelefteq e} \mathcal{F}^T_{v\trianglelefteq e}\phi_e(\mathcal{F}_{v\trianglelefteq e}\mathbf{x}_v - \mathcal{F}_{u\trianglelefteq e}\mathbf{x}_u). \tag{8}$$

The nonlinear sheaf diffusion is described by the same PDE as in the linear case (Equation 2), with the nonlinear Laplacian replacing the linear one:

$$\mathbf{X}(0) = \mathbf{X}, \quad \dot{\mathbf{X}}(t) = -L^{\Phi}_{\mathcal{F}}\mathbf{X}(t). \tag{9}$$

4.2. NLSD: model definition

Our Nonlinear Sheaf Diffusion model is directly built upon NSD (Bodnar et al., 2022) described in Equation 7, by introducing the newly defined nonlinear Laplacian:

$$\mathbf{X}_{t+1} = (1+\boldsymbol{\varepsilon})\,\mathbf{X}_t - \sigma\left(L^{\Phi}_{\mathcal{F}(t)}\left(\mathbf{I}_n \otimes \mathbf{W}_1^t\right)\mathbf{X}_t\mathbf{W}_2^t\right). \tag{10}$$

By decomposing the Laplacian with coboundary operators, it can be equally expressed as:

$$\mathbf{Y}_t = \Phi\left(\delta_{\mathcal{F}(t)}\left(\mathbf{I}_n \otimes \mathbf{W}_1^t\right)\mathbf{X}_t\mathbf{W}_2^t\right), \tag{11}$$

$$\mathbf{X}_{t+1} = (1+\boldsymbol{\varepsilon})\,\mathbf{X}_t - \sigma\left(\delta^T_{\mathcal{F}(t)}\mathbf{Y}_t\right). \tag{12}$$

In these equations, the elements $\epsilon, \mathbf{W}_1^t, \mathbf{W}_2^t$ are learned for each layer t. Additionally, even though the nonlinear Laplacian cannot be expressed as a matrix, the coboundary operator δ and its transpose δ^T can be. Thus, their associated operators in Equation 12 are implemented as sparse matrix multiplications.

Nonlinear function definition. The nonlinear function Φ in the Laplacian $L^{\Phi}_{\mathcal{F}}$ is learned from data by NLSD, allowing for greater flexibility in adapting to the problem at hand.

Restriction maps. In NLSD, restriction maps are learned through a parametric matrix-valued function Ψ, following the approach described for NSD in Section 3.2. In the NSD model, three types of sheaves are considered: *diagonal*, *orthogonal*, and *general*, each differing in the constraints imposed on the matrices learned for the restriction maps. The diagonal case imposes a diagonal constraint on $\mathcal{F}_{v\trianglelefteq e}$, the orthogonal case requires $\mathcal{F}_{v\trianglelefteq e} \in O(d)$, and the general case imposes no constraints on the matrix. Unlike NSD, our model only includes diagonal and orthogonal sheaf classes and does not consider general-structured matrices. The reason is that, despite their flexibility, general matrices are more challenging to normalize, make the model harder to train, and increase the risk of overfitting due to the higher number of free parameters (Bodnar et al., 2022). Additionally, experimental results from NSD and its variations (Bodnar et al., 2022; Barbero et al., 2022; Suk et al., 2022) indicate that general matrices perform the worst in terms of accuracy.

Normalization method. As evident from Equation 2 and from the definitions of the SCN and NSD models (Section 3.2), the normalized Laplacian $\Delta_{\mathcal{F}}$ is preferred over the unnormalized Laplacian $L_{\mathcal{F}}$ in practice, as it ensures stability for diffusion dynamics. In our setting, however, unlike the linear case, there is no matrix associated with $L_{\mathcal{F}}^{\Phi}$ that can be considered independently of $\mathbf{x}$ because Φ acts as a nonlinear function of node features and cannot be reduced to a linear operator. Consequently, it is not straightforward to define a normalized sheaf Laplacian $\Delta_{\mathcal{F}}$ through matrix multiplication as in the linear case (Definition 3.5).

To address this problem, we separately normalize the coboundary operator δ and its transpose δ^T, which are linear, unlike $L_{\mathcal{F}}^{\Phi}$. We achieve this by multiplying them by $D^{-\frac{1}{2}}$, where D is the block-diagonal matrix of the *linear* Laplacian $L_{\mathcal{F}}$. The normalized nonlinear Laplacian is defined as $\Delta_{\mathcal{F}}^{\Phi} = D^{-\frac{1}{2}}\delta_{\mathcal{F}}^{T} \circ \Phi \circ \delta_{\mathcal{F}} D^{-\frac{1}{2}}$. Importantly, when Φ is the identity function, this formulation recovers the symmetrically normalized linear Laplacian: $\Delta_{\mathcal{F}} = D^{-\frac{1}{2}} L_{\mathcal{F}} D^{-\frac{1}{2}} = D^{-\frac{1}{2}}\delta_{\mathcal{F}}^{T} \circ \delta_{\mathcal{F}} D^{-\frac{1}{2}}$. This approach has also demonstrated strong practical performance in experimental evaluations.

