Heterogeneous Sheaf Neural Networks

Luke Braithwaite[1,*] **Alessio Borgi**[1,3,*] **Gabriele Onorato**[2,†,*] **Kristjan Tarantelli**[3,*] **Francesco Restuccia**[2] **Fabrizio Silvestri**[3] **Pietro Liò**[1]

[1]*Department of Computer Science and Technology, University of Cambridge*

[2]*Department of Electrical and Computer Engineering, Northeastern University*

[3]*Department of Computer, Control and Management Engineering, Sapienza University of Rome*

Editors: Louisa Cornelis, Francisco Acosta, Alison Pouplin, David Klindt, Fatih Dinc, Nina Miolane

Abstract

Heterogeneous graphs, whose nodes and edges can belong to different types and feature spaces, arise in many real-world domains, including biology, recommendation, social networks, and computer systems. Existing heterogeneous graph neural networks typically handle this heterogeneity at the architectural level through relation-specific modules, meta-path machinery or type-aware attention, which often leads to increasingly specialised, parameter-heavy designs. In this work, we propose HetSheaf, a framework for learning heterogeneous graphs through *cellular sheaves*. Instead of encoding heterogeneity solely in the architecture, HetSheaf represents it directly in the underlying data structure by assigning type-aware local feature spaces and learning restriction maps conditioned on node features, node types, and edge types. To support graph-level prediction, we further introduce SheafPool, a universal stalk-space readout that aggregates node representations while being invariant to local changes of basis, thereby making graph classification with sheaf networks well-defined and achieving an F1 Score up to 42 percentage points higher than mean pooling. Across a diverse suite of benchmarks (node classification, link prediction and graph classification), HetSheaf consistently achieves up to 2 percentage points higher performance (up to 94.93% Macro F1 Score on node classification and up to 99.64% on link prediction) on the Heterogeneous Graph Benchmark (HGB) framework against homogeneous (GCN, GAT, GIN, GraphSAGE), heterogeneous (R-GCN, HAN, HGT) and type-agnostic sheaf baselines, while reducing the number of parameters by up to 10×.

Keywords: Cellular Sheaves, Heterogeneous Graph Neural Networks, Sheaf Pooling

1. Introduction

Graphs are a natural representation in many scientific fields, with Graph Neural Networks (GNNs) being one of the main tools to model relational data [26, 27, 56, 8, 14, 64, 25]. Traditional GNNs implicitly assume that the underlying graph is *homogeneous*; i.e., all nodes belong to a single category and share the same feature space, and all edges represent the same kind of relationship. However, many real-world graphs are *heterogeneous*; nodes and edges may have different types, different meanings, and even different input dimensionalities [18, 54, 42, 29, 36, 72]. Consequently, treating graphs as homogeneous limits the model's ability to capture complex patterns, leading to sub-optimal performances. To overcome these limitations, *heterogeneous GNNs* (HGNNs) enrich standard GNN architectures with edge-specific transformations, hand-crafted meta-path guided aggregations or attention pipelines [57, 76, 37, 66]. While effective, this work typically handles heterogeneity through

* Equal contribution. †Corresponding author: onorato.g@northeastern.edu

© 2026 L. Braithwaite, A. Borgi, G. Onorato, K.. Tarantelli, F. Restuccia, F. Silvestri & P. Liò.

ad-hoc modifications, which often require more relation-specific components in the architecture, more hand-crafted design choices, or more parameters, as the as the number of node and edge types grows. Second, the burden of representing heterogeneous structure is placed on the neural network itself instead of on the underlying data representation. In other words, the graph topology is converted to its homogeneous variant and the model must compensate for the heterogeneity of local spaces. To address these limitations, we propose HetSheaf, a

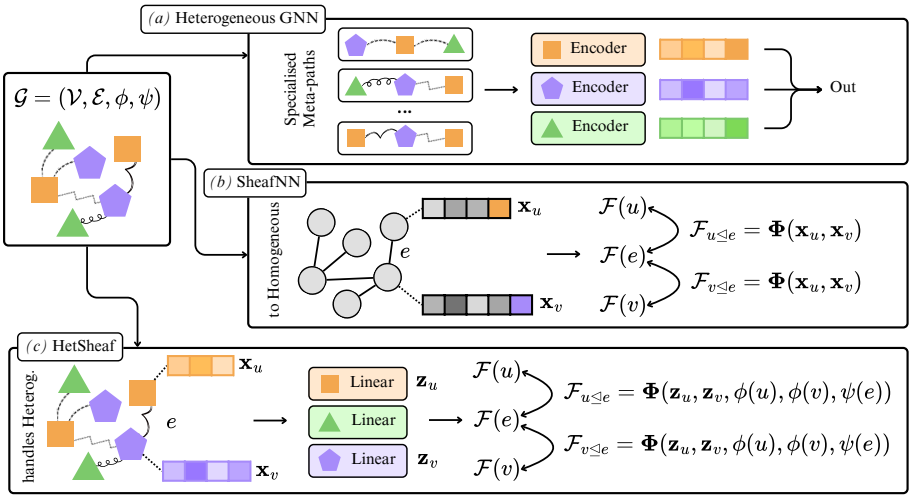


Figure 1: *(a)* Conventional Heterogeneous GNNs encode type information through specialised architectural effects. *(b)* Sheaf Neural Networks (SheafNNs) require a homogeneous graph representation; this can be artificially constructed by encoding node types into the feature vectors $\mathbf{x}_u$ and $\mathbf{x}_v$ to infer the restriction maps $\mathcal{F}_{u \trianglelefteq e := (u,v)}$ between stalks $\mathcal{F}_u$ and $\mathcal{F}_v$. *(c)* HetSheaf maps type-dependent features $\mathbf{x}_{u|v}$ to a shared space $\mathbf{z}_{u|v}$ and natively handles heterogeneity by explicitly parameterising the sheaf's restriction maps with the type information of nodes $\phi(\cdot)$ and edges $\psi(\cdot)$.

framework specifically tailored for learning on heterogeneous graphs through *cellular sheaves*. A *sheaf* augments a graph by attaching a local vector space (or *stalk*) to each node and edge, which acts as a dedicated feature space for that specific entity. Information is translated between these local spaces via linear transformations called *restriction maps*. HetSheaf, first projects type-dependent input features into a shared channel dimension and then infers a cellular sheaf where restriction maps are conditioned on node features, node types, and edge types. Moreover, to execute graph-level inductive tasks on cellular sheaves a new challenge emerges in aggregating feature vectors from all nodes (i.e., graph pooling): in standard GNNs, all node embeddings lie in the same latent space, so graph classification can be performed by applying any permutation-invariant pooling operation (e.g., mean, sum, or max). Conversely, in sheaf networks, nodes and edges have their own stalk space and are only defined *up to a change of basis*, i.e., two embeddings might represent the same underlying information but have different coordinate matrices. Consequently, flattening node embeddings and pooling them can yield different graph representations for mathematically equivalent sheaf states. Therefore, the graph pooling must be *gauge-invariant*, i.e., tolerate node-wise reparameterization of the stalk coordinates. To address this, we introduce SheafPool, the first stalk-space pooling for graph classification. SheafPool constructs

a stalk graph representation by aggregating node stalks independently of their arbitrary local coordinates. To achieve this, it resolves coordinate ambiguities and rotates the node embeddings into a common reference frame prior to the final aggregation step.

Summary of main contributions

1. We propose HetSheaf, a model-agnostic framework for heterogeneous sheaf neural networks that preserves type-specific semantics in the data representation without specialised, parameter-heavy models (Figure 1). We instantiate it with Neural Sheaf Diffusion (NSD) [6], Sheaf Attention Networks (SheafAN) [4], and Polynomial Neural Sheaf Diffusion (PolyNSD) [7].
2. We introduce a family of *heterogeneous sheaf predictors* that infer restriction maps by explicitly conditioning on node features, node and edge types, allowing the learned sheaf structure to adapt to the typed nature of the underlying relational data (Figure 2).
3. We introduce SheafPool, the first graph-level pooling for sheaf neural networks. Since stalk embeddings lie in distinct coordinate systems, standard pooling is not geometrically well defined. SheafPool resolves it through whitening, anchor-based alignment, invariant attention, and late-token processing, yielding an injective, architecture-agnostic pooling that improves Macro F1 by up to 42% over mean pooling (Section 4).
4. We evaluate HetSheaf and SheafPool on node classification, link prediction, and graph classification benchmarks, following the HGB setup [45] (Section 5). HetSheaf outperforms type-agnostic sheaf models and specialised HGNNs by up to 2%, while being up to 10× more parameter efficient compared to R-GCN, HAN, and HGT [37, 57, 66] (Appendix C.4.5).

2. Background and Related Work

Homogeneous and Heterogeneous Graph Neural Networks. Homogeneous GNNs [31, 39, 43, 64, 14, 12] can be applied to heterogeneous graphs by discarding node and edge types, and consequently they do not model type-specific relational structure. Heterogeneous GNNs instead are also able to model the type of nodes and edges. Some of them use meta-paths, predefined type patterns [62, 61, 15], or relation-specific processing. HAN [66] and MAGNN [24] aggregate information through manually specified meta-paths; GTN [74] learns such paths, although they can be costly with many edge types. Relational approaches include RSHN [80], which derives edge features on a coarsened line graph before MPNN propagation [25]; R-GCN [57], which applies GCNs to relation-induced subgraphs; HGT and HetGNN [76], which respectively use sampled transformer attention and random-walk/RNN aggregation; and SlotGAT [78], SimpleHGNN [45], and Space4HGNN [77]. Recent examples include HAGNN [79], which fuses meta-path-based intra-type and meta-path-free inter-type aggregation, and HGNN-AR^2 [44], which adaptively reconstructs relations. Rather than adding specialised modules, we represent type-specific information through cellular sheaves.

Sheaf Neural Networks. Cellular sheaves [59, 13] attach local feature spaces (*stalks*) and linear restriction maps to graphs, offering a geometric response to oversmoothing [9], over-squashing [3], and heterophily. SheafNNs progressed from fixed one-dimensional sheaves [32] to learned restriction maps [6], and now include attention [4], learned connection Laplacians [5], nonlinear and joint diffusion [75, 10], polynomial filtering [7], directional models [23], and extensions to hypergraphs [17, 49], recommendation [53], federated learning [50], cooperative dynamics [55], and copresheaves [30]. Most work concerns node-level tasks. For

graph prediction, readout must be both permutation-invariant and invariant to node-wise stalk reparameterisations; standard pooling [28, 38, 43, 40] only guarantees the former.

Cellular Sheaves on Graphs. For an oriented graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a cellular sheaf $\mathcal{F}$ assigns stalks $\mathcal{F}(u)$ and $\mathcal{F}(e)$ to nodes and edges, with restriction maps $\mathcal{F}_{u \trianglelefteq e} : \mathcal{F}(u) \to \mathcal{F}(e)$ for every incidence. Its 0- and 1-cochains are $C^0(\mathcal{G}, \mathcal{F}) = \bigoplus_{u \in \mathcal{V}} \mathcal{F}(u)$ and $C^1(\mathcal{G}, \mathcal{F}) = \bigoplus_{e \in \mathcal{E}} \mathcal{F}(e)$. For $e = (u \to v)$, the coboundary $\delta(\mathbf{x})_e = \mathcal{F}_{v \trianglelefteq e}\mathbf{x}_v - \mathcal{F}_{u \trianglelefteq e}\mathbf{x}_u$ measures disagreement in the shared edge space [33]; $L_{\mathcal{F}} = \delta^\top \delta$ is the sheaf Laplacian. With d-dimensional stalks and block diagonal D of $L_{\mathcal{F}}$, its normalised action is $\Delta_{\mathcal{F}}(\mathbf{x})_u := D^{-1/2}\left[\sum_{u,v \trianglelefteq e} \mathcal{F}_{u \trianglelefteq e}^\top (\mathcal{F}_{u \trianglelefteq e}\mathbf{x}_u - \mathcal{F}_{v \trianglelefteq e}\mathbf{x}_v)\right] D^{-1/2}$. Thus $L_{\mathcal{F}} \in \mathbb{R}^{dn \times dn}$ for $n = |\mathcal{V}|$; scalar stalks with identity restrictions recover the graph Laplacian. For f feature channels, $\mathbf{X} \in \mathbb{R}^{nd \times f}$, and NSD learns restrictions as $\mathcal{F}_{u \trianglelefteq e := (u,v)} = \mathbf{\Phi}(\mathbf{x}_u, \mathbf{x}_v) = \mathrm{MLP}(\mathbf{x}_u \| \mathbf{x}_v)$.

3. Heterogeneous Sheaf Neural Networks

Heterogeneous Graph Setting. We consider a heterogeneous graph [61] $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \phi, \psi)$, where $\mathcal{V}$ and $\mathcal{E}$ denote the sets of nodes and edges, while $\phi : \mathcal{V} \to \Phi$ and $\psi : \mathcal{E} \to \Psi$ are node and edge type mapping functions respectively. For an edge $e = (u, v)$, we interchangeably write its type as $\psi(e)$ or $\psi(u, v)$. Each node $u \in \mathcal{V}$ is associated with a feature vector $\mathbf{x}_u$. Because feature sizes vary by node type, all nodes of a given type $t \in \Phi$ share a specific input dimension f_t, meaning $\mathbf{x}_u \in \mathbb{R}^{f_{\phi(u)}}$.

3.1. HetSheaf: A Sheaf-based framework for Heterogeneous data

Existing heterogeneous GNNs (Figure 1a) handle heterogeneity at the architectural level, often leading to increasingly specialised and parameter-heavy designs that scale with the number of entity types. Instead, we adopt a *representation-centric view* by modifying the input structure to encode specific heterogeneous knowledge while preserving topological information. Unlike standard graphs, which assume a single shared feature space for all nodes, cellular sheaves naturally capture heterogeneous data through type-specific local vector spaces and restriction maps that preserve local structure while enabling coherent global message passing. Motivated by this perspective, we introduce HetSheaf (Figure 1c), a general framework for heterogeneous SheafNNs. Given a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \phi, \psi)$, it first preprocesses type-dependent node features $\mathbf{x}$ into a common channel dimension $\mathbf{z}$, then infers a cellular sheaf whose restriction maps are conditioned on both features and type information, and finally applies a sheaf neural architecture for downstream prediction. More formally, a heterogeneous sheaf neural network consists of three components:

1. *Type-aware preprocessing*, which projects input features $\mathbf{x}$ into a shared channel space $\mathbf{z}$. Because each node type may have a different number of feature channels, we use a linear layer with bias for each node type to project node features into a common number of input channels. This can be combined with the sheaf projection step of the SheafNN to create the *heterogeneous sheaf cochain* $\mathbf{z} \in C^0(\mathcal{G}, \mathcal{F})$. The modified linear layer takes in the original feature channels f_t and outputs df features as the node stalk projection. We follow [45, 78] to generate the input feature types f by using either all of the node feature types (type-0), only the feature of the node target type (type-1) or replacing all node features with one-hot vectors (type-2).
2. *Heterogeneous sheaf inference*, (Figure 2) which predicts restriction maps conditioned on node features and type information.

3. *Sheaf-based representation learning*, which applies a sheaf neural encoder followed by a task-specific decoder or readout. We apply task-specific representations (e.g., ℓ_2 normalisation [45] or JKNet-style concatenations [69]) before the final predictions, as detailed in Section 3.2.

Let $\mathcal{P}$ denote the preprocessing module, $\mathbf{\Phi}$ the sheaf predictor, $\mathcal{M}$ a sheaf neural encoder, and $\mathcal{D}$ a downstream decoder. The overall pipeline can be written abstractly as:

$$\mathbf{z}_u = \mathcal{P}_{\phi(u)}(\mathbf{x}_u), \quad \mathcal{F}_{u \trianglelefteq e:=(u,v)} = \mathbf{\Phi}(\mathbf{z}_u, \mathbf{z}_v, \phi(u), \phi(v), \psi(e)), \quad \hat{\mathbf{y}} = \mathcal{D}(\mathcal{M}(\mathcal{G}, \mathcal{F}, \{\mathbf{z}_u\}_{u \in \mathcal{V}})) \tag{1}$$

Since heterogeneity is encoded into the inferred sheaf itself, once the sheaf has been constructed, it can be processed by any sheaf-based neural architecture, including NSD [6], Sheaf Attention Networks [4], Polynomial NSD [7] or other sheaf variants, without requiring ad hoc heterogeneous redesign.

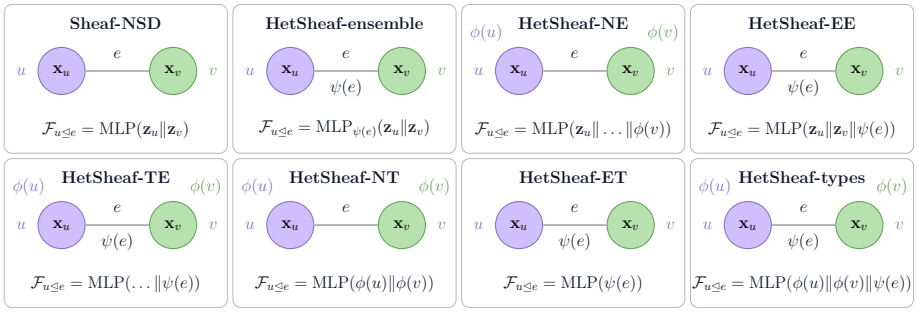


Figure 2: **Heterogeneous Sheaf Predictors.** For an edge $e = (u, v)$, $\phi(u)$ and $\phi(v)$ denote node types, $\psi(e)$ denotes the edge type, x_u, x_v are the raw node features, projected in the shared channel via $\mathbf{z}_* = \mathcal{P}_{\phi(*)}(\mathbf{x}_*)$. $\|$ denotes concatenation, and $\text{MLP}_{\psi(e)}$ denotes a separate MLP for each edge type.

3.2. Heterogeneous Sheaf Predictors and Model Adaptations

Heterogeneous Sheaf Predictors. A key component of HETSHEAF is the construction of restriction maps that explicitly incorporate heterogeneous information. Unlike standard sheaf learning, where maps depend only on node features, HETSHEAF conditions them on projected node features (into the shared channel via $\mathcal{P}_{\phi(*)}$), node types, and edge types, as defined in Eq. 1. Following [6], we consider diagonal, orthogonal, and general restriction-map families (see more details in Appendix C.4.3), and we instantiate the heterogeneous sheaf through a *family of heterogeneous sheaf predictors* that vary along two axes. The former concerns *which type signals are used*: from no type information (Sheaf-NSD, where we only concatenate the projected information relative to the node features $\mathbf{z}_u, \mathbf{z}_v$), to node-only (HETSHEAF-NE, HETSHEAF-NT) or edge-only (HETSHEAF-EE, HETSHEAF-ET) variants, up to predictors conditioned on both node and edge types (HETSHEAF-TE, HETSHEAF-TYPES, HETSHEAF-ENSEMBLE). The latter regards *how type information is routed*: most variants use a shared MLP with type indicators as additional inputs, whereas HETSHEAF-ENSEMBLE allocates a separate predictor to each edge type $\text{MLP}_{\psi(e)}$, increasing expressivity at the cost of more parameters. Figure 2 summarises the predictor family, while full details are deferred to Appendix C.4.4.

Attention Restriction Maps. A further design choice concerns how the raw output of $\mathbf{\Phi}$ is converted into a restriction map. Bodnar et al. [6] formulate an alternative attention-style post-processing step. For the general case, a row-wise softmax yields row-stochastic

restriction maps, and this transformation depends only on the output of $\mathbf{\Phi}$, and we can apply it uniformly to all HetSheaf variants:

$$\mathcal{F}_{u \trianglelefteq e} = \mathbf{I}_d - \mathrm{Softmax}(\mathbf{\Phi}(\mathbf{z}_u, \mathbf{z}_v, \phi(u), \phi(v), \psi(e))) \tag{2}$$

Task-specific Preprocessing. Because different node types may have different input dimensionalities, we use a type-specific linear projection with bias $\mathbf{z}_u = \mathcal{P}_{\phi(u)}(\mathbf{x}_u), \forall u \in \mathcal{V}$ to map all node features into a shared channel dimension. These projections are *learned jointly* with the rest of the model and can be combined with the initial stalk projection of the sheaf encoder to construct the input cochain $\mathbf{z} \in C^0(\mathcal{G}, \mathcal{F})$. Following Lv et al. [45], Zhou et al. [78], we consider multiple feature initialisation strategies, including using all available node features, target-type-only features, and one-hot type indicators. For *node classification*, we apply ℓ_2 normalisation to the final node representation: $\mathbf{o}_u = \frac{\mathbf{h}_u^{(L)}}{\|\mathbf{h}_u^{(L)}\|}$, where $\mathbf{h}_u^{(L)}$ is the final hidden state of node u. For *link prediction*, we follow a JKNet-style strategy [69] and concatenate the normalised hidden states across layers: $\mathbf{o}_u = \|_{l=1}^{L} \frac{\mathbf{h}_u^{(l)}}{\|\mathbf{h}_u^{(l)}\|}$.

Sheaf Models Adaptations. A key advantage of HetSheaf is its model-agnosticity: once a heterogeneous sheaf has been inferred, the resulting typed geometric structure can be processed by a broad class of existing sheaf neural architectures with minimal modifications. To demonstrate this property, we adapt both SheafAN [4] and PolyNSD [7] to the heterogeneous setting. For SheafAN, we make the attention mechanism type-aware by augmenting the attention input with node-type and edge-type information. For PolyNSD, instead, we preserve the original polynomial spectral filtering scheme and apply it directly to the sheaf Laplacian, allowing to perform higher-order diffusion directly over the heterogeneous sheaf without requiring any ad hoc redesign. Full model-specific derivations and implementation details are deferred to Appendices C.4.6 and C.4.7.

4. SheafPool: Stalk-Space Late-token Readout

Graph-level prediction is more delicate in SheafNNs than in GNNs. While relying on the common latent space assumption, for which we can apply a pooling hierarchy, sheaf node embeddings $\mathbf{H}_i \in \mathbb{R}^{d \times C}$ are meaningful only up to node-wise changes of basis.

Table 1: *Pooling ablation on graph classification*, same HetSheaf backbone. Macro/Micro F1, mean ± std over five seeds; best per dataset in **bold**. Attention is the self-attention pooling of Lee et al. [40].

	MUTAG		PROTEINS		ENZYMES		NCI1	
Pooling	Macro F1	Micro F1	Macro F1	Micro F1	Macro F1	Micro F1	Macro F1	Micro F1
Mean	94.88±1.11	94.92±2.26	57.51±1.76	67.04±2.19	19.92±1.77	23.28±1.29	57.88±1.78	57.97±1.29
Max	82.55±1.16	82.41±1.66	67.84±1.82	73.15±2.31	13.59±1.83	15.06±2.22	64.11±1.82	64.02±1.23
Add	74.24±1.40	74.32±1.26	65.68±1.75	67.02±2.22	10.03±1.75	16.61±1.30	57.45±1.75	57.54±1.28
Attention [40]	82.77±2.31	84.94±2.26	69.12±1.84	76.85±2.20	13.74±1.81	16.72±2.24	65.71±1.84	65.61±2.21
SheafPool (ours)	**98.69**±1.31	**98.68**±1.32	**80.55**±1.55	**82.14**±1.42	**62.36**±2.14	**63.30**±1.94	**79.42**±0.87	**79.42**±0.83

Consequently, a valid graph readout must satisfy two requirements simultaneously: it must be permutation-invariant over nodes, and it must be compatible with stalk-wise gauge reparameterisations $\mathbf{H}_i \mapsto \mathbf{G}_i\mathbf{H}_i, \mathbf{G}_i \in \mathbb{G}$, where $\mathbb{G}$ denotes the gauge family induced by $\mathcal{F}$ (e.g. diagonal, orthogonal, or general maps). However, the standard flatten-and-pool procedure on $\mathrm{vec}(\mathbf{H}_i) \in \mathbb{R}^{dC}$ does not satisfy this requirement: they depend on arbitrary stalk coordinates chosen for each stalk, so equivalent sheaf representations may produce different graph embeddings, (as also shown in Table 1). To address this issue, we

introduce SHEAFPOOL, a *Stalk-Space Late-token Readout* graph pooling mechanism that first canonicalises node embeddings (i.e., maps to a common reference stalk frame), then aggregates them directly in stalk space, collapsing the stalk dimension at the very end. The construction preserves the stalk geometry, while remaining applicable to all sheaf families.

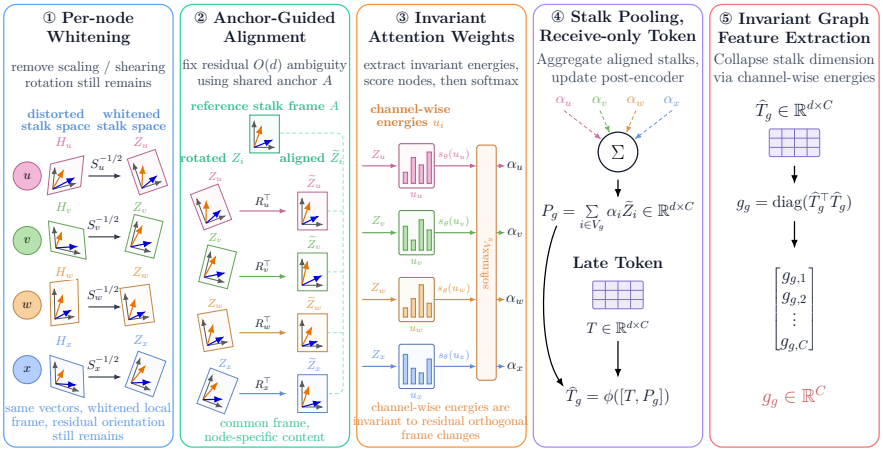


Figure 3: **The SheafPool Architecture.** Our readout processes node stalk embeddings in five steps. *(1) Per-node Whitening* removes GL(d) scaling and shearing. *(2) Anchor Alignment* resolves residual O(d) rotations. *(3) Attention Weights* are computed via stalk energies. *(4) Stalk Pooling* aggregates into a *Late Receive-only Token* without altering message passing. *(5) Feature Extraction* collapses the stalk dimension, yielding a permutation- and gauge-invariant graph embedding.

Stalk-Space Late-Token Formalisation. Formally, we define a graph-level readout as $\mathcal{R} : \{\mathbf{H}_i\}_{i \in V_g} \longmapsto \mathbf{g}_g \in \mathbb{R}^C$, whose output is permutation-invariant over nodes and robust to stalk reparameterisations. The readout consists in five stages that can be summarised (fig. 3) as:

$$\mathbf{H}_i \xrightarrow{\text{whiten}} \mathbf{Z}_i \xrightarrow{\text{align}} \widetilde{\mathbf{Z}}_i \xrightarrow{\text{attn pool}} \mathbf{P}_g \xrightarrow{\text{late token}} \widehat{\mathbf{T}}_g \xrightarrow{\text{energy}} \mathbf{g}_g \tag{3}$$

(1) Per-node whitening. We first remove node-wise scaling and shearing in the stalk dimension by forming $\mathbf{S}_i = \mathbf{H}_i\mathbf{H}_i^\top + \varepsilon \mathbf{I}_d$, and defining $\mathbf{Z}_i = \mathbf{S}_i^{-1/2}\mathbf{H}_i$. This makes the stalk covariance approximately isotropic, thus reducing remaining ambiguity to an orthogonal transform.

(2) Anchor-based alignment. After whitening, different nodes may still be represented in different orthogonal frames. We therefore align each node to a shared learnable anchor $\mathbf{A} \in \mathbb{R}^{d \times C}$ via $\mathbf{R}_i = \arg\min_{\mathbf{R} \in O(d)} \|\mathbf{R}^\top \mathbf{Z}_i - \mathbf{A}\|_F^2$, and set $\widetilde{\mathbf{Z}}_i = \mathbf{R}_i^\top \mathbf{Z}_i$. This places all node embeddings in a common stalk frame before pooling.

(3) Invariant attention weights. To weight node contributions without using basis-dependent coordinates, we score each node through the channel-wise energies of its whitened representation $\mathbf{u}_i = \operatorname{diag}(\mathbf{Z}_i^\top \mathbf{Z}_i) \in \mathbb{R}^C$. Since $\mathbf{u}_i$ is invariant to residual orthogonal reparameterisations, a learnable scorer s_θ can produce graph-wise attention weights $\alpha_i = \dfrac{\exp(s_\theta(\mathbf{u}_i))}{\sum_{j \in V_g} \exp(s_\theta(\mathbf{u}_j))}$.

(4) Stalk-space pooling with a late receive-only token. We then aggregate the aligned embeddings directly in stalk space $\mathbf{P}_g = \sum_{i \in V_g} \alpha_i \widetilde{\mathbf{Z}}_i \in \mathbb{R}^{d \times C}$. To perform readout without inserting a token into the encoder during the diffusion process, which would cause oversmoothing, we use a *late, receive-only token* $\mathbf{T} \in \mathbb{R}^{d \times C}$ updated *only* after pooling $\widehat{\mathbf{T}}_g = \phi([\mathbf{T}, \mathbf{P}_g])$, where ϕ is a small channel-wise MLP with layer normalisation.

(5) Invariant graph feature extraction. Finally, we collapse the stalk dimension through channel-wise energies: $\mathbf{g}_g = \mathrm{diag}(\widehat{\mathbf{T}}_g^\top \widehat{\mathbf{T}}_g) \in \mathbb{R}^C$. The resulting graph embedding is then fed to a standard MLP decoder for graph classification.

Full derivations, invariance arguments, and stability remarks are deferred to Appendix B.

5. Experimental Evaluation

We evaluate HetSheaf on heterogeneous node-, edge-, and graph-level tasks, complemented by ablations on sheaf predictors, restriction-map families, attention mechanisms, SheafPool, and different sheaf architectures. These experiments test four claims: (i) encoding heterogeneity directly in the inferred sheaf improves performance; (ii) conditioning restriction maps on typed information outperforms type-agnostic sheaf learning (Appendix C.3); (iii) HetSheaf is model-agnostic, i.e., can be instantiated across multiple sheaf variants [6, 4, 7]; and (iv) these gains are achieved with favourable parameter efficiency while preserving heterogeneous structure through SheafPool.

Table 2: *Heterogeneous node classification.* Macro/Micro F1, mean ± std over five seeds. Top three by mean: **first**, **second**, **third**.

	ACM		DBLP		IMDB	
	Macro F1	Micro F1	Macro F1	Micro F1	Macro F1	Micro F1
Homogeneous						
GCN	92.25±0.82	91.25±0.84	90.87±1.77	91.68±1.83	58.26±0.36	60.89±0.26
GAT	92.16±1.55	91.95±0.87	92.33±1.13	92.81±0.99	60.23±0.67	**64.06±0.80**
GIN	88.70±2.20	88.78±1.98	**94.83±0.43**	**95.24±0.43**	60.30±0.75	63.96±0.52
SAGE	92.18±1.61	92.07±1.51	90.23±1.31	90.69±1.26	**60.60±1.75**	63.73±0.63
Heterogeneous						
R-GCN	**93.27±0.82**	**93.19±0.86**	**94.30±0.23**	**94.70±0.20**	59.67±0.21	62.36±0.20
HAN	**93.67±0.26**	**93.61±0.28**	94.07±0.41	94.47±0.38	56.85±2.91	61.68±1.08
HGT	92.38±0.51	92.28±0.55	92.82±1.14	93.37±1.08	56.59±0.89	60.25±0.41
Sheaf-NSD	92.07±2.09	91.90±2.09	92.91±1.21	93.15±1.18	**61.94±0.34**	**65.10±0.20**
HetSheaf (ours)	**94.30±0.24**	**94.22±0.24**	**94.93±0.36**	**95.27±0.34**	**64.51±0.68**	**66.57±0.17**

Datasets, Baselines and Evaluation Protocol. We evaluate HetSheaf on a broad suite of heterogeneous benchmarks spanning *node classification*, *link prediction*, and *graph classification*, including citation, knowledge, user–item, biological, molecular, biomedical, and networking graphs [45, 78, 66, 37, 2, 48, 35, 67, 47, 41, 19, 20]. We compare against representative homogeneous (GCN, GAT, GIN, GraphSAGE), heterogeneous (R-GCN, HAN, HGT), and sheaf-based baselines, with additional comparisons to recent heterogeneous models in Appendix C.3. We follow the standard evaluation protocol of each benchmark, and report *Macro/Micro-F1* for classification and *AUPR/AUROC* for link prediction. Hyperparameters are selected through 150-run sweeps per model type, after which the best validation configuration is re-trained over five random seeds (42–46); results are reported as mean ± standard deviation. Dataset details, preprocessing, hyperparameter spaces, hardware, and additional comparisons are provided in Appendices C.1, C.3 and C.4.

Experimental Results. Tables 2 to 4 report the main results across tasks, where HetSheaf emerges as the most consistently strong method. It achieves the best performance on all three node-classification benchmarks, ranks first on the two largest link-prediction datasets, and remains highly competitive on the remaining relation-prediction tasks. On graph classification, where the challenge is no longer only to learn typed local interactions but also to compress them into a stable graph-level representation, HetSheaf again performs strongly. Appendix C.3 situates HetSheaf within a broader landscape of heterogeneous GNNs from the *recent literature*, including a matched-protocol re-run of HAGNN, SeHGNN,

and SlotGAT under the same splits and metric code (Table 23), while Appendix C.2.2 and C.2.3 extend the study to biomedical networks and networking tasks, where HETSHEAF and Sheaf-NSD consistently occupy the top ranks. Appendix C.2 then dissects the source of these gains through a detailed analysis of heterogeneous sheaf predictors and restriction-map families, including extensions of HETSHEAF to additional sheaf architectures such as POLYNSD and SHEAFAN. Further sensitivity analyses over the stalk dimension and the number of layers are reported in Appendices C.2.7 and C.2.8.

Table 3: *Heterogeneous link prediction.* Binary AUPR/AUROC; mean ± std over five seeds. Top three by mean: **first**, **second**, **third**.

	LastFM		MovieLens		PubMed		Amazon		YouTube	
	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC
Homogeneous										
GCN	97.05±0.25	96.95±0.17	**99.67**±0.04	**99.62**±0.04	96.03±0.11	92.88±0.28	92.44±0.28	89.61±0.27	89.44±0.18	83.25±0.17
GAT	63.03±0.22	51.11±0.50	97.50±0.22	97.83±0.18	96.06±0.14	93.23±0.27	88.59±0.19	86.11±0.25	66.79±0.39	58.31±0.28
GIN	**97.98**±0.19	**97.73**±0.30	99.58±0.14	99.54±0.04	96.11±0.07	92.84±0.12	92.04±0.28	89.12±0.29	88.62±0.58	82.92±0.49
SAGE	95.07±1.76	92.70±2.89	99.21±0.09	99.12±0.08	95.53±0.52	88.66±0.97	90.82±0.39	87.82±0.32	84.52±0.09	80.22±0.32
Heterogeneous										
R-GCN	**97.84**±0.27	**97.88**±0.14	99.37±0.10	99.34±0.09	95.02±0.09	93.09±0.09	93.28±0.11	91.48±0.14	**91.58**±0.21	**88.38**±0.14
HAN	88.24±2.41	83.34±2.12	63.58±0.06	52.11±0.13	95.79±0.37	93.64±0.60	**95.19**±0.68	**93.59**±0.87	**93.48**±0.26	**90.14**±0.36
HGT	86.01±0.80	74.55±0.78	77.61±1.46	73.90±1.50	**97.56**±0.30	**96.43**±0.52	**96.13**±0.26	**95.65**±0.16	**94.96**±0.58	**92.55**±0.45
Sheaf-NSD	97.16±0.16	96.58±0.17	**99.69**±0.04	**99.58**±0.04	**96.41**±0.07	**93.69**±0.16	92.58±0.24	90.02±0.25	89.36±0.40	83.22±0.30
HetSheaf (ours)	**98.51**±0.34	**98.33**±0.35	**99.71**±0.04	**99.64**±0.04	96.35±0.59	**93.82**±0.68	**93.72**±0.32	**91.71**±0.37	91.25±0.59	87.32±0.48

Table 4: *Heterogeneous graph classification.* Macro/Micro F1; mean ± std over five seeds. Top three by mean: **first**, **second**, **third**.