Considering the implementation choices for the nonlinearity and the normalization criterion discussed above, the NLSD model's update can be expressed as:

$$\mathbf{Y}_t = \Phi\left(\delta_{\mathcal{F}(t)} D^{-\frac{1}{2}} \left(\mathbf{I} \otimes \mathbf{W}_1^t\right) \mathbf{X}_t \mathbf{W}_2^t\right), \tag{13}$$

$$\mathbf{X}_{t+1} = (1+\varepsilon)\,\mathbf{X}_t - \sigma\left(D^{-\frac{1}{2}} \delta_{\mathcal{F}(t)}^{T} \mathbf{Y}_t\right). \tag{14}$$

A general SNN model. We observe that a nonlinear Laplacian is essentially a generalization of the linear one: when its nonlinearity is set to the identity function, the linear version is restored. By building our model upon the Neural Sheaf Diffusion framework (Bodnar et al., 2022), our approach inherently extends it, with the NSD model emerging as a specific case when Φ is defined as the identity function.

5. Experimental Analysis

In this section, we validate our model's ability to effectively leverage complex interactions between nodes to solve a node classification task. To achieve this, we design multiple variations of a synthetic dataset that present a community detection problem, a task that becomes more challenging as more noisy edges are introduced into the graph.

5.1. Dataset design

Our community detection benchmark consists of a node classification problem where the task is to predict the community membership of each node and where node features are little informative. To study how the behavior of the model changes with different connectivities, we designed an incremental evolution of a base graph G_0, where random edges progressively replace the original graph configuration.

The base graph G_0 consists of a set of 1500 nodes uniformly divided into $C = 3$ communities. For each node $v_i \in V$ with fixed community membership $c_i \in \{1, 2, 3\}$, we sample its node features $\mathbf{x}_i$ as follows:

$$\mathbf{x}_i \mid c_i \sim \mathcal{N}(\mu_{c_i}, \sigma I)$$

where σ is fixed to 3 in all our experiments, and

$$\mu_{c_1} = \begin{pmatrix}0\\0\end{pmatrix}, \quad \mu_{c_2} = \begin{pmatrix}0\\1\end{pmatrix}, \quad \mu_{c_3} = \begin{pmatrix}1\\1\end{pmatrix},$$

such that the three distributions have large overlapping regions, making the classification of individual node features into their respective community hard. The edge connectivities are generated according to the two following strategies. The first corresponds to a k-NN graph, where nodes are connected to their k nearest neighbors inside the same community according to a distance function d, which in our case is the L_2 distance between node features. The second corresponds to a *Stochastic block model* (SBM) (Holland et al., 1983; Jaeger et al., 2023), which defines a distribution on graphs according to C communities and a probability matrix $P \in \mathbb{R}^{C\times C}$, indicating the intra-community probabilities on the main diagonal, and inter-community probabilities on off-diagonal entries. In our experiments $P = 0.005I$, which results in the three communities being disconnected, and $k = 4$. We refer to such edge configurations as G_0^{knn} and G_0^{sbm} respectively, and in both cases the communities represent three disconnected components by construction. In the following, we use the general notation G_0 to refer to both G_0^{knn} and G_0^{sbm} simultaneously.

Then, starting from G_0, we define a sequence of perturbed graphs $G_1 \dots G_{10}$, where each G_i is obtained from G_{i-1} by randomly removing $i \cdot 10\%$ of edges and randomly adding the same number of edges, thereby maintaining the total amount of edges constant. We employ three distinct policies for this perturbation process:

- **Intra-only edges**: The newly added edges connect only nodes inside the same community.
- **Inter-only edges**: The newly added edges connect only nodes belonging to different communities.
- **Intra and inter edges**: The newly added edges connect nodes at random. For the SBM distribution, this is equivalent to changing the probability matrix P until reaching a constant matrix, where edges occur equally likely among any community.

5.2. Implementation details

In practice, the nonlinear function Φ in the Laplacian $L_{\mathcal{F}}^{\Phi}$ is defined as a MLP consisting of 3 linear layers, each followed by a ReLU activation function (Brownlee, 2019).

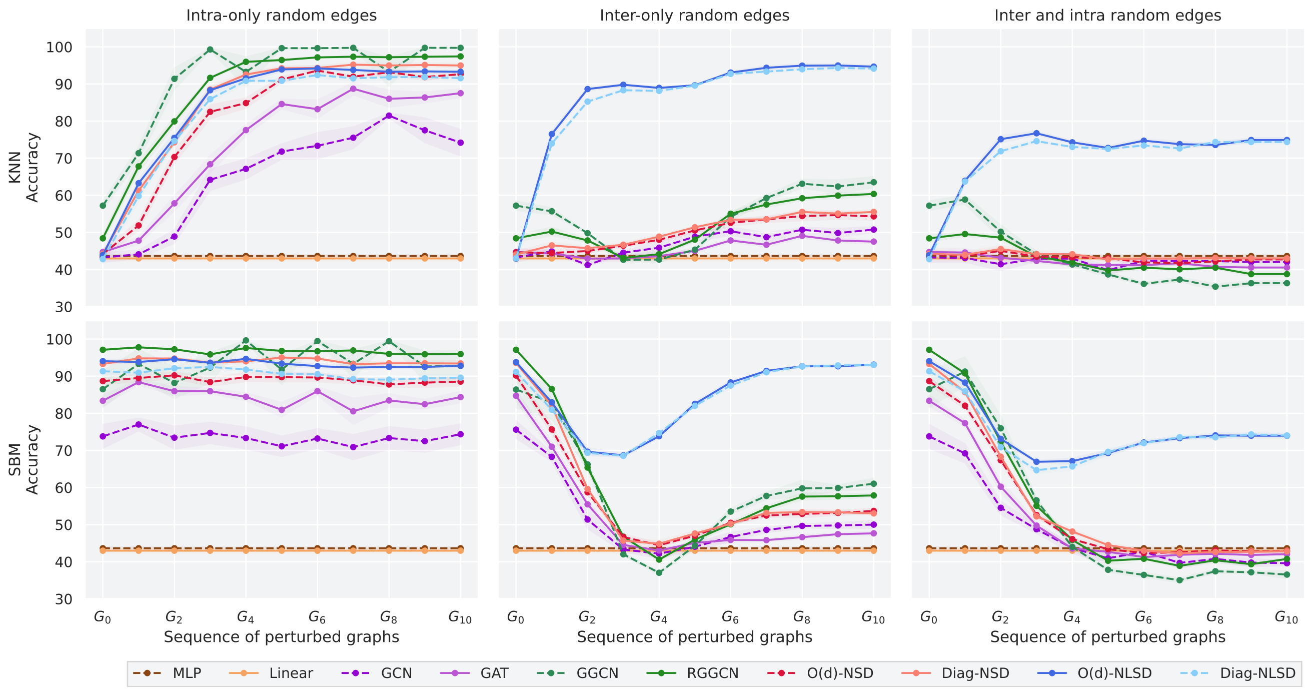

Figure 2. Accuracy results computed on the graph sequence $G_0 \ldots G_{10}$ in six different settings defined by varying the edge perturbation policy and base graph connectivity. The graph sequence is obtained by adding random edges to the base graph G_0 while removing the same number of original edges, keeping the total number of edges in the graph *constant*. The plots showcase the outcomes obtained when only intra-class random edges are added (left column), only inter-class edges (central column), or both inter-class and intra-class edges (right column). The two rows correspond to the two configurations considered for the base graph G_0: k-NN and SBM.

This MLP is directly applied to the result of the coboundary operator, allowing complete flexibility in the shape of the nonlinearity.

In the experiments, we test NLSD with both diagonal and orthogonal sheaf types, referring to them as Diag-NLSD and O(d)-NLSD, respectively. As baselines, we consider four categories of models: (1) Diag-NSD and O(d)-NSD (Bodnar et al., 2022), which are our linear SNN counterparts; (2) classical GNN benchmarks: GCN (Kipf & Welling, 2016) and GAT (Veličković et al., 2017); (3) models particularly suitable for heterophilic settings: GGCN (Yan et al., 2022) and RGGCN (Bresson & Laurent, 2017); and (4) MLP and Linear classifiers on node features.

All architectures are defined with 3 GNN/SNN layers and a final linear classifier. We set the stalk dimension $d = 3$ for all SNN models. We perform a $70\%/30\%$ train/test split on the set of nodes, and we report the results obtained by the models on the test set after 500 training epochs.

5.3. Results

Figure 2 shows the accuracy for the community detection experiment detailed in Section 5.1. The six plots correspond to the different dataset definitions: the upper row refers to the k-NN configuration, and the lower row to SBM, with the columns representing different edge perturbation policies. In each plot, each model is tested on the sequence of graphs $(G_i)_{0\leq i\leq 10}$, with the level of randomness increasing with index i. As expected, simple MLP and Linear baselines, which classify nodes based solely on their features, achieve very low accuracy due to overlapping feature distributions.

The intra-only edge perturbation policy preserves the overall graph connectivity of the SBM case, explaining why the accuracy results are stable over the graph sequence. With k-NN, instead, nodes are connected only to similar nodes within the same community, which makes classification harder for those in the region where distributions overlap. The intra-only results visually show these differences between SBM and k-NN configurations. The difference in accuracy obtained on the first graphs of the sequence $G_0 \ldots G_{10}$ is also reflected in the other two edge perturbation policies.

In the case of inter-only edges, the accuracy of NLSD in the k-NN scenario is similar to the intra-only case. This implies that the newly introduced random edges are highly informative for the model, enabling it to clearly discriminate between them and the initial edges in G_0^{knn}. The same does not hold for the other models, for which the random inter-class edges have a negative effect, becoming slightly