	MUTAG		PROTEINS		ENZYMES		NCI1	
	Macro F1	Micro F1	Macro F1	Micro F1	Macro F1	Micro F1	Macro F1	Micro F1
Homogeneous								
GCN	89.56±1.04	90.00±1.35	76.82±2.45	80.34±2.47	12.71±1.92	28.30±2.71	64.13±2.12	64.94±2.31
GAT	89.56±1.02	90.00±1.32	75.76±1.51	78.54±1.75	14.12±1.29	16.60±2.27	64.19±2.44	64.90±2.07
GIN	**94.88**±0.72	**95.00**±0.78	70.64±1.38	72.32±1.32	20.10±2.46	28.33±2.43	64.17±2.18	64.91±2.86
SAGE	89.56±1.04	90.00±1.35	74.59±1.56	78.55±1.35	8.55±1.80	18.33±2.26	63.64±0.85	65.41±1.09
Heterogeneous								
R-GCN	84.97±1.86	85.00±1.45	75.71±2.73	78.54±0.91	10.35±2.07	16.67±2.43	64.83±2.11	65.60±3.62
HAN	82.90±4.25	85.00±1.16	**79.01**±1.94	**83.02**±1.04	**46.13**±2.28	**44.97**±2.59	**77.94**±1.59	**78.15**±1.56
HGT	84.64±1.44	85.00±1.40	**77.54**±1.60	**81.26**±1.35	**48.30**±1.83	**48.33**±2.87	**79.29**±2.79	**79.31**±3.73
Sheaf-NSD	**98.23**±1.77	**98.26**±1.74	73.58±1.75	75.84±1.65	33.66±2.08	34.97±1.90	72.32±1.87	72.32±1.82
HetSheaf (ours)	**98.69**±1.31	**98.68**±1.32	**80.55**±1.55	**82.14**±1.42	**62.36**±2.14	**63.30**±1.94	**79.42**±0.87	**79.42**±0.83

Finally, Appendices B and C.4 provides a complementary systems-level view, including runtime and parameter-count comparisons, showing that the empirical gains are obtained with modest computational overhead. Taken together, the results support three conclusions. First, explicitly encoding heterogeneity in the inferred sheaf yields systematic gains over both type-agnostic sheaf models and strong heterogeneous GNN baselines. Moreover these results extend to graph-level inductive reasoning, providing empirical support for the proposed SHEAFPOOL. Third, the advantage of HETSHEAF is robust across different dataset regimes.

6. Conclusion

We presented HETSHEAF, a model-agnostic framework that captures graph heterogeneity through cellular sheaves to enable principled type-specific interactions. By explicitly conditioning restriction maps on node and edge types, HETSHEAF avoids the need for increasingly specialized architectures. To enable graph-level learning, we introduced SHEAFPOOL, a stalk-space readout that makes sheaf-network graph classification effective, improving Macro F1 by up to 42 percentage points over mean pooling. Empirically, HETSHEAF achieves state-of-the-art results on the Heterogeneous Graph Benchmark, reaching up to 95.39% Macro F1 in node classification and 99.62% in link prediction. It outperforms specialized heterogeneous GNNs by up to 2 percentage points while being up to 10× more parameter-efficient. Despite the limitations discussed in Appendix D, HETSHEAF establishes a robust, highly efficient foundation for heterogeneous sheaf learning that naturally extends to hypergraphs and broader topological domains.

Acknowledgments

This work was supported in part by NSF under Grants CNS-2312875 and OAC-2530896, DARPA under Cooperative Agreement D25AC00374-00, ONR under Award N00014-23-1-2221, and AFOSR under Award FA9550-23-1-0261.

References

[1] Inductive Representation Learning on Large Graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964. URL https://dl.acm.org/doi/10.5555/3294771.3294869.

[2] Heterogeneous Network Representation Learning: A Unified Framework With Survey and Benchmark. *IEEE Transactions on Knowledge and Data Engineering*, 34(10): 4854–4873, October 2022. ISSN 1041-4347. doi: 10.1109/TKDE.2020.3045924. URL https://doi.org/10.1109/TKDE.2020.3045924.

[3] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=i80OPhOCVH2.

[4] 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*, November 2022. URL https://openreview.net/forum?id=LIDvgVjpkZr.

[5] Federico Barbero, Cristian Bodnar, Haitz Sáez de Ocáriz Borde, Michael Bronstein, Petar Veličković, and Pietro Liò. Sheaf neural networks with connection laplacians. In Alexander Cloninger, Timothy Doster, Tegan Emerson, Manohar Kaul, Ira Ktena, Henry Kvinge, Nina Miolane, Bastian Rieck, Sarah Tymochko, and Guy Wolf, editors, *Proceedings of Topological, Algebraic, and Geometric Learning Workshops 2022*, volume 196 of *Proceedings of Machine Learning Research*, pages 28–36. PMLR, 25 Feb–22 Jul 2022. URL https://proceedings.mlr.press/v196/barbero22a.html.

[6] Cristian Bodnar, Francesco Di Giovanni, Benjamin Paul Chamberlain, Pietro Lio, and Michael M. Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in GNNs. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=vbPsD-BhOZ.

[7] Alessio Borgi, Fabrizio Silvestri, and Pietro Liò. Polynomial neural sheaf diffusion: A spectral filtering approach on cellular sheaves, 2026. URL https://arxiv.org/abs/2512.00242.

[8] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations*, 2014. URL https://arxiv.org/abs/1312.6203.

[9] Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*. URL https://grlplus.github.io/papers/79.pdf.

[10] Ferran Hernandez Caralt, Guillermo Bernardez, Iulia Duta, Eduard Alarcon, and Pietro Lio. Joint diffusion processes as an inductive bias in sheaf neural networks. In *ICML 2024 Workshop on Geometry-grounded Representation Learning and Generative Modeling*, 2024. URL https://openreview.net/forum?id=0xkPQvpXIS.

[11] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. Representation learning for attributed multiplex heterogeneous network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 1358–1368, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330964. URL https://doi.org/10.1145/3292500.3330964.

[12] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Velickovic. Principal neighbourhood aggregation for graph nets. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546. URL https://dl.acm.org/doi/10.5555/3495724.3496836.

[13] Justin Michael Curry. *Sheaves, Cosheaves and Applications*. PhD thesis, University of Pennsylvania, 2014.

[14] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 3844–3852, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819. URL https://dl.acm.org/doi/10.5555/3157382.3157527.

[15] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 135–144, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348874. doi: 10.1145/3097983.3098036. URL https://doi.org/10.1145/3097983.3098036.

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

[17] Iulia Duta, Giulia Cassarà, Fabrizio Silvestri, and Pietro Lio. Sheaf hypergraph networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=NvcVXzJvhX.

[18] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The World Wide Web Conference*, WWW '19, page 417–426, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366748. doi: 10.1145/3308558.3313488. URL https://doi.org/10.1145/3308558.3313488.

[19] Miquel Ferriol-Galmés, Krzysztof Rusek, José Suárez-Varela, Shihan Xiao, Xiang Shi, Xiangle Cheng, Bo Wu, Pere Barlet-Ros, and Albert Cabellos-Aparicio. Routenet-erlang: A graph neural network for network performance evaluation. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, page 2018–2027. IEEE Press, 2022. doi: 10.1109/INFOCOM48880.2022.9796944. URL https://doi.org/10.1109/INFOCOM48880.2022.9796944.

[20] Miquel Ferriol-Galmés, Jordi Paillisse, José Suárez-Varela, Krzysztof Rusek, Shihan Xiao, Xiang Shi, Xiangle Cheng, Pere Barlet-Ros, and Albert Cabellos-Aparicio. Routenet-fermi: Network modeling with graph neural networks. *IEEE/ACM Trans. Netw.*, 31 (6):3080–3095, May 2023. ISSN 1063-6692. doi: 10.1109/TNET.2023.3269983. URL https://doi.org/10.1109/TNET.2023.3269983.

[21] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. URL https://rlgm.github.io/papers/2.pdf.

[22] Matthias Fey, Jinu Sunil, Akihiro Nitta, Rishi Puri, Manan Shah, Blaž Stojanovič, Ramona Bendias, Alexandria Barghi, Vid Kocijan, Zecheng Zhang, Xinwei He, Jan Eric Lenssen, and Jure Leskovec. Pyg 2.0: Scalable learning on real world graphs. In *Temporal Graph Learning Workshop @ KDD 2025*, 2025. URL https://openreview.net/forum?id=DHHLkQvWqs.

[23] Stefano Fiorini, Hakan Aktas, Iulia Duta, Pietro Morerio, Alessio Del Bue, Pietro Lio, and Stefano Coniglio. Sheaves reloaded: A direction awakening. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=iDiiETH7Qv.

[24] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. WWW '20, page 2331–2341, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370233. doi: 10.1145/3366423.3380297. URL https://doi.org/10.1145/3366423.3380297.

[25] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1263–1272. JMLR.org, 2017. URL https://dl.acm.org/doi/10.5555/3305381.3305512.

[26] C. Goller and A. 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 vol.1, 1996. doi: 10.1109/ICNN.1996.548916. URL https://doi.org/10.1109/ICNN.1996.548916.

[27] M. Gori, G. Monfardini, and F. 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 vol. 2, 2005. doi: 10.1109/IJCNN.2005.1555942. URL https://doi.org/10.1109/IJCNN.2005.1555942.

[28] Daniele Grattarola, Daniele Zambon, Filippo Maria Bianchi, and Cesare Alippi. Understanding pooling in graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 35(2):2708–2718, 2024. doi: 10.1109/TNNLS.2022.3190922. URL https://doi.org/10.1109/TNNLS.2022.3190922.

[29] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. 2019. doi: 10.1609/aaai.v33i01.3301922. URL https://doi.org/10.1609/aaai.v33i01.3301922.

[30] Mustafa Hajij, Lennart Bastian, Sarah Osentoski, Hardik Kabaria, John L. Davenport, Sheik Dawood, Balaji Cherukuri, Joseph G. Kocheemoolayil, Nastaran Shahmansouri, Adrian Lew, Theodore Papamarkou, and Tolga Birdal. Copresheaf topological neural networks: A generalized deep learning framework. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=3G56xClPYg.

[31] William L. Hamilton. *Graph Representation Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Springer International Publishing, Cham, 2020. ISBN 978-3-031-00460-5. URL https://doi.org/10.1007/978-3-031-01588-5.

[32] Jakob Hansen and Thomas Gebhart. Sheaf neural networks. In *NeurIPS Workshop on TDA & Beyond*, 2020. URL https://openreview.net/forum?id=GgcgIJsT8HD.

[33] Jakob Hansen and Robert Ghrist. Toward a spectral theory of cellular sheaves. *Journal of Applied and Computational Topology*, 3(4):315–358, December 2019. ISSN 2367-1734. URL https://doi.org/10.1007/s41468-019-00038-7.

[34] Yu He, Cristian Bodnar, and Pietro Liò. Sheaf-based positional encodings for graph neural networks. In Sophia Sanborn, Christian Shewmake, Simone Azeglio, and Nina Miolane, editors, *Proceedings of the 2nd NeurIPS Workshop on Symmetry and Geometry in Neural Representations*, volume 228 of *Proceedings of Machine Learning Research*, pages 1–18. PMLR, 16 Dec 2024. URL https://proceedings.mlr.press/v228/he24a.html.

[35] Junwei Hu, Michael Bewong, Selasi Kwashie, Wen Zhang, Hong-Yu Zhang, and Zaiwen Feng. Heterogeneous graph neural networks for link prediction in biomedical networks. *Bioinformatics Advances*, 5(1):vbaf187, 08 2025. ISSN 2635-0041. doi: 10.1093/bioadv/vbaf187. URL https://doi.org/10.1093/bioadv/vbaf187.

[36] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 1857–1867, New York, NY, USA, 2020. Association for Computing Machinery.

ISBN 9781450379984. doi: 10.1145/3394486.3403237. URL https://doi.org/10.1145/3394486.3403237.

[37] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, WWW '20, page 2704–2710, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370233. doi: 10.1145/3366423.3380027. URL https://doi.org/10.1145/3366423.3380027.

[38] Chenqing Hua, Guillaume Rabusseau, and Jian Tang. High-order pooling for graph neural networks with tensor decomposition. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088. URL https://dl.acm.org/doi/10.5555/3600270.3600706.

[39] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

[40] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3734–3743. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/lee19c.html.

[41] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, page 177–187, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 159593135X. doi: 10.1145/1081870.1081893. URL https://doi.org/10.1145/1081870.1081893.

[42] Chang Li and Dan Goldwasser. Encoding social information with graph convolutional networks forPolitical perspective detection in news media. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2594–2604, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1247. URL https://aclanthology.org/P19-1247/.

[43] Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. Gated graph sequence neural networks. In *Proceedings of ICLR'16*, 2016. URL http://arxiv.org/abs/1511.05493.

[44] Weihong Lin, Zhaoliang Chen, Yuhong Chen, and Shiping Wang. Heterogeneous graph neural network with adaptive relation reconstruction. *Neural Networks*, 187: 107313, 2025. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2025.107313. URL https://www.sciencedirect.com/science/article/pii/S0893608025001923.

[45] Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. Are we really making much

progress? Revisiting, benchmarking and refining heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pages 1150–1160, New York, NY, USA, August 2021. Association for Computing Machinery. ISBN 978-1-4503-8332-5. doi: 10.1145/3447548.3467350. URL https://doi.org/10.1145/3447548.3467350.

[46] Qiheng Mao, Zemin Liu, Chenghao Liu, and Jianling Sun. Hinormer: Representation learning on heterogeneous information networks with graph transformer. In *Proceedings of the ACM Web Conference 2023*, WWW '23, page 599–610, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394161. doi: 10.1145/3543507.3583493. URL https://doi.org/10.1145/3543507.3583493.

[47] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, page 165–172, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450324090. doi: 10.1145/2507157.2507163. URL https://doi.org/10.1145/2507157.2507163.

[48] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. URL www.graphlearning.io.

[49] Emanuele Mule, Stefano Fiorini, Antonio Purificato, Federico Siciliano, Stefano Coniglio, and Fabrizio Silvestri. Directional sheaf hypergraph networks: Unifying learning on directed and undirected hypergraphs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=d1hQkcDy7c.

[50] Bao Nguyen, Lorenzo Sani, Xinchi Qiu, Pietro Liò, and Nicholas D. Lane. Sheaf hypernetworks for personalized federated learning, 2024. URL https://doi.org/10.48550/arXiv.2405.20882.

[51] Anton Obukhov. Efficient householder transformation in pytorch, 2021. URL www.github.com/toshas/torch-householder. Version: 1.0.1, DOI: 10.5281/zenodo.5068733.

[52] Gabriele Onorato. Bayesian optimization for hyperparameters tuning in neural networks, 2024. URL https://arxiv.org/abs/2410.21886.

[53] Antonio Purificato, Giulia Cassarà, Federico Siciliano, Pietro Liò, and Fabrizio Silvestri. Sheaf4rec: Sheaf neural networks for graph-based recommender systems. *ACM Trans. Recomm. Syst.*, 4(2), November 2025. doi: 10.1145/3742898. URL https://doi.org/10.1145/3742898.

[54] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. Deepinf: Social influence prediction with deep learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*,

KDD '18, page 2110–2119, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355520. doi: 10.1145/3219819.3220077. URL https://doi.org/10.1145/3219819.3220077.

[55] André Ribeiro, Ana Luiza Tenorio, Juan Belieni, Amauri H Souza, and Diego Mesquita. Cooperative sheaf neural networks. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=AHpexliCTM.

[56] 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, 2009. doi: 10.1109/TNN.2008.2005605. URL https://doi.org/10.1109/TNN.2008.2005605.

[57] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling Relational Data with Graph Convolutional Networks. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam, editors, *The Semantic Web*, pages 593–607, Cham, 2018. Springer International Publishing. URL https://doi.org/10.1007/978-3-319-93417-4_38.

[58] Vasileios Sevetlidis and George Pavlidis. Gauge-invariant representation holonomy. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=czJqKToDGq.

[59] Allen D. Shepard. *A Cellular Description of the Derived Category of a Stratified Space*. PhD thesis, Brown University, 1985.

[60] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf.

[61] Yizhou Sun and Jiawei Han. *Mining Heterogeneous Information Networks: Principles and Methodologies*. Synthesis Lectures on Data Mining and Knowledge Discovery. Springer International Publishing, Cham, 2012. ISBN 978-3-031-01902-9. URL https://doi.org/10.1007/978-3-031-01902-9.

[62] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. PathSim: meta path-based top-K similarity search in heterogeneous information networks. *Proc. VLDB Endow.*, 4(11):992–1003, August 2011. ISSN 2150-8097. doi: 10.14778/3402707.3402736. URL https://doi.org/10.14778/3402707.3402736.

[63] Damian Szklarczyk, Annika L. Gable, Katerina C. Nastou, David Lyon, Rebecca Kirsch, Sampo Pyysalo, Nadezhda T. Doncheva, Marc Legeay, Tao Fang, Peer Bork, Lars J. Jensen, and Christian von Mering. The STRING database in 2021: customizable protein–protein networks, and functional characterization of user-uploaded gene/measurement

sets. *Nucleic Acids Research*, 49(D1):D605–D612, 2021. URL https://doi.org/10.1093/nar/gkaa1074.

[64] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

[65] Petar Veličković, Wiliam Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep Graph Infomax. 2019. URL https://openreview.net/forum?id=rklz9iAcKQ.

[66] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous Graph Attention Network. In *The World Wide Web Conference*, WWW '19, page 2022–2032, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366748. doi: 10.1145/3308558.3313562. URL https://doi.org/10.1145/3308558.3313562.

[67] Xiaoqi Wang, Yaning Yang, Kenli Li, Wentao Li, Fei Li, and Shaoliang Peng. Bioerp: biomedical heterogeneous network-based self-supervised representation learning approach for entity relationship predictions. *Bioinformatics*, 2021. doi: 10.1093/bioinformatics/btab565. URL https://doi.org/10.1093/bioinformatics/btab565.

[68] David S. Wishart, Yannick D. Feunang, An C. Guo, Elvis J. Lo, Ana Marcu, Jason R. Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, Nazanin Assempour, Ithayavani Iynkkaran, Yifeng Liu, Adam Maciejewski, Nicola Gale, Alex Wilson, Lucy Chin, Ryan Cummings, Diana Le, Allison Pon, Craig Knox, and Michael Wilson. DrugBank 5.0: a major update to the DrugBank database for 2018. *Nucleic Acids Research*, 46(D1):D1074–D1082, 2018.

[69] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation Learning on Graphs with Jumping Knowledge Networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5453–5462. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/xu18c.html.

[70] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

[71] Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/pdf/1412.6575.

[72] Carl Yang, Aditya Pal, Andrew Zhai, Nikil Pancha, Jiawei Han, Charles Rosenberg, and Jure Leskovec. MultiSage: Empowering GCN with Contextualized Multi-Embeddings on Web-Scale Multipartite Networks. In *Proceedings of the 26th ACM SIGKDD*

International Conference on Knowledge Discovery & Data Mining, KDD '20, pages 2434–2443, New York, NY, USA, August 2020. Association for Computing Machinery. ISBN 978-1-4503-7998-4. doi: 10.1145/3394486.3403293. URL https://doi.org/10.1145/3394486.3403293.

[73] Xiaocheng Yang, Mingyu Yan, Shirui Pan, Xiaochun Ye, and Dongrui Fan. Simple and efficient heterogeneous graph neural network. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. ISBN 978-1-57735-880-0. doi: 10.1609/aaai.v37i9.26283. URL https://doi.org/10.1609/aaai.v37i9.26283.

[74] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph Transformer Networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://dl.acm.org/doi/10.5555/3454287.3455360.

[75] Olga Zaghen, Antonio Longa, Steve Azzolin, Lev Telyatnikov, Andrea Passerini, and Pietro Lio. Sheaf diffusion goes nonlinear: Enhancing GNNs with adaptive sheaf laplacians. In *ICML 2024 Workshop on Geometry-grounded Representation Learning and Generative Modeling*, 2024. URL https://openreview.net/forum?id=MGQtGV5gPO.

[76] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. Heterogeneous Graph Neural Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 793–803, New York, NY, USA, July 2019. Association for Computing Machinery. ISBN 978-1-4503-6201-6. doi: 10.1145/3292500.3330961. URL https://doi.org/10.1145/3292500.3330961.

[77] Tianyu Zhao, Cheng Yang, Yibo Li, Quan Gan, Zhenyi Wang, Fengqi Liang, Huan Zhao, Yingxia Shao, Xiao Wang, and Chuan Shi. Space4HGNN: A Novel, Modularized and Reproducible Platform to Evaluate Heterogeneous Graph Neural Network. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, pages 2776–2789, New York, NY, USA, July 2022. Association for Computing Machinery. ISBN 978-1-4503-8732-3. doi: 10.1145/3477495.3531720. URL https://doi.org/10.1145/3477495.3531720.

[78] Ziang Zhou, Jieming Shi, Renchi Yang, Yuanhang Zou, and Qing Li. SlotGAT: Slot-based Message Passing for Heterogeneous Graphs. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023. URL https://dl.acm.org/doi/10.5555/3618408.3620206.

[79] Guanghui Zhu, Zhennan Zhu, Hongyang Chen, Chunfeng Yuan, and Yihua Huang. Hagnn: Hybrid aggregation for heterogeneous graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 36(8):14536–14550, 2025. doi: 10.1109/TNNLS.2024.3519427. URL https://doi.org/TNNLS.2024.3519427.

[80] Shichao Zhu, Chuan Zhou, Shirui Pan, Xingquan Zhu, and Bin Wang. Relation Structure-Aware Heterogeneous Graph Neural Network. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 1534–1539, November 2019. doi: 10.1109/ICDM.2019.00203. URL https://doi.org/10.1109/ICDM.2019.00203.

Heterogeneous Sheaf Neural Networks

Supplementary Material

Contents

A Extended Related Work **21**
A.1 Homogeneous and Heterogeneous Graph Neural Networks 21
A.2 Sheaf Neural Networks 21
A.3 Cellular Sheaves on Graphs 22

B Stalk-Space Late-token Readout: Formalisation and Properties **22**
B.1 Problem Setting and Gauge Quotient Viewpoint 23
B.2 Flatten-and-Pool fails for Sheaf Neural Networks 24
B.3 Design Principle: Canonicalise and Pool 25
B.4 Proposed Stalk-Space Late-token Readout 26
B.5 Classification Token in Vision Transformers Comparison 31
B.6 Injective Stalk-Space Readout on the Canonicalised Descriptor Space 31

C Additional Experimental, Ablation, and Implementation Details **36**
C.1 Experimental Protocol, Hyperparameters, and Datasets 36
C.2 Additional Results and Model Analysis 37
C.3 External Comparisons with Additional Heterogeneous GNNs 52
C.4 Implementation Details 55

D Limitations **62**
D.1 Broader Impact 62

Appendix A. Extended Related Work

This appendix provides the extended version of the discussion in Section 2.

A.1. Homogeneous and Heterogeneous Graph Neural Networks

A wide range of GNNs have been proposed for *homogeneous* graphs [31, 39, 43, 64, 14, 12]. Although these models can be directly applied to heterogeneous graphs by ignoring node and edge types, they are not explicitly designed to capture the type-specific structure of heterogeneous relational data. While in homogeneous graphs all nodes/edges encode the same type of data, in *Heterogeneous GNNs* nodes and edges can represent different types of information. Several HGNNs have been proposed: a specific typology of them uses *meta-paths* [62, 61, 15], i.e., pre-defined node- or edge-type patterns that aim to capture important semantic information. HAN [66] employs a hierarchical attention mechanism to capture node-level importance among nodes and semantic-level importance across meta-paths. MAGNN [24] combines several meta-path encoders to aggregate all information along the meta-path. Both HAN and MAGNN require meta-paths to be generated manually in advance. Graph Transformer Networks (GTNs) [74] automate this process by learning meta-paths with graph transformation layers. However, Meta-paths are computationally expensive, making them infeasible for graphs with many distinct edge types. Instead, the graph may be treated as a relational graph. RSHN [80] constructs a coarsened line graph to generate edge features, and then applies a Message Passing Neural Network (MPNN) [25] to propagate node and edge features. R-GCN [57] splits the heterogeneous graph into the subgraph induced by each edge type, then applies a standard GCN to each subgraph. HGT [76] is a graph transformer that can process large heterogeneous graphs using subgraph sampling. HetGNN [76] uses random walks to sample strongly correlated heterogeneous neighbours and then applies an RNN for representation learning. SlotGAT [78] uses a slot-based message passing approach with an attention mechanism for aggregation. SimpleHGNN [45], applies a series of additional techniques to lift GATs to heterogeneous domains and Space4HGNN [77] defines a unified design space of modular components for heterogeneous GNNs and allows exhaustive evaluation of many combinations of techniques. More recently, HAGNN [79] combines meta-path-based intra-type aggregation with meta-path-free inter-type aggregation through a fused meta-path graph, while HGNN-AR^2 [44] adaptively reconstructs relations to address same-category connection deficiencies in meta-path methods. Instead of introducing a new architecture for heterogeneous graphs, we propose shifting from representing heterogeneous relational data as graphs to cellular sheaves, which preserve all type-specific information and enable powerful processing with sheaf-specific tools.

A.2. Sheaf Neural Networks

Standard GNNs are known to suffer from limitations such as oversmoothing [9], oversquashing [3] and degradation of performances in heterophilic settings, i.e., when adjacent nodes belong to different classes or carry contrasting attributes. Cellular sheaf theory [59, 13] equips graphs with local feature spaces, or *stalks*, together with linear restriction maps along edges, thereby providing a principled geometric framework for modelling and alleviating these pathologies. Sheaf Neural Networks were originally introduced using a hand-crafted

sheaf with a single dimensionality in [32], and further improved by allowing the sheaf to be learned using a learnable parametric function in Bodnar et al. [6]. Since then, sheaf-based graph learning has been extended in several directions: attention mechanisms over sheaves [4], connection Laplacians learned during preprocessing [5], sheaf-based positional encodings [34], nonlinear diffusion processes [75], joint diffusion biases [10], polynomial spectral filtering [7], and directional extensions [23]. Beyond standard graphs, the sheaf framework has also been extended to hypergraphs [17, 49], recommendation systems [53], federated learning [50], cooperative sheaf dynamics [55], and more general topological constructions via copresheaves [30]. Existing sheaf neural networks have primarily focused on node-level tasks, leaving graph-level prediction largely unexplored. Indeed, graph-level readout in sheaf networks is more subtle than standard aggregation, since a valid pooling operator must not only be permutation-invariant over nodes, but also invariant to node-wise stalk reparameterisations. Existing graph pooling methods [28, 38, 43, 40] satisfy the former requirement but not the latter, and therefore cannot be directly applied in a principled way to sheaf representations.

A.3. Cellular Sheaves on Graphs

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a finite undirected graph endowed with an arbitrary orientation on its edges. A *cellular sheaf* $\mathcal{F}$ on $\mathcal{G}$ assigns a finite-dimensional vector space, or *stalk*, $\mathcal{F}(u)$ to each node $u \in \mathcal{V}$ and $\mathcal{F}(e)$ to each edge $e \in \mathcal{E}$, together with linear *restriction maps* $\mathcal{F}_{u \trianglelefteq e} : \mathcal{F}(u) \to \mathcal{F}(e)$ for every incident node-edge pair $u \trianglelefteq e$. By stacking the node and edge stalks, we obtain the spaces of 0- and 1-cochains, $C^0(\mathcal{G}, \mathcal{F}) \coloneqq \bigoplus_{u \in \mathcal{V}} \mathcal{F}(u), \quad C^1(\mathcal{G}, \mathcal{F}) \coloneqq \bigoplus_{e \in \mathcal{E}} \mathcal{F}(e)$. Given an oriented edge $e = (u \to v)$, these spaces induce the *coboundary map* $\delta : C^0(\mathcal{G}, \mathcal{F}) \to C^1(\mathcal{G}, \mathcal{F})$, defined edgewise by $\delta(\mathbf{x})_e \coloneqq \mathcal{F}_{v \trianglelefteq e}\mathbf{x}_v - \mathcal{F}_{u \trianglelefteq e}\mathbf{x}_u$. The coboundary measures disagreement between neighbouring node assignments after they are mapped into the shared edge space [33]. The corresponding *sheaf Laplacian* $L_{\mathcal{F}} \coloneqq \delta^\top \delta$ and its normalised version is then defined node-wise, as:

$$\Delta_{\mathcal{F}}(\mathbf{x})_u \coloneqq D^{-1/2} \left[\sum_{u,v \trianglelefteq e} \mathcal{F}_{u \trianglelefteq e}^\top (\mathcal{F}_{u \trianglelefteq e}\mathbf{x}_u - \mathcal{F}_{v \trianglelefteq e}\mathbf{x}_v) \right] D^{-1/2} \tag{4}$$

where D is the block diagonal of $L_{\mathcal{F}}$. As for the other sheaf neural networks, we assume all stalks have the same dimension d, i.e. $\mathcal{F}(u) = \mathbb{R}^d$ and $\mathcal{F}(e) = \mathbb{R}^d$, so that $L_{\mathcal{F}} \in \mathbb{R}^{dn \times dn}$, with $n = |\mathcal{V}|$. In the special case where all stalks are $\mathbb{R}$, and all restriction maps are the identity, the sheaf Laplacian reduces to the standard graph Laplacian. Generalizing the 0-cochains to f feature channels results in a feature matrix $\mathbf{X} \in \mathbb{R}^{nd \times f}$, enabling *Neural Sheaf Diffusion (NSD)* [6], to learn restriction maps via a parametric function $\mathcal{F}_{u \trianglelefteq e := (u,v)} = \mathbf{\Phi}(\mathbf{x}_u, \mathbf{x}_v) = \mathrm{MLP}(\mathbf{x}_u \| \mathbf{x}_v)$, with $\|$ denoting vector concatenation.

Appendix B. Stalk-Space Late-token Readout: Formalisation and Properties

This appendix provides a formal treatment of SheafPool, our graph-level readout for sheaf neural networks. We proceed in four steps. First, we formalise graph classification in the presence of stalk gauges and explain why standard flatten-and-pool readouts are not

well-defined in this setting. Second, we derive the proposed *Stalk-Space Late-token Readout* and prove its permutation invariance and gauge compatibility under natural identifiability assumptions. Third, we compare the proposed *late, receive-only token* to the *classification token* used in Vision Transformers (ViTs), clarifying why the latter cannot be imported naively into sheaf encoders. Finally, following the multiset perspective of expressive GNN theory, we state and prove an *injective readout theorem* for the canonicalised stalk descriptor space. This theorem formalises that injective graph-level pooling is possible in principle after gauge canonicalisation, and our practical late-token energy head should be viewed as a structured, low-overhead instantiation of this broader family.

B.1. Problem Setting and Gauge Quotient Viewpoint

Let $G_g = (V_g, E_g)$ be a graph and let a sheaf neural encoder produce node embeddings:

$$\mathbf{H}_i \in \mathbb{R}^{d\times C} \quad \text{for} \quad i \in V_g \tag{5}$$

where d is the stalk dimension and C is the channel dimension. For graph classification, the goal is to construct a readout that summarises the graph into a single graph representation:

$$\mathcal{R} : \{\mathbf{H}_i\}_{i\in V_g} \mapsto \mathbf{g}_g \in \mathbb{R}^C \tag{6}$$

In ordinary GNNs, all node embeddings live in the same latent space, usually $\mathbb{R}^C$. To obtain a graph-level representation, we can simply pool the set of node embeddings $\mathbf{h}_i, i \in V_g$ using a permutation-invariant operation such as sum, mean, or max pooling. In a sheaf model, the situation is different: the hidden state of node i lives in its own local vector space, called the stalk $\mathcal{F}(i)$. After choosing a basis for this stalk, the node state can be written as a matrix $\mathbf{H}_i \in \mathbb{R}^{d\times C}$. However, this coordinate representation is not intrinsic: different nodes may use different bases, and there is generally no canonical basis shared by all stalks. Therefore, if we change the basis of the stalk $\mathcal{F}(i)$ by a matrix $\mathbf{G}_i \in \mathbb{G}_i \subseteq GL(d)$, the same underlying sheaf signal is represented in different coordinates as:

$$\mathbf{H}_i \mapsto \mathbf{H}'_i := \mathbf{G}_i\mathbf{H}_i \quad \text{with} \quad \mathbf{G}_i \in \mathbb{G} \tag{7}$$

where $\mathbb{G}$ depends on the restriction-map family: a diagonal subgroup of $GL(d)$ in the diagonal case, $O(d)$ or a metric-preserving subgroup in the orthogonal/bundle case, and $GL(d)$ in the general case. We refer to this action as a *local gauge action* on the stalk.

Definition 1 (Gauge Orbit) *For a node embedding $\mathbf{H} \in \mathbb{R}^{d\times C}$, define its gauge orbit as:*

$$[\mathbf{H}]_{\mathbb{G}} := \{\mathbf{GH} : \mathbf{G} \in \mathbb{G}\} \tag{8}$$

A graph-level readout is gauge-invariant *if it depends only on the multiset of node-wise gauge orbits.*

Thus, a principled sheaf readout must satisfy two requirements simultaneously:

1. **Permutation invariance over nodes:** reordering the nodes of a graph must not change the output.
2. **Gauge compatibility / invariance:** equivalent local coordinate realisations of the same stalk signal must produce the same graph descriptor.

Definition 2 (Gauge-compatible graph readout) *Let $\mathbb{G} := \prod_{i \in V_g} \mathbb{G}_i$ be the product of the admissible node-wise gauge groups, where $\mathbb{G}_i$ is determined by the restriction-map family (e.g. diagonal, orthogonal, or general invertible transforms). A graph-level readout*

$$\mathcal{R} : \{\mathbf{H}_i\}_{i \in V_g} \mapsto \mathbf{g}_g \tag{9}$$

is said to be gauge-compatible *if*

$$\mathcal{R}(\{\mathbf{G}_i \mathbf{H}_i\}_{i \in V_g}) = \mathcal{R}(\{\mathbf{H}_i\}_{i \in V_g}) \qquad \forall (\mathbf{G}_i)_{i \in V_g} \in \mathbb{G}, \tag{10}$$

and permutation-invariant *if its value is unchanged under any reordering of the node index set V_g.*

Remark 3 *In our implementation, gauge compatibility is achieved through a normalisation-and-alignment procedure, similar to [58]. We first apply node-wise whitening, which places each stalk representation in a normalised coordinate system and reduces general $GL(d)$ gauge freedom to a residual orthogonal ambiguity, namely, removes scale and shear effects. However, each whitened stalk may still differ by an orthogonal rotation or reflection. To make node representations comparable before pooling, we then align all whitened stalks to a common learnable reference, or anchor, by solving an anchor-based Procrustes problem. This step selects, for each node, the orthogonal transformation that best matches its whitened stalk representation to the anchor. Once all stalks are expressed in this shared aligned frame, we apply a permutation-invariant pooling operator to obtain the graph-level descriptor. This design makes the readout robust to local changes of basis. The invariance we obtain here is approximate in finite samples and may degrade when the stalk covariance matrices are ill-conditioned.*

B.2. Flatten-and-Pool fails for Sheaf Neural Networks

In sheaf neural networks, since each node representation $\mathbf{H}_i \in \mathbb{R}^{d \times C}$ lives in a *stalk space*, two tensors that represent the same underlying local signal may differ by a transformation of the form $\mathbf{H}_i \mapsto \mathbf{G}_i \mathbf{H}_i$, with $\mathbf{G}_i$ taken from the relevant gauge group. Any pooling rule that acts directly on the raw coordinates of $\mathbf{H}_i$ therefore risks depending on arbitrary local frame choices rather than on the intrinsic sheaf signal itself. This issue is particularly important after flattening. Indeed, vectorising $\mathbf{H}_i$ identifies the stalk and channel axes with a single ambient Euclidean coordinate system, thereby implicitly assuming that all nodes are already expressed in a common stalk basis. But this assumption is precisely what fails in the sheaf setting: different nodes may carry equivalent information in different local coordinates. As a result, applying a standard graph readout to $\mathrm{vec}(\mathbf{H}_i) \in \mathbb{R}^{dC}$ mixes together quantities that are not canonically comparable across nodes.

From a *geometric viewpoint*, flatten-and-pool collapses the quotient structure induced by the local gauge action. A correct sheaf readout should first construct canonical representatives of these classes before aggregation, and naive flattening does not do that. Therefore, even if the encoder has learned meaningful stalk-structured node states, a basis-dependent readout may destroy this structure at the final stage and produce graph embeddings that vary under purely representational changes.

Proposition 4 (Flatten-and-pool is not gauge-invariant) *Let* $\mathrm{vec}(\cdot)$ *denote vectorization. Consider any readout that pools flattened node embeddings through a coordinate-wise operator such as sum, mean, max, or standard attention on* $\mathrm{vec}(\mathbf{H}_i) \in \mathbb{R}^{d\times C}$. *Then, this readout is not invariant under node-wise gauge transformations* $\mathbf{H}_i \mapsto \mathbf{G}_i\mathbf{H}_i$.

Proof Using the standard Kronecker identity, we have that:

$$\mathrm{vec}(\mathbf{GH}) = (\mathbf{I}_C \otimes \mathbf{G})\,\mathrm{vec}(\mathbf{H}) \tag{11}$$

Hence, under a node-wise gauge transform we obtain:

$$\mathrm{vec}(\mathbf{H}_i') = (\mathbf{I}_C \otimes \mathbf{G}_i)\,\mathrm{vec}(\mathbf{H}_i) \tag{12}$$

If we consider, for instance, sum pooling, we get:

$$\mathbf{g}_{\mathrm{sum}} := \sum_{i\in V_g} \mathrm{vec}(\mathbf{H}_i) \tag{13}$$

After the reparameterisation, we then have:

$$\mathbf{g}'_{\mathrm{sum}} = \sum_{i\in V_g} (\mathbf{I}_C \otimes \mathbf{G}_i)\,\mathrm{vec}(\mathbf{H}_i) \tag{14}$$

which is generically different from $\mathbf{g}_{\mathrm{sum}}$, unless all $\mathbf{G}_i = \mathbf{I}_d$ or a non-generic cancellation occurs. Mean pooling is just a rescaled version of sum pooling and fails for the same reason. Max pooling is not equivariant under general invertible linear maps, and thus coordinate-wise maxima are also basis-dependent. Likewise, attention on flattened features changes because both attention logits and aggregated values depend on the chosen coordinates unless the scoring and value maps are themselves gauge-invariant. Therefore, direct flatten-and-pool is not a principled sheaf readout. ■

The proposition shows that one must either pool purely gauge-invariant node descriptors or first canonicalise stalk frames before pooling in stalk space. The former is robust but collapses the stalk structure too early, while the latter is the route we adopt.