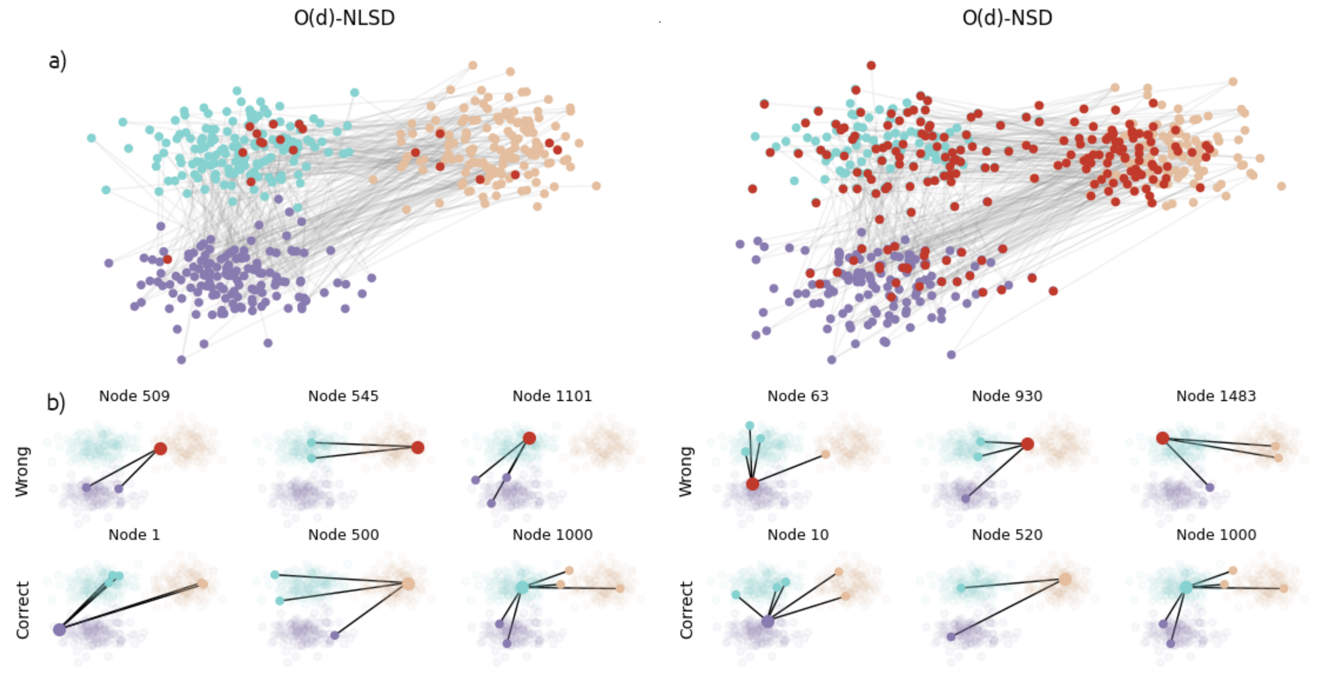

Figure 3. Analysis of the dependency on node features versus connectivity patterns for node classification in O(d)-NLSD and O(d)-NSD. (a) Configuration of wrongly classified nodes (in red), showing no feature-dependent pattern for O(d)-NLSD, while such a pattern emerges for O(d)-NSD, with red nodes clustering in the overlapping region of node feature distributions. (b) Connectivity patterns of classified nodes, indicating that O(d)-NLSD effectively leverages multi-community connections, whereas O(d)-NSD does not.

informative only for sufficiently high i values, where they constitute the majority of edges. The initial drop in accuracy faced by NLSD on $(G_i^{sbm})_{0\leq i\leq 10}$ is explained by the higher difficulty in discriminating between random edges within and between communities with almost overlapping positional node features. The inter-class edges start being leveraged as informative as their percentage increases.

When both intra and inter edges are added, our model behaves similarly to the inter-only case, but with the curves being lower. This decrease is caused by the simultaneous presence of both edge types, which makes them harder to leverage as informative rather than as simple noise. This effect is particularly worse for all other competitors, for which the edge perturbation causes a steady decline in accuracy, leading to worse values than Linear and MLP for $i > 5$.

We conclude that the overall superior performance of NLSD derives from its ability to discriminate between different types of edges (k-NN versus random, random inter-class versus random intra-class) by leveraging information on the distance – in positional features – of neighboring nodes and effectively generalizing from it. Furthermore, we observe that in this setting, the performance of NLSD is robust to the choice of sheaf type, with O(d) slightly outperforming Diag only in a few cases.

As i increases, G_i^{knn} and G_i^{sbm} tend to become increasingly similar, as their connectivity is perturbed in the same way. This explains why the accuracies for the SBM and k-NN cases converge to similar trends across all models and all edge-insertion policies.

5.4. What causes the performance gap?

To gain further insights into the factors behind the *performance gap* between NLSD and the other models, we conducted an in-depth analysis.

Given that the divergence in accuracy plots in Figure 2 occurs upon introducing noisy edges, it is expected to correlate with the models' tendencies to rely more on node features or connectivities. To explore this intuition, we analyzed the positional node features and connectivity patterns of correctly and incorrectly classified nodes in G_{10} when **inter-only** random edges are added. We compared the behavior of our best-performing nonlinear sheaf model, O(d)-NLSD, with its linear counterpart, O(d)-NSD. Our analysis focused on the experiment G_{10}^{knn}, with results generally applicable to both G_{10}^{knn} and G_{10}^{sbm}, since they share the same definition.

The results are shown in Figure 3, where the nodes in the three communities are plotted in positions corresponding to their features, with the means translated enough to avoid overlap, helping visualization. Intuitively, they are shifted

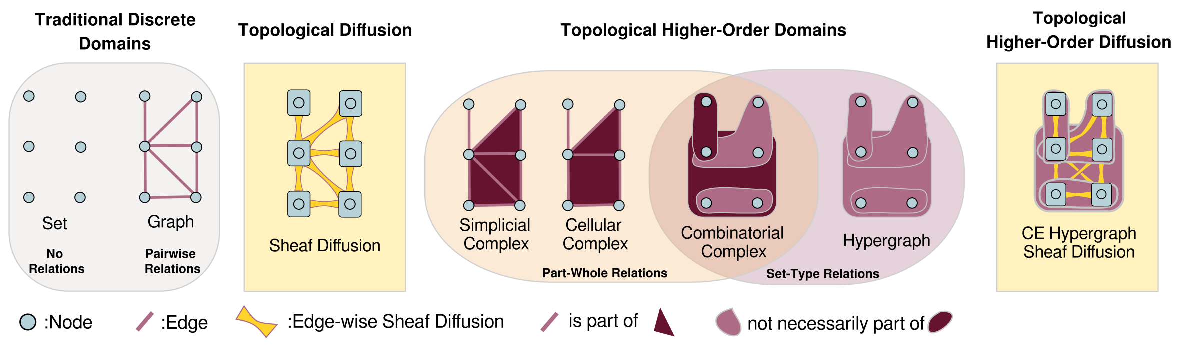

Figure 4. Overview of the Topological Deep Learning framework. We categorize sheaf diffusion and sheaf hypergraph diffusion as *topological diffusion* models, a class of geometry-inspired methods orthogonal to the topological domains. Representation of topological domains adapted from Papillon et al. (2023b).