B.3. Design Principle: Canonicalise and Pool

A valid readout for sheaf graph classification should satisfy the following desiderata: it must support graph-level prediction, it must remain meaningful for nontrivial stalk dimension $d > 1$, it must remain valid across diagonal, orthogonal, and general restriction-map families and it should preserve stalk-structured information as long as possible before the final graph compression.

We have seen that the natural alternatives are indeed insufficient. First, standard pooling on flattened embeddings is computationally simple but fails Proposition 4. Second, direct stalk-space pooling with energy-based weights is valid only if the residual gauge is orthogonal, and therefore does not extend to the general $GL(d)$ case. Third, invariant-only pooling, for

example pooling per-node channel energies, is universal but collapses the stalk dimension before cross-node interactions in stalk space can be exploited.

Our solution is a *two-stage canonicalisation followed by stalk-space aggregation*. We first use per-node whitening to remove the non-orthogonal part of the local gauge. We then use anchor-based orthogonal Procrustes alignment to express all nodes in a common stalk frame. Only after this canonicalisation, we pool in $\mathbb{R}^{d\times C}$. A late, receive-only token is then used to add graph-level expressive capacity without altering the encoder dynamics. The proposed *Stalk-Space Late-token Readout* has already been described in Section 4, but now we are going to see it again together with mathematical derivations and proofs.

B.4. Proposed Stalk-Space Late-token Readout

Given node embeddings $\mathbf{H}_i \in \mathbb{R}^{d\times C}$, we define the readout:

$$\mathcal{R} : \{\mathbf{H}_i\}_{i\in V_g} \longmapsto \mathbf{g}_g \in \mathbb{R}^C \tag{15}$$

through the five-stage pipeline:

$$\mathbf{H}_i \xrightarrow{\text{whiten}} \mathbf{Z}_i \xrightarrow{\text{align}} \widetilde{\mathbf{Z}}_i \xrightarrow{\text{attn pool}} \mathbf{P}_g \xrightarrow{\text{late token}} \widehat{\mathbf{T}}_g \xrightarrow{\text{energy}} \mathbf{g}_g \tag{16}$$

Step 1: Per-node whitening. For each node, we define the regularised stalk-side Gram matrix as:

$$\mathbf{S}_i = \mathbf{H}_i\mathbf{H}_i^\top + \varepsilon\mathbf{I}_d \in \mathbb{R}^{d\times d} \tag{17}$$

where the term $\varepsilon\mathbf{I}_d$ acts as a Tikhonov-style regulariser that yields a slightly smoothed, but well-defined, whitening transform. Its purpose is to ensure that Eq. 17 remains positive definite, even when $\mathbf{H}_i\mathbf{H}_i^\top$ is singular or ill-conditioned. In particular, if some directions in stalk space have very small variance, the inverse square root $\mathbf{S}_i^{-1/2}$ may otherwise be unstable or undefined. By shifting all eigenvalues of $\mathbf{H}_i\mathbf{H}_i^\top$ by ε, this term stabilises the whitening operation, prevents the amplification of near-zero directions, and makes the computation numerically robust. Then, the whitened representation is:

$$\mathbf{Z}_i = \mathbf{S}_i^{-1/2}\mathbf{H}_i \in \mathbb{R}^{d\times C} \tag{18}$$

When $\varepsilon = 0$ and $\mathbf{H}_i\mathbf{H}_i^\top$ is invertible, we have $\mathbf{Z}_i\mathbf{Z}_i^\top = \mathbf{I}_d$. Thus, whitening removes anisotropic scaling and shearing in stalk space and reduces the ambiguity to an orthogonal one.

Step 2: Anchor-based Procrustes alignment. After whitening, different nodes may still differ by a residual orthogonal transform. We fix a shared learnable anchor $\mathbf{A} \in \mathbb{R}^{d\times C}$ and solve:

$$\mathbf{R}_i^\star = \arg\min_{\mathbf{R}\in O(d)} \|\mathbf{R}^\top\mathbf{Z}_i - \mathbf{A}\|_F^2 \tag{19}$$

The aligned representation is then done by:

$$\widetilde{\mathbf{Z}}_i = \mathbf{R}_i^{\star\top}\mathbf{Z}_i \tag{20}$$

Step 3: Invariant attention weights. We define a per-node invariant descriptor using channel-wise energies of the whitened features:

$$\mathbf{u}_i = \mathrm{diag}(\mathbf{Z}_i^\top \mathbf{Z}_i) \in \mathbb{R}^C \tag{21}$$

A scorer $s_\theta : \mathbb{R}^C \to \mathbb{R}$ gives graph-wise attention weights:

$$\alpha_i = \frac{\exp(s_\theta(\mathbf{u}_i))}{\sum_{j \in V_g} \exp(s_\theta(\mathbf{u}_j))} \tag{22}$$

Step 4: Stalk-space pooling and a late receive-only token. We pool directly in the stalk space:

$$\mathbf{P}_g = \sum_{i \in V_g} \alpha_i \widetilde{\mathbf{Z}}_i \in \mathbb{R}^{d \times C} \tag{23}$$

A learnable late token $\mathbf{T} \in \mathbb{R}^{d \times C}$ is then updated only *after* encoder message passing:

$$\widehat{\mathbf{T}}_g = \phi([\mathbf{T}, \mathbf{P}_g]) \in \mathbb{R}^{d \times C} \tag{24}$$

where ϕ is a small channel-wise MLP with layer normalisation.

Step 5: Final invariant graph descriptor. We finally collapse the stalk dimension by channel-wise energies:

$$\mathbf{g}_g = \mathrm{diag}(\widehat{\mathbf{T}}_g^\top \widehat{\mathbf{T}}_g) \in \mathbb{R}^C, \qquad [\mathbf{g}_g]_c = \sum_{a=1}^{d} (\widehat{\mathbf{T}}_g)_{a,c}^2 \tag{25}$$

B.4.1. Whitening Removes the $GL(d)$ Gauge up to an Orthogonal Factor

We first state the exact whitening equivariance result in the ideal full-row-rank setting, and then discuss the practically relevant regularised case.

Theorem 5 (Exact whitening canonicalises up to $O(d)$ in the full-rank case) *Fix a node and let $\mathbf{H} \in \mathbb{R}^{d \times C}$. Assume that $\mathbf{H}$ has full row rank, so that:*

$$\mathbf{S} := \mathbf{H}\mathbf{H}^\top \succ 0 \tag{26}$$

Let's define:

$$\mathbf{Z} = \mathbf{S}^{-1/2}\mathbf{H} \tag{27}$$

Let $\mathbf{H}' = \mathbf{G}\mathbf{H}$ with $\mathbf{G} \in GL(d)$, and define:

$$\mathbf{S}' = \mathbf{H}'\mathbf{H}'^\top = \mathbf{G}\mathbf{S}\mathbf{G}^\top, \qquad \mathbf{Z}' = (\mathbf{S}')^{-1/2}\mathbf{H}' \tag{28}$$

Then there exists $\mathbf{Q} \in O(d)$ such that:

$$\mathbf{Z}' = \mathbf{Q}\mathbf{Z} \tag{29}$$

Proof Let $\mathbf{M} := \mathbf{G}\mathbf{S}^{1/2}$. Then, we have that:

$$\mathbf{S}' = \mathbf{G}\mathbf{S}\mathbf{G}^\top = \mathbf{M}\mathbf{M}^\top \tag{30}$$

By the polar decomposition, there exist $\mathbf{Q} \in O(d)$ and $\mathbf{P} \succ 0$ such that:

$$\mathbf{M} = \mathbf{Q}\mathbf{P} \tag{31}$$

Hence:

$$\mathbf{S}' = \mathbf{Q}\mathbf{P}^2\mathbf{Q}^\top, \qquad (\mathbf{S}')^{-1/2} = \mathbf{Q}\mathbf{P}^{-1}\mathbf{Q}^\top \tag{32}$$

Therefore, we get:

$$\begin{aligned}\mathbf{Z}' &= (\mathbf{S}')^{-1/2}\mathbf{H}' = (\mathbf{S}')^{-1/2}\mathbf{G}\mathbf{H} = (\mathbf{S}')^{-1/2}\mathbf{G}\mathbf{S}^{1/2}\mathbf{S}^{-1/2}\mathbf{H} \\ &= (\mathbf{S}')^{-1/2}\mathbf{M}\mathbf{Z} = \mathbf{Q}\mathbf{P}^{-1}\mathbf{Q}^\top\mathbf{Q}\mathbf{P}\mathbf{Z} = \mathbf{Q}\mathbf{Z}\end{aligned} \tag{33}$$

Thus, whitening removes the non-orthogonal part of the $GL(d)$ gauge, leaving only a residual orthogonal ambiguity. ■

Remark 6 (On the full-rank assumption) *The matrix $\mathbf{H}\mathbf{H}^\top$ is always positive semidefinite, but it is positive definite only when $\mathbf{H}$ has full row rank. This need not hold automatically in learned sheaf embeddings, since channel mixing and feature collapse may produce rank-deficient or ill-conditioned stalk-side Gram matrices.*

Remark 7 (Regularized case used in practice) *In practice, we use the regularised matrix:*

$$\mathbf{S}_\varepsilon := \mathbf{H}\mathbf{H}^\top + \varepsilon\mathbf{I}_d, \qquad \varepsilon > 0 \tag{34}$$

which is always symmetric positive definite, regardless of the rank of $\mathbf{H}$. The corresponding whitening transform:

$$\mathbf{Z}_\varepsilon := \mathbf{S}_\varepsilon^{-1/2}\mathbf{H} \tag{35}$$

is therefore always well defined and numerically stable. However, because:

$$\mathbf{H}'\mathbf{H}'^\top + \varepsilon\mathbf{I}_d \neq \mathbf{G}(\mathbf{H}\mathbf{H}^\top + \varepsilon\mathbf{I}_d)\mathbf{G}^\top \tag{36}$$

in general, the exact identity $\mathbf{Z}'_\varepsilon = \mathbf{Q}\mathbf{Z}_\varepsilon$ no longer holds. The regularised whitening should therefore be viewed as a stable approximation to the ideal canonicalisation result of Theorem 5.

B.4.2. Orthogonal Procrustes Alignment

We next derive the alignment step.

Proposition 8 (Closed-form Procrustes solution) *Given $\mathbf{Z} \in \mathbb{R}^{d\times C}$ and $\mathbf{A} \in \mathbb{R}^{d\times C}$, consider:*

$$\mathbf{R}^\star = \arg\min_{\mathbf{R}\in O(d)} \|\mathbf{R}^\top\mathbf{Z} - \mathbf{A}\|_F^2 \tag{37}$$

Let $\mathbf{C} := \mathbf{Z}\mathbf{A}^\top$ *and let its singular value decomposition be* $\mathbf{C} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$*. Then:*

$$\mathbf{R}^\star = \mathbf{U}\mathbf{V}^\top \tag{38}$$

is an optimiser. If one restricts to $\mathbf{R} \in SO(d)$*, then an optimizer is:*

$$\mathbf{R}^\star = \mathbf{U}\mathbf{D}\mathbf{V}^\top \tag{39}$$

where $\mathbf{D}$ *is diagonal with last entry* $\det(\mathbf{U}\mathbf{V}^\top)$ *and all other diagonal entries equal to* 1*.*

Proof Expanding the objective here, we get:

$$\|\mathbf{R}^\top\mathbf{Z} - \mathbf{A}\|_F^2 = \operatorname{tr}(\mathbf{Z}\mathbf{Z}^\top) + \operatorname{tr}(\mathbf{A}\mathbf{A}^\top) - 2\operatorname{tr}(\mathbf{R}^\top\mathbf{Z}\mathbf{A}^\top) \tag{40}$$

The first two terms do not depend on $\mathbf{R}$, so minimising is equivalent to maximising:

$$\operatorname{tr}(\mathbf{R}^\top\mathbf{C}), \qquad \mathbf{C} = \mathbf{Z}\mathbf{A}^\top \tag{41}$$

Let's write $\mathbf{C} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$. Then, we get:

$$\operatorname{tr}(\mathbf{R}^\top\mathbf{C}) = \operatorname{tr}(\mathbf{V}^\top\mathbf{R}^\top\mathbf{U}\mathbf{\Sigma}) = \operatorname{tr}(\mathbf{M}\mathbf{\Sigma}) \tag{42}$$

where $\mathbf{M} := \mathbf{V}^\top\mathbf{R}^\top\mathbf{U} \in O(d)$. Since $\mathbf{\Sigma}$ is diagonal with nonnegative entries, the maximum is attained when $\mathbf{M} = \mathbf{I}_d$, which gives $\mathbf{R}^\star = \mathbf{U}\mathbf{V}^\top$. The proper-rotation case is obtained by the usual determinant correction. ■

B.4.3. Exact Invariance of the Canonicalised Representative

The key point is that whitening and alignment together can produce an exact canonical representative in the idealised regime.

Theorem 9 (Exact gauge invariance of the aligned representative) *Assume* $\varepsilon = 0$*,* $\mathbf{H}_i\mathbf{H}_i^\top \succ 0$ *for every node* i*, and that for every node* i*, the cross-covariance* $\mathbf{C}_i := \mathbf{Z}_i\mathbf{A}^\top$ *has full rank and a simple singular spectrum, so that the orthogonal Procrustes solution is unique. Let* $\mathbf{H}'_i = \mathbf{G}_i\mathbf{H}_i$ *with* $\mathbf{G}_i \in GL(d)$*. Let* $\widetilde{\mathbf{Z}}_i$ *and* $\widetilde{\mathbf{Z}}'_i$ *denote the aligned representatives obtained from* $\mathbf{H}_i$ *and* $\mathbf{H}'_i$ *using the same anchor* $\mathbf{A}$*. Then:*

$$\widetilde{\mathbf{Z}}'_i = \widetilde{\mathbf{Z}}_i \qquad \text{for all } i \tag{43}$$

Proof By Theorem 5, there exists $\mathbf{Q}_i \in O(d)$ such that:

$$\mathbf{Z}'_i = \mathbf{Q}_i\mathbf{Z}_i \tag{44}$$

Now let:

$$\mathbf{C}_i = \mathbf{Z}_i\mathbf{A}^\top, \qquad \mathbf{C}'_i = \mathbf{Z}'_i\mathbf{A}^\top = \mathbf{Q}_i\mathbf{C}_i \tag{45}$$

Suppose $\mathbf{C}_i = \mathbf{U}_i\mathbf{\Sigma}_i\mathbf{V}_i^\top$ is the SVD of $\mathbf{C}_i$. Then an SVD of $\mathbf{C}'_i$ is:

$$\mathbf{C}'_i = (\mathbf{Q}_i\mathbf{U}_i)\mathbf{\Sigma}_i\mathbf{V}_i^\top \tag{46}$$

By Proposition 8, the unique Procrustes solutions are:

$$\mathbf{R}_i^\star = \mathbf{U}_i \mathbf{V}_i^\top, \qquad \mathbf{R}_i'^\star = (\mathbf{Q}_i \mathbf{U}_i)\mathbf{V}_i^\top = \mathbf{Q}_i \mathbf{R}_i^\star \tag{47}$$

Therefore, we get:

$$\widetilde{\mathbf{Z}}_i' = (\mathbf{R}_i'^\star)^\top \mathbf{Z}_i' = (\mathbf{Q}_i \mathbf{R}_i^\star)^\top \mathbf{Q}_i \mathbf{Z}_i = \mathbf{R}_i^{\star\top} \mathbf{Z}_i = \widetilde{\mathbf{Z}}_i \tag{48}$$

Hence, the aligned representative is invariant to the original $GL(d)$ gauge action. ■

Corollary 10 (Invariance of the energy descriptor) *Under the assumptions of Theorem 9, the descriptor* $\mathbf{u}_i = \mathrm{diag}(\mathbf{Z}_i^\top \mathbf{Z}_i)$ *is gauge-invariant.*

Proof By Theorem 5, $\mathbf{Z}_i' = \mathbf{Q}_i \mathbf{Z}_i$ with $\mathbf{Q}_i \in O(d)$, hence:

$$\mathbf{Z}_i'^\top \mathbf{Z}_i' = \mathbf{Z}_i^\top \mathbf{Q}_i^\top \mathbf{Q}_i \mathbf{Z}_i = \mathbf{Z}_i^\top \mathbf{Z}_i \tag{49}$$

Taking diagonals proves the claim. ■

B.4.4. Permutation Invariance and Gauge Invariance of the Readout

Theorem 11 (Permutation and gauge invariance of the readout) *Assume the setting of Theorem 9. Then the readout can be written as:*

$$\mathbf{g}_g = \mathrm{diag}\left(\phi\left[\mathbf{T}, \sum_{i \in V_g} \alpha_i \widetilde{\mathbf{Z}}_i\right]^\top \phi\left[\mathbf{T}, \sum_{i \in V_g} \alpha_i \widetilde{\mathbf{Z}}_i\right] \right) \tag{50}$$

with:

$$\alpha_i = \frac{\exp(s_\theta(\mathbf{u}_i))}{\sum_{j \in V_g} \exp(s_\theta(\mathbf{u}_j))} \tag{51}$$

This is permutation-invariant over nodes and gauge-invariant under node-wise $GL(d)$ reparameterisations.

Proof Permutation invariance follows because the logits $s_\theta(\mathbf{u}_i)$ are computed independently at each node and the normalisation is a symmetric sum over nodes. Therefore the multiset $\{(\alpha_i, \widetilde{\mathbf{Z}}_i)\}_{i \in V_g}$ is merely reindexed under a node permutation, and:

$$\sum_{i \in V_g} \alpha_i \widetilde{\mathbf{Z}}_i \tag{52}$$

is unchanged. Since ϕ, concatenation with $\mathbf{T}$, and the final energy map are deterministic functions of this pooled tensor, the output is permutation-invariant. Gauge invariance follows from Theorem 9 and Corollary 10: under node-wise $GL(d)$ reparameterization, $\widetilde{\mathbf{Z}}_i$ and $\mathbf{u}_i$ are unchanged. Therefore the attention weights α_i are unchanged, so $\mathbf{P}_g$, $\widehat{\mathbf{T}}_g$, and finally $\mathbf{g}_g$ are all unchanged. ■

B.5. Classification Token in Vision Transformers Comparison

In order to better understand the role of the late token, we can make a comparison with *classification token* (`[CLS]`) in *Vision Transformers* [16], even if the two mechanisms play fundamentally different roles.

The reason for this difference is structural: while in a ViT, all patch tokens already live in a single global embedding space, so a global token may interact with them directly throughout the encoder, in sheaf neural networks, node states live in stalk spaces and are only defined up to node-wise basis changes. Hence a token cannot be inserted meaningfully into the encoder without first resolving or otherwise respecting the stalk gauge. Moreover, in graph message passing, inserting a token node connected to all nodes changes the propagation graph itself. This effectively introduces global shortcuts, reduces the effective graph diameter, and may accelerate oversmoothing by forcing repeated mixing through a global hub. Our token is therefore intentionally *late* and *receive-only*.

Proposition 12 (The late token does not alter encoder propagation) *Let $\mathcal{E}_\theta$ denote the sheaf encoder, so that:*

$$\{\mathbf{H}_i\}_{i\in V_g} = \mathcal{E}_\theta(G_g, X_g) \tag{53}$$

Let $\mathcal{R}_\omega$ denote the late-token readout. Then the full model factors as:

$$\widehat{y}_g = \mathcal{D}_\eta \circ \mathcal{R}_\omega \circ \mathcal{E}_\theta(G_g, X_g) \tag{54}$$

and the encoder mapping $\mathcal{E}_\theta$ is independent of the late token parameters.

Proof By construction, the late token is introduced only after the encoder has already produced $\{\mathbf{H}_i\}_{i\in V_g}$. It is therefore not part of the message passing graph and does not enter any propagation, diffusion, or attention update inside $\mathcal{E}_\theta$. Consequently the propagation operator of the encoder is unchanged. ■

Remark 13 *In this sense, our token is closer to a* post-encoder set summarisation token *than to a standard transformer class token. Its job is not to mediate message passing, but to add expressive readout capacity after gauge-compatible aggregation has already taken place.*

B.6. Injective Stalk-Space Readout on the Canonicalised Descriptor Space

As concerns the *injectivity property*, the correct place to state an injectivity theorem is *after canonicalisation*, i.e. on the space of gauge-compatible node descriptors. This follows the same multiset philosophy used in expressive GNN theory: if one can map multisets of node descriptors injectively, then graph-level pooling can be made discriminative.

Canonical descriptor space. For each node, define the canonical descriptor as:

$$\xi_i := (\mathrm{vec}(\widetilde{\mathbf{Z}}_i), \mathbf{u}_i) \tag{55}$$

This descriptor contains both the aligned stalk tensor and the invariant energy information. After exact canonicalisation, ξ_i is gauge-invariant.

Assumption 14 (Bounded graphs and countable descriptor universe) *There exists a countable set $\mathfrak{X}$ such that all canonical node descriptors satisfy $\xi_i \in \mathfrak{X}$, and there exists $B \in \mathbb{N}$ such that every graph satisfies $|V_g| \leq B$.*

This assumption mirrors the standard bounded-multiset/countable-universe assumption used in expressivity theory for injective multiset aggregation. It is exact when descriptors are discrete, quantised, or drawn from a countable universe; in practice, with continuous features, it should be interpreted as a theorem about the idealised discrete case and as guidance for approximate learnable implementations on compact domains.

Theorem 15 (Existence of an injective gauge-aware multiset sum) *Under the previous assumptions, we know that there exists a function:*

$$\psi : \mathfrak{X} \to \mathbb{R} \tag{56}$$

such that the multiset map:

$$\mathcal{S}(\{\xi_i\}_{i \in V_g}) := \sum_{i \in V_g} \psi(\xi_i) \tag{57}$$

is injective on multisets of canonical descriptors of size at most B.

Proof Because $\mathfrak{X}$ is countable, there exists an injective indexing map $Z : \mathfrak{X} \to \mathbb{N}$. Let's choose any integer $M > B$. We can define then:

$$\psi(\xi) := M^{-Z(\xi)} \tag{58}$$

We claim that:

$$\mathcal{S}(X) = \sum_{\xi \in X} \psi(\xi) \tag{59}$$

is injective on multisets $X \subset \mathfrak{X}$ of cardinality at most B. Indeed, write a multiset X through its multiplicity function $m_X : \mathfrak{X} \to \{0, 1, \ldots, B\}$. Then, we get:

$$\mathcal{S}(X) = \sum_{\xi \in \mathfrak{X}} m_X(\xi)\, M^{-Z(\xi)} \tag{60}$$

This is a base-M expansion whose "digits" are exactly the multiplicities $m_X(\xi)$, each strictly smaller than M. Such an expansion is unique. Therefore if $\mathcal{S}(X) = \mathcal{S}(Y)$, then all digits coincide, i.e. $m_X(\xi) = m_Y(\xi)$ for every $\xi \in \mathfrak{X}$. Hence $X = Y$ as multisets, proving injectivity. ■

Corollary 16 (Representation of bounded multiset functions) *Under the same assumptions, for any function $g : \{\text{bounded multisets over } \mathfrak{X}\} \to \mathcal{Y}$, there exists a function*

ρ such that:

$$g(X) = \rho\left(\sum_{\xi \in X} \psi(\xi)\right) \tag{61}$$

where ψ is the injective map from Theorem 15.

Proof By Theorem 15, the quantity $\sum_{\xi \in X} \psi(\xi)$ uniquely identifies the multiset X. Therefore, we may also define:

$$\rho\left(\sum_{\xi \in X} \psi(\xi)\right) := g(X) \tag{62}$$

This is well-defined because the sum is injective. ■

The previous theorems/corollaries are the exact analogues, on the canonicalised sheaf descriptor space, of the classical injective-sum multiset theorems used to analyse expressive GNNs.

B.6.1. Injective Readout on Gauge Orbits

We now strengthen the previous result by making explicit the quotient-space structure induced by the local gauge action and by stating precisely the condition under which the canonicalisation map is not only gauge-invariant, but also *orbit-separating*. This allows us to prove injectivity at the level of bounded multisets of node-wise gauge orbits, rather than only at the level of bounded multisets of canonical descriptors.

Definition 17 (Orbit multiset) *For a graph g, we can define the multiset of node-wise gauge orbits as:*

$$\Omega_g := \{[\mathbf{H}_i]_{\mathbb{G}} : i \in V_g\} \tag{63}$$

where $[\mathbf{H}_i]_{\mathbb{G}}$ denotes the orbit of $\mathbf{H}_i$ under the local gauge group $\mathbb{G}$.

Assumption 18 (Orbit-separating canonicalization) *Let*

$$\mathcal{C}(\mathbf{H}) := \xi = (\mathrm{vec}(\widetilde{\mathbf{Z}}), \mathbf{u}) \tag{64}$$

denote the canonical descriptor produced by whitening and alignment. We assume that, on the domain of interest, $\mathcal{C}$ is orbit-separating*, i.e.*

$$\mathcal{C}(\mathbf{H}) = \mathcal{C}(\mathbf{H}') \quad \Longrightarrow \quad [\mathbf{H}]_{\mathbb{G}} = [\mathbf{H}']_{\mathbb{G}}. \tag{65}$$

Equivalently, the induced map on the quotient space

$$\overline{\mathcal{C}} : \mathbb{R}^{d \times C}/\mathbb{G} \to \mathfrak{X}, \qquad \overline{\mathcal{C}}([\mathbf{H}]_{\mathbb{G}}) := \mathcal{C}(\mathbf{H}), \tag{66}$$

is injective, where $\mathfrak{X}$ is the canonical descriptor space.

Theorem 19 (Injective readout on bounded multisets of gauge orbits) *Assume Theorem 9, Assumption 14, and Assumption 18. Let:*

$$\mathcal{C}(\mathbf{H}_i) := \xi_i = \big(\mathrm{vec}(\widetilde{\mathbf{Z}}_i), \mathbf{u}_i\big) \tag{67}$$

be the canonical descriptor map. Then there exists a graph readout of the form:

$$\mathcal{R}_{\mathrm{inj}}(\{\mathbf{H}_i\}_{i\in V_g}) := \rho\left(\sum_{i\in V_g} \psi(\mathcal{C}(\mathbf{H}_i))\right) \tag{68}$$

such that:

1. $\mathcal{R}_{\mathrm{inj}}$ *is permutation-invariant over nodes;*
2. $\mathcal{R}_{\mathrm{inj}}$ *is gauge-invariant under node-wise* $GL(d)$ *actions;*
3. $\mathcal{R}_{\mathrm{inj}}$ *is injective on bounded multisets of node-wise gauge orbits, i.e.:*

$$\Omega_g \neq \Omega_{g'} \quad \Longrightarrow \quad \mathcal{R}_{\mathrm{inj}}(\{\mathbf{H}_i\}_{i\in V_g}) \neq \mathcal{R}_{\mathrm{inj}}(\{\mathbf{H}'_j\}_{j\in V_{g'}}) \tag{69}$$

In particular, one may choose ρ *to be the identity on the image of the multiset sum.*

Proof We proceed in four steps.

Step 1: The canonical descriptor induces a well-defined map on gauge orbits. By Theorem 9, if two node embeddings belong to the same gauge orbit, i.e. $\mathbf{H}' = \mathbf{G}\mathbf{H}$ for some $\mathbf{G} \in \mathbb{G}$, then their aligned representatives and invariant descriptors coincide. Therefore, we have $\mathcal{C}(\mathbf{H}') = \mathcal{C}(\mathbf{H})$. Hence $\mathcal{C}$ is constant on gauge orbits, and so it induces a well-defined quotient map:

$$\overline{\mathcal{C}} : \mathbb{R}^{d\times C}/\mathbb{G} \to \mathfrak{X}, \qquad \overline{\mathcal{C}}([\mathbf{H}]_{\mathbb{G}}) := \mathcal{C}(\mathbf{H}) \tag{70}$$

Step 2: The quotient map is injective on the domain of interest. By Assumption 18, the induced map $\overline{\mathcal{C}}$ is injective. Therefore, two distinct gauge orbits are mapped to two distinct canonical descriptors. In particular, if $[\mathbf{H}]_{\mathbb{G}} \neq [\mathbf{H}']_{\mathbb{G}}$, then, we have:

$$\overline{\mathcal{C}}([\mathbf{H}]_{\mathbb{G}}) \neq \overline{\mathcal{C}}([\mathbf{H}']_{\mathbb{G}}) \tag{71}$$

Thus, on the domain of interest, the canonical descriptor does not merely forget the gauge, but it provides a faithful representative of the gauge orbit.

Step 3: Canonical orbit multisets are injectively encoded by a multiset sum. Let $\Xi_g := \{\xi_i : i \in V_g\} = \{\mathcal{C}(\mathbf{H}_i) : i \in V_g\}$ be the multiset of canonical descriptors associated with graph g. Since $\mathfrak{X}$ is countable and graph sizes are bounded by Assumption 14, Theorem 15 guarantees the existence of a function $\psi : \mathfrak{X} \to \mathbb{R}^m$ for some m such that the multiset sum $\mathcal{S}(\Xi_g) := \sum_{\xi\in\Xi_g} \psi(\xi)$ is injective on bounded multisets of descriptors. Equivalently, we have that:

$$\mathcal{S}(\Xi_g) = \mathcal{S}(\Xi_{g'}) \quad \Longrightarrow \quad \Xi_g = \Xi_{g'} \tag{72}$$

Step 4: Injectivity transfers from descriptor multisets to orbit multisets. Now suppose that two graphs g and g' satisfy $\mathcal{R}_{\text{inj}}(\{\mathbf{H}_i\}_{i\in V_g}) = \mathcal{R}_{\text{inj}}(\{\mathbf{H}'_j\}_{j\in V_{g'}})$. Choose ρ to be injective on the image of the multiset sum and, in particular, one may simply take ρ to be the identity. Then:

$$\sum_{i\in V_g} \psi(\mathcal{C}(\mathbf{H}_i)) = \sum_{j\in V_{g'}} \psi(\mathcal{C}(\mathbf{H}'_j)) \tag{73}$$

By injectivity of the multiset sum from Step 3, this implies that $\Xi_g = \Xi_{g'}$. Since $\overline{\mathcal{C}}$ is injective by Step 2, equality of descriptor multisets implies equality of the corresponding multisets of gauge orbits, i.e., $\Omega_g = \Omega_{g'}$. Taking the contrapositive, if $\Omega_g \neq \Omega_{g'}$, then:

$$\mathcal{R}_{\text{inj}}(\{\mathbf{H}_i\}_{i\in V_g}) \neq \mathcal{R}_{\text{inj}}(\{\mathbf{H}'_j\}_{j\in V_{g'}}) \tag{74}$$

This proves injectivity on bounded multisets of node-wise gauge orbits.
Finally, permutation invariance follows immediately from the commutativity of summation: $\sum_{i\in V_g} \psi(\mathcal{C}(\mathbf{H}_i))$ depends only on the multiset $\{\mathcal{C}(\mathbf{H}_i)\}_{i\in V_g}$ and not on any ordering of the nodes. Gauge invariance follows from Step 1, since $\mathcal{C}$ is constant on each gauge orbit. Therefore, the readout is permutation-invariant, gauge-invariant, and injective on bounded multisets of node-wise gauge orbits. ■

Remark 20 (Practical Interpretation) *Theorem 19 shows that, once the local gauge has been removed through an orbit-separating canonicalisation, graph-level readout reduces to injective multiset encoding over canonical node descriptors. This is the precise sheaf analogue of injective multiset readout results in expressive graph representation learning: the only additional subtlety is that injectivity must now be established on* gauge orbits, *not on raw node coordinates.*

Appendix C. Additional Experimental, Ablation, and Implementation Details

This appendix consolidates all supplementary material related to the empirical study and implementation of HetSheaf. We begin by describing the experimental protocol, hyperparameter search space, and datasets used throughout the paper. We then present a detailed model analysis, including ablations over heterogeneous sheaf predictors, restriction-map families, attention-based variants, and architectural choices such as depth and stalk dimension. Next, we provide broader comparisons with recent heterogeneous GNNs reported in the literature, in order to situate HetSheaf within the wider benchmark landscape. Finally, we collect the main implementation details, including hardware, decoders, restriction-map parameterisations, the full family of heterogeneous sheaf predictors, computational overhead, graph-level readout costs, and the heterogeneous extensions of Sheaf Attention Networks and Polynomial Neural Sheaf Diffusion. Together, these experiments complement the main paper by clarifying both the empirical behaviour and the practical design choices behind the model.

C.1. Experimental Protocol, Hyperparameters, and Datasets

We gather here the information necessary to reproduce the empirical setup used in the main paper. We first describe the hyperparameter search procedure shared across the evaluated models, and then summarise both the benchmark datasets and the task-specific datasets used in our study.

For this paper, we rely on a broad collection of heterogeneous benchmarks [45, 78, 66, 37, 2, 48, 35, 67, 47, 41, 19, 20] spanning *node classification*, *link prediction*, and *graph classification*. The suite covers citation graphs (e.g., DBLP, ACM), knowledge graphs (e.g., IMDB), user-item interaction networks (e.g., LastFM, MovieLens, Amazon, YouTube), link prediction graphs (e.g., PubMed), biological graphs (e.g., PROTEINS, ENZYMES), chemical compounds (e.g., MUTAG, NCI1), biomedical networks (e.g., CTD-DDA, DrugBank, NDFRT-DDA, String-PPI), and networking tasks (e.g., AS773, RouteNet Traffic). We note that, although PROTEINS, ENZYMES, and NCI1 contain a single edge type, they comprise multiple node types (3, 3, and 37, respectively): they therefore directly probe heterogeneous node/stalk spaces and the gauge-compatible SheafPool readout, while MUTAG combines several node types with four bond types, exercising both node- and edge-type conditioning. We compare against representative homogeneous (GCN [39], GAT [65], GIN [70], GraphSAGE [1]), heterogeneous (R-GCN [57], HAN [66], HGT [37]), and sheaf-based baselines [6]. In Appendix C.3, we compare against recent state-of-the-art heterogeneous models, HAGNN [79], HINormer [46], SeHGNN [73], and SlotGAT [78], re-run under a fully matched protocol, together with contextual results from the literature. We follow the standard protocols for each benchmark and use the HGB evaluation setup [45]. We report *Macro/Micro-F1* for classification and *AUPR/AUROC* for link prediction. Negative edges for link prediction are generated following prior work on heterogeneous relation prediction [24, 76, 11]. Model selection is performed through hyperparameter sweeps (150 runs per model type, both for baselines and for restriction map type for Sheaf-NSD and HetSheaf). The best validation-selected configuration of each model is then re-trained across five random seeds (42–46), and all reported results are the mean $\pm$ standard deviation

across these seeds. In the following Appendices C.1, C.3 and C.4, we provide detailed dataset descriptions, preprocessing choices, additional baseline comparisons from the literature, hyperparameter search spaces, hardware details, and other implementation notes.

C.1.1. Model Hyperparameters

Table 5 reports the search space used to optimise all models. For each method, we adopt the same approach usually followed by the main Sheaf Neural Networks frameworks [6], [4], [7] and we run large-scale Bayesian hyperparameter sweeps (150 runs per model type, both for baselines, and for restriction map type for Sheaf-NSD and HetSheaf) [60, 52] and select the best configuration according to the validation metric appropriate to the task. This shared protocol allows a fair comparison between homogeneous baselines, heterogeneous GNNs, and sheaf-based models under a common tuning budget.

Table 5: **Hyperparameter search space.**

Hyperparameter	Search space
Sheaf type	$\{\text{general}, O(d), \text{diagonal}, \text{SheafAN}\}$
Sheaf learner	$\{\text{NSD}, \text{TE}, \text{EE}, \text{NE}, \text{types}, \text{NT}, \text{ET}\}$
Attention Learner[†]	$\{\text{Yes}, \text{No}\}$
d	$\{2, 3, 4, 5\}$
# Layers	$\{2, 3, 4, 5, 6, 7, 8\}$
input_dropout	$[0, 0.9]$
dropout	$[0, 0.9]$
initial_dropout	$[0, 0.9]$
Learning rate	$\{1, 5\} \times \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$
Weight decay	$\{1, 5\} \times \{10^{-5}, 10^{-4}, 10^{-3}\}$

[†] Not supported for *diagonal* sheaf type.

C.2. Additional Results and Model Analysis

This section complements the headline results from the main paper with a more detailed analysis of where the gains of HetSheaf come from. We study in particular the effect of the heterogeneous sheaf predictor, the restriction-map family, the use of attention-based restriction-map constructions, and architectural choices such as depth and stalk dimension. The goal is not only to identify the strongest configuration, but also to understand which modelling components matter most for different tasks and datasets.

C.2.1. Benchmark Datasets

Each of the datasets used for node classification is publicly available as part of the Heterogeneous Graph Benchmark (HGB)[1] first introduced by Lv et al. [45]. The link prediction datasets (LastFM, MovieLens, PubMed and Amazon) are also drawn from HGB. On the other hand, graph classification datasets are drawn from the TU Datasets benchmark [48], a widely used collection of graph-level prediction benchmarks spanning biological and chemical domains. We provide brief descriptions of each dataset.