as if the means of the three distributions were:

$$\mu_{c_1} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \mu_{c_2} = \begin{pmatrix} 0 \\ 6 \end{pmatrix}, \quad \mu_{c_3} = \begin{pmatrix} 6 \\ 6 \end{pmatrix}.$$

Position of wrongly classified nodes. Figure 3 (a) displays the configuration of wrongly classified nodes in red. We observe that, while their positions lack a specific pattern in the O(d)-NLSD case, they tend to cluster within the overlapping region of distributions in the O(d)-NSD case, particularly noticeable for classes 1 and 3. These findings suggest that NSD heavily relies on node features for classification, which may lack adequate informativeness, while NLSD's classification pattern seems to not depend on them.

Analysis of connectivity patterns. Intuitively, since all edges in G_{10} are inter-class, a model that leverages connections will correctly classify a node connected to both communities where it does not belong, using information from neighboring nodes to exclude those communities. However, if a node is connected to only one different class, the model's performance should suffer due to insufficient information for accurate class discrimination. Figure 3 (b) illustrates that the O(d)-NLSD model tends to classify nodes more accurately when they are connected to both of the other two communities. In contrast, wrongly classified nodes are typically connected to just one other community, suggesting that the model heavily relies on edge connections. On the other hand, the performance of O(d)-NSD appears to be independent of connectivity information, as there is no evident correlation between classification accuracy and multi-community connections.

Furthermore, we observe a significant difference between the average degree of correctly and wrongly classified nodes in the O(d)-NLSD case, but not in the O(d)-NSD case. This also suggests a correlation between classification accuracy and connectivity patterns for the former model, but not for the latter. More details are in Appendix A.

These analyses confirm that the linear model primarily relies on node features instead of leveraging connections, while the opposite holds for our nonlinear model.

5.5. Limitations

A preliminary analysis on standard real-world node-classification datasets did not show clear improvements over linear NSD (Bodnar et al., 2022) models (see Appendix B for details), likely because these datasets do not exhibit clear community patterns. We leave the investigation of the effectiveness of NLSD on real-world community detection datasets to future work.

6. SNNs in Topological Deep Learning

TDL focuses on modeling multi-way higher-order relations among nodes while establishing a common ground for multiple topological domains and Topological Neural Networks (TNNs) (Papillon et al., 2023b;a; Hajij et al., 2024). Due to the scarcity of natural higher-order data, TDL generally employs a lifting mechanism applied to graph topology and features. SNNs inherently perform feature lifting, while the underlying topological structure remains unaltered.

In Figure 4, we contextualize existing cellular sheaf diffusion models within the general TDL framework defined by Papillon et al. (2023b), identifying them as *topological diffusion* models. The topological enhancement derives from the implicit feature lifting they employ and the geometrically richer information propagation between nodes, complementing both topological domains and TNN classes.

While it is natural to define a cellular sheaf on a graph structure, it is not as straightforward on higher-order domains, making the definition of a sheaf diffusion process arbitrary and open to different interpretations. Alongside the basic sheaf diffusion model on graphs, we include in Figure 4 the clique-expansion-based (CE) hypergraph sheaf diffusion of the SheafHyperGNN model from Duta et al. (2024), which defines a clique for each hyperedge and executes sheaf diffusion on the newly defined edges. The SheafHyperGCN model arises from a slightly different underlying sheaf definition, which we do not visualize for brevity.

7. Conclusion

In this paper, we introduced the Nonlinear Sheaf Diffusion model, which enhances SNNs for graph learning by incorporating a learned nonlinear function into the Laplacian. This approach allows for greater flexibility and expressiveness in capturing complex node interactions. Tests conducted on a community detection synthetic benchmark showed that the NLSD model outperforms both linear SNNs and common GNN baselines. Our findings indicate the potential of NLSD to significantly improve graph-based learning tasks and suggest future research directions, particularly in exploring real-world community detection applications.

Acknowledgements and Disclosure of Funding

O.Z. is grateful to Cristian Bodnar, Francesco Di Giovanni, Lorenzo Giusti, and Federico Siciliano for their precious advice and insightful discussions. We also appreciate the anonymous reviewers for their constructive feedback and comments.

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Health and Digital Executive Agency (HaDEA). Neither the European Union nor the granting authority can be held responsible for them. Grant Agreement no. 101120763 - TANGO. A.L. acknowledges the support of the MUR PNRR project FAIR - Future AI Research (PE00000013) funded by the NextGenerationEU.

References

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

Bodnar, C. *Topological Deep Learning: Graphs, Complexes, Sheaves*. PhD thesis, Apollo - University of Cambridge Repository, 2022. URL `https://www.repository.cam.ac.uk/handle/1810/350982`.

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

Bresson, X. and Laurent, T. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017.

Brownlee, J. A gentle introduction to the rectified linear unit (relu). *Machine learning mastery*, 6, 2019.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

Curry, J. M. *Sheaves, cosheaves and applications*. University of Pennsylvania, 2014.

Duta, I., Cassarà, G., Silvestri, F., and Liò, P. Sheaf hypergraph networks. *Advances in Neural Information Processing Systems*, 36, 2024.

Feng, Y., You, H., Zhang, Z., Ji, R., and Gao, Y. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 3558–3565, 2019.

Giusti, L., Battiloro, C., Testa, L., Di Lorenzo, P., Sardellitti, S., and Barbarossa, S. Cell attention networks. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2023.

Gori, M., Monfardini, G., and Scarselli, F. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pp. 729–734. IEEE, 2005.

Hajij, M., Istvan, K., and Zamzmi, G. Cell complex neural networks. *arXiv preprint arXiv:2010.00743*, 2020.

Hajij, M., Zamzmi, G., Papamarkou, T., Miolane, N., Guzmán-Sáenz, A., Ramamurthy, K. N., Birdal, T., Dey, T. K., Mukherjee, S., Samaga, S. N., et al. Topological deep learning: Going beyond graph data. *arXiv preprint arXiv:2206.00606*, 2022.

Hajij, M., Papillon, M., Frantzen, F., Agerberg, J., AlJabea, I., Ballester, R., Battiloro, C., Bernárdez, G., Birdal, T., Brent, A., Chin, P., Escalera, S., Fiorellino, S., Gardaa, O. H., Gopalakrishnan, G., Govil, D., Hoppe, J., Karri, M. R., Khouja, J., Lecha, M., Livesay, N., Meißner, J., Mukherjee, S., Nikitin, A., Papamarkou, T., Prílepok, J., Ramamurthy, K. N., Rosen, P., Guzmán-Sáenz, A., Salatiello, A., Samaga, S. N., Scardapane, S., Schaub, M. T., Scofano, L., Spinelli, I., Telyatnikov, L., Truong, Q., Walters, R., Yang, M., Zaghen, O., Zamzmi, G., Zia, A., and Miolane, N. Topox: A suite of python packages for machine learning on topological domains, 2024.

Hansen, J. and Gebhart, T. Sheaf neural networks. *arXiv preprint arXiv:2012.06333*, 2020.

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

Hansen, J. and Ghrist, R. Opinion dynamics on discourse sheaves. *SIAM Journal on Applied Mathematics*, 81(5): 2033–2060, 2021.

Holland, P. W., Laskey, K. B., and Leinhardt, S. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983. ISSN 0378-8733. doi: https://doi.org/10.1016/0378-8733(83)90021-7. URL https://www.sciencedirect.com/science/article/pii/0378873383900217.

Jaeger, M., Longa, A., Azzolin, S., Schulte, O., and Passerini, A. A simple latent variable model for graph learning and inference. In *The Second Learning on Graphs Conference*, 2023. URL https://openreview.net/forum?id=S9jem2KZVr.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Micheli, A. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009.

Namata, G., London, B., Getoor, L., Huang, B., and Edu, U. Query-driven active surveying for collective classification. In *10th international workshop on mining and learning with graphs*, volume 8, pp. 1, 2012.

Papamarkou, T., Birdal, T., Bronstein, M., Carlsson, G., Curry, J., Gao, Y., Hajij, M., Kwitt, R., Liò, P., Di Lorenzo, P., et al. Position paper: Challenges and opportunities in topological deep learning. *arXiv preprint arXiv:2402.08871*, 2024.

Papillon, M., Hajij, M., Myers, A., , Jenne, H., Mathe, J., Papamarkou, T., Guzmán-Sáenz, A., Livesay, N., Dey, T., Rabinowitz, A., Brent, A., Salatiello, A., Nikitin, A., Zia, A., Battiloro, C., Gavrilev, D., Magai, G., Bazhenov, G., Bernardez, G., Spinelli, I., Agerberg, J., Nadimpalli, K., Telyatninkov, L., Scofano, L., Testa, L., Lecha, M., Yang, M., Hassanin, M., Gardaa, O. H., Zaghen, O., Hausner, P., Snopoff, P., Ballester, R., Barikbin, S., Escalera, S., Fiorellino, S., Kvinge, H., Ramamurthy, K. N., Rosen, P., Walters, R., Samaga, S. N., Mukherjee, S., Sanborn, S., Emerson, T., Doster, T., Birdal, T., Khamis, A., Scardapane, S., Singh, S., Malygina, T., Yue, Y., and Miolane, N. Icml 2023 topological deep learning challenge: Design and results. In Doster, T., Emerson, T., Kvinge, H., Miolane, N., Papillon, M., Rieck, B., and Sanborn, S. (eds.), *Proceedings of 2nd Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML)*, volume 221 of *Proceedings of Machine Learning Research*, pp. 3–8. PMLR, 28 Jul 2023a. URL https://proceedings.mlr.press/v221/papillon23a.html.

Papillon, M., Sanborn, S., Hajij, M., and Miolane, N. Architectures of topological deep learning: A survey on topological neural networks. *Arxiv. Submitted to Transactions on Pattern Analysis and Machine Intelligence*, 2023b.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

Rozemberczki, B., Allen, C., and Sarkar, R. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

Rusch, T. K., Bronstein, M. M., and Mishra, S. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Schaub, M. T., Seby, J.-B., Frantzen, F., Roddenberry, T. M., Zhu, Y., and Segarra, S. Signal processing on simplicial complexes. In *Higher-Order Systems*, pp. 301–328. Springer, 2022.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Shepard, A. D. *A cellular description of the derived category of a stratified space*. Brown University, 1985.

Sperduti, A. Encoding labeled graphs by labeling raam. *Advances in Neural Information Processing Systems*, 6, 1993.

Suk, J., Giusti, L., Hemo, T., Lopez, M., Barmpas, K., and Bodnar, C. Surfing on the neural sheaf. In *NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations*, 2022. URL https://openreview.net/forum?id=xOXFkyRzTlu.

Tang, J., Sun, J., Wang, C., and Yang, Z. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 807–816, 2009.