1. https://www.biendata.xyz/hgb/

Table 6: **Statistics of benchmark datasets.**

Node classification	# Nodes	# Node types	# Edges	# Edge types	Target	# Classes
DBLP	26,128	4	239,566	6	author	4
IMDB	21,420	4	86,642	6	movie	5
ACM	10,942	4	547,806	7	paper	3
Link prediction					Target	
LastFM	20,612	3	141,521	3	user-artist	
MovieLens	10,352	2	201,672	2	user-movie	
PubMed	63,109	4	244,986	10	disease-disease	
YouTube	2,000	1	1,177,130	5	user-user	
Amazon	10,099	1	148,659	2	product-product	
Graph classification	# Graphs	# Node types	# Edges	# Edge types	Target	
MUTAG	188	7	7,442	4	mutagenicity	
PROTEINS	1,113	3	162,088	1	enzyme/non-enzyme	
ENZYMES	600	3	19,580	1	enzyme class	
NCI1	4,110	37	49,205	1	anti-cancer	

- **DBLP**[2] is a citation graph based on the DBLP computer science bibliographic database. It has four node types: authors, papers, venues and terms. The six edge types include paper-author, paper-venue, paper-term and author-venue. The target is to predict the class label of authors based on their research area: databases, data mining, AI or information retrieval.

- **IMDB**[3] is an information graph containing movie data. The four node types are movies, directors, actors and keywords. The six edge types include movie-director, movie-actor and movie-keyword. Movies are classified based on their genre: action, comedy, drama, romance and thriller. Each movie can have multiple genres, i.e., a multi-label node classification task.

- **ACM** [66] is a citation graph containing node types of authors, papers, terms and subjects. The edge types include paper-cite-paper, paper-author, paper-subject and paper-term. The target is to predict each paper's category: databases, wireless communication, or data mining.

- **LastFM**[4] is a knowledge graph generated from an online music website. The three node types are user, artist, and tags, with edge types of user-artist, user-user and artist-tag. The link prediction task aims to predict edges between users and artists.

- **MovieLens**[5] is a heterogeneous rating graph from the MovieLens website and consists of two node types: movie and user. The task is to predict whether or not a user rates a movie.

- **PubMed**[6] is a biomedical knowledge graph from the HGB benchmark [45], constructed from the PubMed literature database. It contains four node types (gene, disease,

2. https://web.cs.ucla.edu/~yzsun/data/
3. https://www.kaggle.com/datasets/karrrimba/movie-metadatacsv
4. https://grouplens.org/datasets/hetrec-2011/
5. https://movielens.org/
6. https://pubmed.ncbi.nlm.nih.gov

chemical and species) connected by ten edge types encoding biological relations such as gene-disease associations, chemical-chemical interactions and species-disease links. The link prediction task targets disease-disease edges.

- **Amazon** is a single-node-type product graph from the HGB benchmark [45] containing 10,099 items connected by co-viewing and co-purchasing relations (two edge types). The link prediction task targets product-product edges.
- **YouTube** is a single-node-type social network from the HGB benchmark [45] containing 2,000 users connected by five types of interactions. The link prediction task targets user-user edges.
- **MUTAG** [48] is a collection of 188 mutagenic aromatic and heteroaromatic nitro compounds, with 7 atom types and 4 bond types. The task is to predict whether a compound has a mutagenic effect on a bacterium.
- **PROTEINS** [48] is a dataset of 1,113 proteins represented as graphs, where nodes are secondary structure elements with 3 node types and edges connect amino acids that are neighbours in the sequence or in 3D space. The task is to classify proteins as enzymes or non-enzymes.
- **ENZYMES** [48] is a dataset of 600 protein tertiary structures obtained from the BRENDA enzyme database, with 3 node types representing secondary structure elements. The task is to classify each protein into one of 6 enzyme commission top-level classes.
- **NCI1** [48] is a dataset of 4,110 chemical compounds screened for their ability to suppress or inhibit the growth of a panel of human tumour cell lines, with 37 atom types. The task is to predict anti-cancer activity.

C.2.2. Biomedical Datasets

The following datasets are used exclusively for biomedical link prediction [35, 67] experiments and are provided in the appendix for completeness.

Table 7: **Statistics of biomedical link prediction datasets.**

Biomedical	# Nodes	# Node types	# Edges	# Edge types	Target
CTD-DDA	12,765	2	43,492	1	drug-disease
DrugBank-DDI	2,191	1	242,027	1	drug-drug
NDFRT-DDA	13,545	2	4,709	1	drug-disease
String-PPI	15,131	1	359,776	1	protein-protein

- **CTD-DDA**[7] is a heterogeneous graph derived from the Comparative Toxicogenomics Database, containing curated associations between chemical compounds and diseases. The link prediction task aims to predict drug-disease associations.

7. https://github.com/Zaiwen/Link_Prediction_in_Biomedical_Network

- **DrugBank**[7] is a comprehensive heterogeneous graph constructed from the DrugBank database [68], containing drugs, targets, enzymes, and carriers. The link prediction task aims to predict drug-target interactions.
- **NDFRT-DDA**[7] is a heterogeneous graph derived from the National Drug File Reference Terminology, containing drug-disease associations curated from clinical data. The task is to predict drug-disease associations.
- **String-PPI**[7] is a protein-protein interaction network derived from the STRING database [63], containing functional associations between proteins across multiple organisms. The task is to predict protein-protein interactions.

Table 8: **Performance on heterogeneous biomedical networks link prediction benchmarks.** Results for the heterogeneous sheaf learners and baselines are shown. We report the highest binary AUROC and AUPR scores achieved across the hyperparameter sweep for each model. The top three models are coloured by **First**, **Second** and **Third**.

	CTD-DDA		DrugBank		NDFRT-DDA		String-PPI	
	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC
Homogeneous								
GCN	**99.36**	**99.09**	93.46	91.57	97.80	96.51	95.11	92.62
GAT	89.55	89.92	88.73	87.46	94.01	92.88	94.77	92.26
GIN	99.34	99.07	93.71	92.43	97.83	96.44	95.27	92.77
SAGE	93.78	93.82	94.20	92.28	94.50	92.87	**95.94**	93.78
Heterogeneous								
R-GCN	99.03	98.93	**94.77**	**94.13**	**98.18**	**97.32**	95.79	**94.15**
HAN	63.34	51.76	88.68	85.54	63.89	52.82	93.39	90.08
HGT	63.32	51.72	89.95	87.38	63.97	52.99	93.07	90.27
Sheaf-NSD	**99.47**	**99.23**	**94.90**	**93.90**	**98.13**	**97.02**	**96.20**	**94.40**
HetSheaf (ours)	**99.53**	**99.31**	**94.92**	**93.93**	**98.23**	**97.67**	**96.05**	**94.24**

Table 8 reports link prediction results on four biomedical networks. HetSheaf ranks first on CTD-DDA (99.53 AUPR, 99.31 AUROC), DrugBank (94.92 AUPR), and NDFRT-DDA (98.23 AUPR, 97.67 AUROC), while Sheaf-NSD leads on String-PPI (96.20 AUPR, 94.40 AUROC). The two sheaf models consistently occupy the top two positions across all datasets, with R-GCN as the strongest non-sheaf competitor. HAN and HGT fail on CTD-DDA and NDFRT-DDA (AUROC ≈ 52, near random), whereas the sheaf models remain robust. The slight advantage of Sheaf-NSD over HetSheaf on String-PPI is consistent with this dataset having a single node type, which limits the benefit of heterogeneous type conditioning.

C.2.3. Networking Datasets

The following datasets are used exclusively for networking link prediction and node classification experiments and are provided in the appendix for completeness.

- **AS-733**[8] is a heterogeneous graph of the Internet's Autonomous System (AS) topology, derived from BGP routing tables collected by the University of Oregon Route Views

8. https://snap.stanford.edu/data/as-733.html

Table 9: **Statistics of networking datasets.**

Networking	# Nodes	# Node types	# Edges	# Edge types	Task	Target
AS-733	6,474	1	13,895	3	LP	customer-provider
RouteNet Traffic Models	∼674	2	∼1,200	2	NC	link congestion

Project [41]. Nodes represent Autonomous Systems and edges represent BGP peering relationships, classified into three semantically distinct types: *customer–provider* (paid transit), *peer–to–peer* (settlement-free peering), and *provider–customer* (reverse transit). The link prediction task aims to predict missing or future peering agreements between Autonomous Systems. The dataset contains 6,474 nodes and 13,895 edges from the January 2, 2000 snapshot.

- **RouteNet Traffic Models**[9] is a bipartite heterogeneous graph for network performance modelling, generated using the BNNetSimulator [19, 20]. The graph consists of two node types: *link* nodes (physical network connections with bandwidth capacity features) and *path* nodes (logical end-to-end routes with traffic and delay features), connected by *traverses* and *belongs_to* edges encoding the routing table. The node classification task predicts which links become congested (utilisation exceeding a threshold of 0.50, yielding a 54%/46% class balance). The dataset comprises five traffic generation patterns: Constant Bit Rate (CBR), which produces deterministic fixed-rate traffic; ON-OFF, which alternates between active and idle periods to model bursty applications; Autocorrelated Exponentials, where successive inter-arrival times exhibit temporal correlations capturing long-range dependence, and Modulated Exponentials, whose rate parameter varies over time to model changing activity levels. We evaluate on the *All Multiplexed* configuration, which mixes all four traffic types simultaneously across different source-destination pairs, creating the most realistic scenario in which the model must handle qualitatively different congestion dynamics within a single graph. Training uses NSFNET (14 nodes) and GEANT2 (24 nodes) topologies.

Table 10 reports results on two networking benchmarks. On AS-733, HetSheaf achieves the highest AUPR (97.94) and AUROC (96.81), followed closely by Sheaf-NSD (97.84 / 96.65). The three heterogeneous edge types (customer–provider, peer–to–peer, provider–customer) encode semantically distinct peering relationships whose directionality and economic asymmetry provide a natural testbed for type-aware restriction maps; even a modest gain over Sheaf-NSD suggests that the heterogeneous predictor captures inter-type structure that a single shared learner averages away. Among baselines, GIN and GCN are competitive on AUPR, while R-GCN attains the third-best AUROC, indicating that explicit relation typing helps on this task even without the sheaf structure.

On RouteNet Traffic Models, HetSheaf and GIN both reach the top Macro F1 (87.30) and Micro F1 (87.50), with GCN as the second-best model (80.57 / 81.25). Notably, GAT, SAGE, HAN, and HGT all collapse to majority-class predictions (Macro F1 of 33.33, Micro F1 of 50.00), suggesting that the bipartite link–path topology and the diversity of traffic generation patterns (CBR, ON-OFF, Autocorrelated, Modulated Exponentials) pose a

9. https://github.com/BNN-UPC/NetworkModelingDatasets, v5

Table 10: **Performance on networking benchmarks.** Results for the heterogeneous sheaf learners and baselines are shown. We report the highest binary AUROC and AUPR for AS-733 (link prediction) and the highest Macro F1 and Micro F1 for RouteNet Traffic Models (node classification), achieved across the hyperparameter sweep for each model. The top three models are coloured by **First**, **Second** and **Third**.

	AS-733 (LP)		RouteNet Traffic (NC)	
	AUPR	AUROC	Macro F1	Micro F1
Homogeneous				
GCN	97.25	96.18	**80.57**	**81.25**
GAT	93.43	91.20	33.33	50.00
GIN	**97.33**	96.23	**87.30**	**87.50**
SAGE	96.71	95.21	33.33	50.00
Heterogeneous				
R-GCN	97.15	**96.59**	**62.50**	**62.50**
HAN	93.80	94.03	33.33	50.00
HGT	96.31	95.77	33.33	50.00
Sheaf-NSD	**97.84**	**96.65**	**87.30**	**87.50**
HetSheaf (ours)	**97.94**	**96.81**	**87.30**	**87.50**

difficult optimisation landscape for attention-based and certain heterogeneous architectures. R-GCN avoids the degenerate solution but lags considerably behind GIN and the sheaf models, confirming that per-relation linear transforms alone are insufficient to capture traffic-dependent congestion dynamics. The fact that HetSheaf matches GIN, the strongest homogeneous baseline, while additionally providing interpretable stalk-space representations, underscores its versatility across graph domains.

C.2.4. Sheaf Predictor Ablation Study

Tables 11–13 report the full ablation grid over all heterogeneous sheaf predictors and restriction-map families for node classification, link prediction, and graph classification, respectively. These results isolate the contribution of explicit type conditioning from the contribution of the geometric family itself.

For node classification (Table 11), heterogeneous predictors consistently improve over the vanilla NSD baseline on at least one dataset, confirming that typed conditioning of the restriction maps is beneficial in practice. On ACM and DBLP, the strongest configurations typically pair typed predictors with polynomial or general restriction-map families, indicating that type conditioning and higher-order diffusion are complementary. On IMDB, however, the best results are obtained by non-polynomial families, suggesting that on semantically noisier heterogeneous graphs, a simpler local transport geometry may generalise more robustly. Notably, predictors that use only type indicators without node features remain competitive, indicating that type information alone already carries substantial signal for heterogeneous node classification.

For link prediction (Table 12), the gap between predictors is generally smaller, consistent with the fact that several datasets are already close to saturation under AUROC/AUPR.

Table 11: *Sheaf predictor × restriction map ablation, node classification.* Best restriction map per sheaf learner highlighted; best sheaf learner in **bold**.

Sheaf Predictor		ACM		DBLP		IMDB	
		Macro F1	Micro F1	Macro F1	Micro F1	Macro F1	Micro F1
NSD	Diag	91.51	91.41	91.32	91.97	61.96	65.33
	Bundle	90.40	90.46	92.12	92.64	61.23	64.68
	General	89.81	89.75	89.33	89.96	61.77	64.85
	Diag-Poly	92.05	91.97	91.38	91.87	60.56	63.24
	Bundle-Poly	89.07	88.95	88.95	89.58	59.47	62.62
	Gen-Poly	86.74	86.64	91.33	91.76	59.62	62.62
	SheafAN	87.64	87.58	91.08	91.73	54.38	58.79
TE	Diag	91.99	91.88	94.03	94.51	63.55	65.40
	Bundle	91.17	91.12	91.71	92.22	62.94	66.39
	General	91.63	91.50	93.13	93.66	61.62	64.58
	Diag-Poly	**94.22**	**94.15**	94.34	94.79	61.52	64.71
	Bundle-Poly	92.07	91.97	93.04	93.56	60.26	63.29
	Gen-Poly	88.57	88.39	94.08	94.47	60.59	63.71
	SheafAN	88.20	88.20	92.39	92.96	56.06	58.83
Ensemble	Diag	92.58	92.45	93.38	93.84	63.22	65.83
	Bundle	90.29	90.23	93.11	93.56	62.62	65.47
	General	91.39	91.31	93.93	94.40	61.45	64.48
	Diag-Poly	93.36	93.25	94.26	94.65	60.34	63.73
	Bundle-Poly	87.63	87.54	90.08	90.70	61.45	63.47
	Gen-Poly	90.97	90.89	94.16	94.65	62.50	65.27
	SheafAN	86.48	86.54	81.94	83.63	55.79	60.21
NE	Diag	91.90	91.78	93.10	93.56	**63.65**	**66.56**
	Bundle	91.15	91.08	92.94	93.45	61.28	65.00
	General	91.74	91.69	93.68	94.08	62.26	65.06
	Diag-Poly	92.55	92.49	93.40	93.84	61.30	64.11
	Bundle-Poly	89.60	89.52	91.82	92.36	59.73	63.77
	Gen-Poly	91.29	91.17	94.34	94.75	60.22	62.97
	SheafAN	88.76	88.76	91.06	91.76	54.90	58.52
EE	Diag	92.50	92.40	93.50	93.91	62.86	66.49
	Bundle	91.46	91.50	92.52	92.99	62.16	64.39
	General	91.90	91.78	93.77	94.23	61.86	64.56
	Diag-Poly	93.76	93.67	94.35	94.79	60.57	63.46
	Bundle-Poly	84.38	84.37	85.66	86.55	56.44	61.57
	Gen-Poly	88.31	88.15	94.11	94.58	60.94	64.33
	SheafAN	88.14	88.05	90.62	91.23	55.56	59.23
NT	Diag	93.72	93.63	93.53	94.01	62.48	65.22
	Bundle	90.99	90.84	92.29	92.82	62.16	63.86
	General	91.66	91.55	93.50	93.98	60.95	63.99
	Diag-Poly	92.32	92.26	94.34	94.75	60.03	63.72
	Bundle-Poly	88.76	88.67	93.27	93.80	60.39	63.60
	Gen-Poly	93.58	93.48	94.06	94.47	61.18	64.20
	SheafAN	88.95	88.95	81.45	82.25	55.25	58.33
ET	Diag	92.95	92.87	94.11	94.51	61.41	64.57
	Bundle	89.90	89.80	90.35	90.85	60.23	63.42
	General	92.08	91.97	93.95	94.37	62.99	65.78
	Diag-Poly	93.07	92.97	**94.91**	**95.28**	60.71	63.80
	Bundle-Poly	89.62	89.47	86.73	87.61	60.35	63.52
	Gen-Poly	91.11	91.08	94.78	95.18	59.69	63.32
	SheafAN	88.39	88.39	81.76	82.78	55.64	58.78

Even so, typed predictors remain consistently competitive with or stronger than NSD. General and diagonal restriction maps perform particularly well on recommendation-style datasets, while the ensemble predictor occasionally benefits from edge-type-specific routing when the relation semantics are rich enough to justify separate predictors.

For graph classification (Table 13), predictor choice has a much stronger effect on performance, especially on PROTEINS, ENZYMES, and NCI1. Here, type-only predictors such as NT and ET often become surprisingly effective. This is consistent with the inductive nature of graph classification: when the model must generalise across unseen graphs, structural type information can transfer more robustly than instance-specific node features. Overall, the graph-classification ablations support the view that typed restriction-map inference is particularly valuable when the model must compress heterogeneous local geometry into a stable graph-level representation.

C.2.5. Effect of Restriction Map Type

The predictor ablations also reveal a second important axis: the choice of restriction-map family. Across tasks, no single family dominates. Polynomial families are often strongest on node classification and on several graph classification benchmarks, where longer-range typed interactions are beneficial. By contrast, non-polynomial diagonal or bundle families can be preferable on more semantically heterogeneous or noisier datasets such as IMDB and ENZYMES. This task dependence is itself informative: it suggests that the optimal local transport geometry depends on whether the problem rewards smooth long-range propagation, local typed alignment, or robust inductive generalisation.