Telyatnikov, L., Bucarelli, M. S., Bernardez, G., Zaghen, O., Scardapane, S., and Lio, P. Hypergraph neural networks through the lens of message passing: a common perspective to homophily and architecture design. *arXiv preprint arXiv:2310.07684*, 2023.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Wu, X., Chen, Z., Wang, W., and Jadbabaie, A. A non-asymptotic analysis of oversmoothing in graph neural networks. *arXiv preprint arXiv:2212.10701*, 2022.

Yan, Y., Hashemi, M., Swersky, K., Yang, Y., and Koutra, D. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In *2022 IEEE International Conference on Data Mining (ICDM)*, pp. 1287–1292. IEEE, 2022.

Yang, M. and Isufi, E. Convolutional learning on simplicial complexes. *arXiv preprint arXiv:2301.11163*, 2023.

A. Further Analysis for Synthetic Experiments

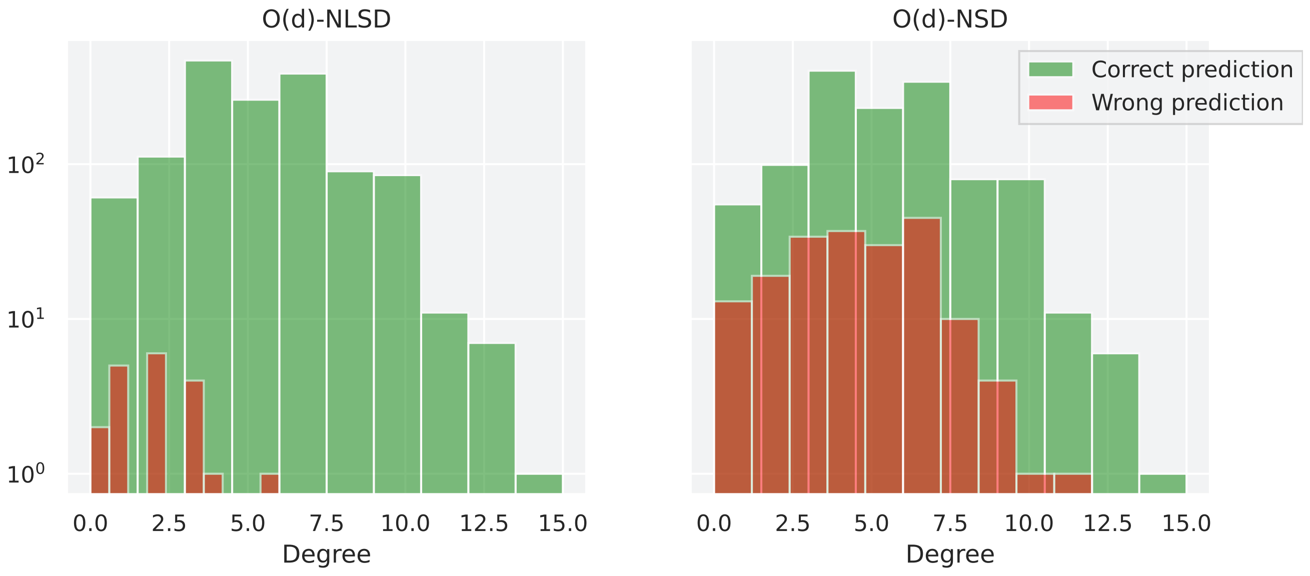

Figure 5. Degree distribution of correctly and wrongly classified nodes by O(d)-NLSD (left) and O(d)-NSD (right) on G_{10}^{knn} in the inter-only edge perturbation pattern.

In Figure 5, we present the degree distribution for correctly predicted nodes (shown in green) and incorrectly predicted nodes (shown in red) for both the O(d)-NLSD and O(d)-NSD models. Notably, our model tends to make incorrect predictions primarily for nodes with a lower degree, especially in cases where the target node is exclusively connected to a single community. This observation corroborates the analysis discussed in Section 5.4.

To further illustrate this point, Table 1 provides the average degree of correctly and incorrectly predicted nodes with O(d)-NLSD and O(d)-NSD in the synthetic experiment using the k-NN dataset. The data reveals a clear discrepancy between the degrees of correctly and incorrectly predicted nodes, underscoring the relationship between node degree and prediction accuracy.

Table 1. Average degree of correctly and wrongly predicted nodes with O(d)-NLSD and O(d)-NSD on G_{10}^{knn} in the inter-only edge perturbation pattern.

	Correct	Wrong
O(d)-NLSD	5.0 ($\pm$ 2.2)	2.1 ($\pm$ 1.4)
O(d)-NSD	5.0 ($\pm$ 2.2)	4.5 ($\pm$ 2.1)

From a more general perspective, these results further demonstrate a correlation between classification accuracy and connectivity patterns for NLSD, but not for NSD. This highlights that the former relies more on connectivity, while the latter relies more on node features.

B. Real-World Experiments

As preliminary experiments on real-world data, we tested our model in the same experimental setting considered by Bodnar et al. (2022), which aimed at testing the ability of their model to tackle the oversmoothing issue emerging in GNNs, as well as its ability to handle heterophilic datasets.

B.1. Datasets

Motivated by the experiments conducted by Bodnar et al. (2022), we assess the performance of Diag-NLSD and O(d)-NLSD across various homophilic and heterophilic real-world datasets (Pei et al., 2020; Namata et al., 2012; Rozemberczki et al., 2021; Sen et al., 2008; Tang et al., 2009), comparing them with models from the SNN and GNN literature. These real-world datasets exhibit edge homophily coefficients ranging from $h = 0.11$ (indicating high heterophily) to $h = 0.81$ (indicating high homophily). We gather the results through 10 fixed splits and 10-fold Cross Validation, allocating 48%, 32%, and 20% of nodes per class for training, validation, and testing, respectively. The results displayed in Table 2 are chosen based on the test accuracy corresponding to the highest validation accuracy.

Table 2. Results on node classification datasets sorted by their homophily level. Top three models are coloured by **First**, **Second**, **Third**. Our models are marked **NLSD** (Non Linear Sheaf Diffusion). Table adapted from Bodnar et al. (2022).

	Texas	**Wisconsin**	**Film**	**Squirrel**	**Chameleon**	**Cornell**	**Citeseer**	**Pubmed**	**Cora**
Hom level	**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
Diag-NLSD	**86.22**±3.91	**89.02**±3.19	**37.45**±0.94	47.63±1.51	62.57±2.31	**86.76**±4.60	75.76±1.62	89.52±0.32	86.38±1.20
O(d)-NLSD	**86.22**±4.90	**89.02**±3.84	37.22±1.15	51.96±2.65	65.37±2.73	**87.03**±4.49	76.11±1.81	**89.60**±0.29	86.20±1.24
NSP (best)	**87.03**±5.51	**89.02**±3.84	37.12±1.31	50.11±2.03	62.85±1.98	76.49±5.28	76.85±1.48	89.42±0.33	87.38±1.14
SheafAN (best)	–	–	–	–	**68.62**±2.81	85.68±4.53	76.86±1.71	–	87.08±1.26
NSD (best)	**85.95**±5.51	**89.41**±4.74	**37.81**±1.15	**56.34**±1.32	**68.68**±1.73	**86.49**±7.35	**77.14**±1.85	89.49±0.40	87.30±1.15
GGCN	84.86±4.55	86.86±3.29	**37.54**±1.56	**55.17**±1.58	**71.14**±1.84	85.68±6.63	**77.14**±1.45	89.15±0.37	**87.95**±1.05
H2GCN	84.86±7.23	87.65±4.98	35.70±1.00	36.48±1.86	60.11±2.15	82.70±5.28	77.11±1.57	89.49±0.38	**87.87**±1.20
GPRGNN	78.38±4.36	82.94±4.21	34.63±1.22	31.61±1.24	46.58±1.71	80.27±8.11	77.13±1.67	87.54±0.38	**87.95**±1.18
FAGCN	82.43±6.89	82.94±7.95	34.87±1.25	42.59±0.79	55.22±3.19	79.19±9.79	N/A	N/A	N/A
MixHop	77.84±7.73	75.88±4.90	32.22±2.34	43.80±1.48	60.50±2.53	73.51±6.34	76.26±1.33	85.31±0.61	87.61±0.85
GCNII	77.57±3.83	80.39±3.40	37.44±1.30	38.47±1.58	63.86±3.04	77.86±3.79	**77.33**±1.48	**90.15**±0.43	**88.37**±1.25
Geom-GCN	66.76±2.72	64.51±3.66	31.59±1.15	38.15±0.92	60.00±2.81	60.54±3.67	**78.02**±1.15	**89.95**±0.47	85.35±1.57
PairNorm	60.27±4.34	48.43±6.14	27.40±1.24	50.44±2.04	62.74±2.82	58.92±3.15	73.59±1.47	87.53±0.44	85.79±1.01
GraphSAGE	82.43±6.14	81.18±5.56	34.23±0.99	41.61±0.74	58.73±1.68	75.95±5.01	76.04±1.30	88.45±0.50	86.90±1.04
GCN	55.14±5.16	51.76±3.06	27.32±1.10	**53.43**±2.01	64.82±2.24	60.54±5.30	76.50±1.36	88.42±0.50	86.98±1.27
GAT	52.16±6.63	49.41±4.09	27.44±0.89	40.72±1.55	60.26±2.50	61.89±5.05	76.55±1.23	87.30±1.10	86.33±0.48
MLP	80.81±4.75	85.29±3.31	36.53±0.70	28.77±1.56	46.21±2.99	81.89±6.40	74.02±1.90	87.16±0.37	75.69±2.00

B.2. Models

In the experiments, we test NLSD with both diagonal and orthogonal sheaf types, referring to them as Diag-NLSD and O(d)-NLSD, respectively. As linear SNN baselines, we report the best results obtained by Diag-NLSD and O(d)-NLSD (Bodnar et al., 2022), as well as those of NSP (Suk et al., 2022) and SheafAN (Barbero et al., 2022), two variants of the NSD framework. The other models used for comparison are exactly the same as those chosen in the analysis carried out by Bodnar et al. (2022). For a complete description and motivation of the choices, please refer to their paper.

B.3. Results

Experiments on real-world benchmark datasets show that NLSD achieves comparable results to NSD (Bodnar et al., 2022) and its variations SheafAN (Barbero et al., 2022) and NSP (Suk et al., 2022). The results in Table 2 indicate that the introduction of a nonlinearity in the sheaf Laplacian does not lead to substantial differences in the considered real-world scenarios. We can deduce from the results that our model performs better when handling heterophilic datasets compared to homophilic ones, with results being close to state-of-the-art.

We recognize the limitations of NLSD on these datasets, as it does not consistently outperform its linear counterpart, NSD. We believe that these real-world settings are not ideal for leveraging the expressiveness of our model, and we propose future work to test our model on real-world community detection datasets, which we believe would provide a better application.