For node classification, Diag-Poly and Gen.-Poly are often strongest on ACM and DBLP, whereas non-polynomial families dominate on IMDB. For link prediction, the restriction-map family tends to have a smaller effect, with diagonal maps performing consistently well and general maps providing only modest gains in some settings. For graph classification, polynomial families are essential on some datasets but can also become unstable on others, especially when small graph size and structural diversity interact poorly with higher-order spectral filtering. This reinforces the importance of treating the restriction-map family as a genuine modelling choice rather than as a fixed default.

C.2.6. Attention-based Restriction Map Variants

Tables 14–16 repeat the main ablation study using attention-based restriction-map constructions. These experiments test whether replacing the standard predictor output by a normalised attention-style map provides a useful inductive bias.

For node classification (Table 14), attention-based variants broaden the set of competitive configurations and occasionally outperform their non-attention counterparts, especially when paired with bundle or general maps. For link prediction (Table 15), attention predictors are competitive but do not provide a uniform improvement, suggesting that the advantage of additional normalisation depends strongly on the relation structure of the dataset. For graph classification (Table 16), the picture is mixed: attention expands the space of viable configurations, but does not yield a systematic gain over the standard construction.

Table 12: *Sheaf predictor × restriction map ablation, link prediction.* Best restriction map per sheaf learner highlighted; best sheaf learner in **bold**.

Sheaf Predictor		LastFM		MovieLens		Amazon		YouTube	
		AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC
NSD	Diag	98.06	97.73	99.68	99.60	92.59	90.03	89.06	82.85
	Bundle	97.69	97.35	99.66	99.60	92.58	89.98	89.30	83.19
	General	97.98	97.68	99.68	**99.62**	92.48	89.95	88.98	82.73
	Diag-Poly	97.34	96.56	99.65	99.56	92.60	89.79	89.45	83.32
	Bundle-Poly	97.49	97.04	99.67	99.58	92.69	90.00	89.37	83.23
	Gen-Poly	97.62	97.22	99.63	99.57	92.25	89.37	–	–
	SheafAN	93.05	91.52	99.48	99.37	91.30	87.95	82.45	76.24
TE	Diag	98.39	98.24	**99.69**	99.60	93.85	91.59	89.46	83.38
	Bundle	98.50	98.20	99.65	99.58	93.51	91.22	89.01	82.78
	General	98.45	98.30	99.67	99.60	93.71	91.55	88.87	82.58
	Diag-Poly	98.40	98.02	99.66	99.57	93.33	90.71	89.39	83.36
	Bundle-Poly	98.35	98.01	99.68	99.60	93.27	90.77	89.35	83.24
	Gen-Poly	98.30	98.07	99.65	99.59	93.49	91.28	89.21	82.95
	SheafAN	97.71	97.26	99.51	99.42	91.40	87.62	86.44	80.23
Ensemble	Diag	98.47	98.16	99.68	99.60	93.87	91.52	90.53	85.61
	Bundle	97.98	97.72	99.66	99.61	93.45	91.10	88.79	82.45
	General	**98.54**	**98.33**	99.67	99.60	93.76	91.58	90.65	85.95
	Diag-Poly	98.08	97.51	99.61	99.56	93.46	90.88	89.49	83.46
	Bundle-Poly	97.49	96.95	99.67	99.59	93.40	90.95	89.39	83.30
	Gen-Poly	98.39	98.07	99.66	99.61	93.55	91.23	–	–
	SheafAN	96.43	95.39	99.48	99.41	91.99	88.52	87.10	80.61
NE	Diag	98.43	98.24	**99.69**	99.61	92.59	90.04	89.27	83.36
	Bundle	98.42	98.16	99.67	99.60	92.49	89.88	89.33	83.23
	General	98.46	98.25	99.67	99.61	92.64	90.08	89.25	83.10
	Diag-Poly	98.42	98.05	99.66	99.57	92.56	89.79	89.41	83.28
	Bundle-Poly	98.22	97.87	99.67	99.58	92.53	89.90	89.38	83.21
	Gen-Poly	98.30	98.02	99.66	99.59	92.48	89.50	–	–
	SheafAN	97.81	97.56	99.47	99.36	91.46	88.21	82.68	76.62
EE	Diag	98.41	98.16	99.68	99.60	**93.89**	91.65	89.95	84.06
	Bundle	97.62	97.28	99.66	99.61	93.62	91.27	88.86	82.65
	General	98.51	98.13	99.67	99.61	93.71	91.51	**91.05**	**86.76**
	Diag-Poly	97.87	97.34	99.62	99.53	93.29	90.53	89.87	84.02
	Bundle-Poly	97.74	97.19	99.67	99.58	93.12	90.63	89.40	83.31
	Gen-Poly	98.17	97.77	99.66	99.59	93.60	91.30	–	–
	SheafAN	86.12	79.22	99.46	99.40	91.27	87.48	86.11	79.63
NT	Diag	98.47	98.25	99.68	99.61	92.59	90.13	89.31	83.26
	Bundle	98.45	98.24	99.66	99.61	92.37	89.75	89.23	83.17
	General	98.41	98.29	99.67	99.61	92.51	89.95	89.34	83.24
	Diag-Poly	98.28	98.08	99.66	99.57	92.61	89.82	89.41	83.26
	Bundle-Poly	98.22	97.94	99.68	99.59	92.60	89.89	89.40	83.29
	Gen-Poly	98.32	98.08	99.66	99.58	92.50	89.98	–	–
	SheafAN	97.74	97.31	99.50	99.41	91.26	87.53	82.54	76.39
ET	Diag	98.43	98.15	**99.69**	99.61	**93.89**	91.56	90.74	86.03
	Bundle	97.56	97.18	99.67	99.61	92.31	89.85	89.26	83.11
	General	98.43	98.16	99.67	99.60	93.81	**91.71**	90.62	85.95
	Diag-Poly	97.97	97.40	99.63	99.55	93.18	90.49	90.62	85.82
	Bundle-Poly	97.69	97.20	99.65	99.58	92.71	89.95	89.28	83.11
	Gen-Poly	98.03	97.72	99.67	99.58	93.77	91.58	–	–
	SheafAN	95.87	95.66	99.56	99.49	90.89	86.58	86.29	79.95

Table 13: *Sheaf predictor × restriction map ablation, graph classification.* Best restriction map per sheaf learner highlighted; best sheaf learner in **bold**.

Sheaf Predictor		MUTAG		PROTEINS		ENZYMES		NCI1	
		Macro F1	Micro F1	Macro F1	Micro F1	Macro F1	Micro F1	Macro F1	Micro F1
NSD	Diag	75.00	75.00	62.20	62.50	18.82	28.33	55.11	55.11
	Bundle	68.75	70.00	73.60	75.89	9.56	23.33	38.69	38.69
	General	76.19	80.00	26.67	33.04	8.75	23.33	–	–
	Diag-Poly	96.00	96.00	51.77	51.79	8.38	21.67	45.73	45.73
	Bundle-Poly	94.88	95.00	71.56	76.79	26.67	30.00	63.24	63.24
	Gen-Poly	89.01	90.00	66.21	71.43	5.71	20.00	72.00	72.00
	SheafAN	89.90	90.00	24.32	32.14	9.48	20.00	62.08	62.08
TE	Diag	75.00	75.00	49.98	50.00	12.70	21.67	61.85	61.85
	Bundle	69.70	70.00	64.54	65.18	12.66	21.67	62.08	62.08
	General	37.50	60.00	27.22	33.93	7.31	18.33	62.08	62.08
	Diag-Poly	95.65	95.65	33.60	35.71	13.20	23.33	61.36	61.36
	Bundle-Poly	**100.00**	**100.00**	62.78	71.43	18.54	23.33	68.62	68.62
	Gen-Poly	100.00	100.00	55.36	55.36	30.75	33.33	72.54	72.54
	SheafAN	84.00	85.00	24.32	32.14	16.98	21.67	69.70	69.70
Ensemble	Diag	78.57	78.57	73.60	75.89	11.85	20.00	29.03	29.03
	Bundle	69.70	70.00	58.28	58.93	15.92	25.00	21.43	21.43
	General	67.03	70.00	53.20	53.57	5.56	20.00	64.42	64.42
	Diag-Poly	95.65	95.65	41.27	67.86	8.20	21.67	62.08	62.08
	Bundle-Poly	100.00	100.00	69.22	71.43	24.52	25.00	51.18	51.18
	Gen-Poly	100.00	100.00	58.92	58.93	38.05	41.67	54.98	54.98
	SheafAN	37.50	60.00	24.32	32.14	15.22	23.33	62.08	62.08
NE	Diag	75.00	75.00	68.93	72.32	16.10	23.33	9.95	9.95
	Bundle	78.02	80.00	71.96	74.11	6.38	21.67	23.11	23.11
	General	64.29	70.00	28.90	33.04	9.96	21.67	54.15	54.15
	Diag-Poly	100.00	100.00	25.79	33.04	16.42	28.33	60.20	60.20
	Bundle-Poly	100.00	100.00	72.97	74.11	19.31	26.67	69.52	69.52
	Gen-Poly	100.00	100.00	63.39	66.96	13.66	20.00	41.41	41.41
	SheafAN	89.01	90.00	75.47	80.36	35.53	38.33	24.63	24.63
EE	Diag	76.19	80.00	63.87	64.29	20.03	28.33	46.99	46.99
	Bundle	37.50	60.00	25.79	33.04	5.56	20.00	18.35	18.35
	General	79.80	80.00	38.00	40.18	6.02	21.67	59.94	59.94
	Diag-Poly	96.00	96.00	38.24	41.07	13.62	21.67	52.10	52.10
	Bundle-Poly	94.88	95.00	54.29	60.71	32.82	31.67	63.89	63.89
	Gen-Poly	94.88	95.00	27.22	33.93	10.47	20.00	66.09	66.09
	SheafAN	94.88	95.00	66.16	68.75	17.13	20.00	62.08	62.08
NT	Diag	80.00	80.00	73.60	75.89	**52.96**	**53.33**	**78.24**	**78.24**
	Bundle	79.80	80.00	69.84	70.54	50.80	50.00	62.64	62.64
	General	74.94	75.00	63.25	63.39	49.92	51.67	70.85	70.85
	Diag-Poly	94.88	95.00	72.51	74.11	22.67	26.67	23.01	23.01
	Bundle-Poly	94.88	95.00	**78.84**	80.36	50.01	50.00	74.74	74.74
	Gen-Poly	89.90	90.00	78.66	**81.25**	48.13	48.33	72.86	72.86
	SheafAN	89.90	90.00	76.95	79.46	46.58	48.33	75.06	75.06
ET	Diag	76.19	80.00	48.65	67.86	12.73	26.67	57.60	57.60
	Bundle	74.42	75.00	54.37	54.46	10.39	25.00	44.92	44.92
	General	89.90	90.00	24.32	32.14	8.80	21.67	59.20	59.20
	Diag-Poly	94.88	95.00	61.35	68.75	9.15	21.67	62.08	62.08
	Bundle-Poly	94.88	95.00	65.06	74.11	15.57	25.00	44.88	44.88
	Gen-Poly	100.00	100.00	24.32	32.14	17.86	23.33	72.87	72.87
	SheafAN	58.33	60.00	61.23	61.61	17.86	20.00	62.08	62.08

Table 14: *Sheaf* attention *predictor* × *restriction map ablation, node classification.* Best restriction map per sheaf learner highlighted; best sheaf learner in **bold**.

Sheaf Predictor		ACM		DBLP		IMDB	
		Macro F1	Micro F1	Macro F1	Micro F1	Macro F1	Micro F1
Att. NSD	Bundle	89.49	89.38	92.10	92.57	61.88	64.49
	General	90.35	90.32	90.94	91.58	61.71	64.51
	Bundle-Poly	88.24	88.20	90.51	91.13	61.29	64.04
	Gen-Poly	90.93	90.89	92.96	93.42	59.88	63.40
	SheafAN	88.67	88.62	87.77	88.77	55.64	59.05
Att. TE	Bundle	90.22	90.18	93.21	93.70	63.88	66.34
	General	91.76	91.64	93.13	93.52	62.16	65.55
	Bundle-Poly	88.11	87.91	94.56	95.00	61.00	64.28
	Gen-Poly	90.18	90.09	92.79	93.27	62.39	65.81
	SheafAN	88.01	87.96	85.70	86.65	55.25	59.35
Att. Ens.	Bundle	**92.57**	**92.54**	93.46	93.94	**64.54**	**67.18**
	General	90.75	90.65	93.96	94.40	62.38	65.37
	Bundle-Poly	89.28	89.14	93.28	93.77	59.15	64.31
	Gen-Poly	88.65	88.62	93.91	94.33	58.87	62.83
	SheafAN	89.72	89.71	81.28	81.94	56.23	59.23
Att. NE	Bundle	90.76	90.70	93.66	94.12	63.03	66.47
	General	91.09	90.98	93.11	93.59	60.32	63.51
	Bundle-Poly	89.67	89.52	92.10	92.71	60.97	63.72
	Gen-Poly	88.98	88.95	94.61	95.04	58.30	61.27
	SheafAN	87.15	87.02	81.32	81.94	55.25	58.70
Att. EE	Bundle	91.79	91.69	93.32	93.77	63.50	66.44
	General	90.59	90.51	93.95	94.44	60.97	64.70
	Bundle-Poly	86.21	86.26	92.90	93.59	60.00	62.22
	Gen-Poly	91.01	90.93	92.70	93.24	60.53	63.10
	SheafAN	85.59	85.41	91.13	91.87	54.78	58.32
Att. NT	Bundle	90.78	90.70	93.73	94.23	63.00	66.37
	General	90.95	90.84	93.52	93.98	60.77	63.84
	Bundle-Poly	89.72	89.61	92.89	93.35	60.01	62.54
	Gen-Poly	91.52	91.41	94.17	94.58	59.34	63.35
	SheafAN	88.16	88.15	80.82	81.44	55.76	58.38
Att. ET	Bundle	91.76	91.69	93.81	94.26	62.57	65.97
	General	91.14	91.08	94.67	95.07	62.02	64.87
	Bundle-Poly	90.30	90.23	**94.67**	**95.11**	58.11	61.80
	Gen-Poly	88.28	88.20	93.85	94.30	62.29	65.11
	SheafAN	85.49	85.41	83.86	84.61	56.36	59.18

C.2.7. Stalk Dimension Ablation Study

Tables 17–19 report the effect of varying the stalk dimension $d \in \{2, 3, 4, 5\}$ across all restriction map families, tasks, and datasets. Each cell reports Macro-F1 / Micro-F1 for classification tasks and AUPR / AUROC for link prediction. Several patterns emerge.

First, no single stalk dimension is universally optimal: the best d varies by dataset and family, suggesting that d should be treated as a hyperparameter tuned per task. For instance,

Table 15: *Sheaf* attention *predictor* × *restriction map ablation, link prediction.* Best restriction map per sheaf learner highlighted; best sheaf learner in **bold**.

Sheaf Predictor		LastFM		MovieLens		Amazon		YouTube	
		AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC
Att. NSD	Bundle	97.98	97.71	99.66	99.60	92.53	90.00	89.32	83.36
	General	97.85	97.52	99.66	99.60	92.46	89.79	89.22	83.09
	Bundle-Poly	97.56	97.20	99.66	99.58	92.63	90.04	89.31	83.10
	Gen-Poly	97.86	97.37	99.66	99.59	92.54	89.81	–	–
	SheafAN	94.25	92.73	99.47	99.35	92.03	89.22	82.70	76.66
Att. TE	Bundle	98.46	98.20	99.65	99.59	93.85	91.59	89.17	82.99
	General	98.33	98.24	99.66	**99.61**	93.87	91.59	89.24	83.10
	Bundle-Poly	98.16	97.94	**99.68**	99.60	93.54	91.12	89.36	83.25
	Gen-Poly	97.10	96.71	99.65	99.57	92.78	90.07	–	–
	SheafAN	97.74	97.50	99.47	99.35	92.14	89.04	86.56	80.22
Att. Ens.	Bundle	98.43	98.13	99.66	99.60	93.82	91.61	**90.73**	**85.89**
	General	98.42	98.21	99.66	**99.61**	93.67	91.48	89.26	83.09
	Bundle-Poly	98.28	98.06	99.66	99.59	93.61	91.15	89.43	83.36
	Gen-Poly	98.16	97.83	99.64	99.57	**100.00**	**100.00**	64.60	64.60
	SheafAN	96.78	96.11	99.48	99.36	92.23	89.05	86.72	80.14
Att. NE	Bundle	98.45	98.28	99.65	99.59	92.67	90.02	89.31	83.20
	General	98.43	98.25	99.66	**99.61**	92.54	90.04	89.18	83.00
	Bundle-Poly	98.25	97.90	99.66	99.59	92.60	89.92	89.33	83.18
	Gen-Poly	98.29	97.93	99.67	99.59	92.58	89.90	–	–
	SheafAN	97.62	97.22	99.50	99.38	91.74	88.70	83.05	76.96
Att. EE	Bundle	98.38	98.17	99.65	99.60	93.78	91.54	89.28	83.17
	General	98.44	98.14	99.64	99.58	93.87	91.55	89.93	84.48
	Bundle-Poly	98.05	97.61	99.66	99.59	93.54	91.13	89.41	83.26
	Gen-Poly	**100.00**	**100.00**	99.65	99.58	93.61	91.23	–	–
	SheafAN	94.78	92.51	99.52	99.41	91.81	88.42	86.27	80.13
Att. NT	Bundle	98.41	98.24	99.66	99.59	92.55	90.05	89.28	83.16
	General	98.45	98.28	99.65	99.60	92.41	89.94	89.04	82.83
	Bundle-Poly	98.36	97.98	99.66	99.58	92.63	89.82	89.39	83.19
	Gen-Poly	98.34	98.06	99.67	99.59	92.56	89.79	–	–
	SheafAN	97.94	97.62	99.49	99.42	91.18	87.58	82.40	76.25
Att. ET	Bundle	98.35	98.13	99.66	99.60	93.75	91.44	90.27	84.88
	General	98.43	98.19	99.66	99.60	93.82	91.55	90.05	84.88
	Bundle-Poly	98.27	97.95	99.67	99.58	93.41	91.09	89.42	83.34
	Gen-Poly	98.22	97.82	99.66	99.58	93.81	91.49	–	–
	SheafAN	93.70	91.14	99.52	99.44	91.92	87.95	86.11	79.88

Gen.-P peaks at $d=2$ on both ACM (94.3/94.2) and DBLP (95.0/95.4), but at $d=3$ on PROTEINS (80.6/82.1) and $d=2$ on NCI1 (79.5/79.5).

Second, polynomial variants (Diag-P, Bundle-P, Gen.-P) consistently match or outperform their non-polynomial counterparts on node and graph classification. Gen.-P achieves the global best Macro-F1 on ACM (94.3), DBLP (95.0), PROTEINS (80.6), and NCI1 (79.5). On MUTAG, both Bundle-P and Gen.-P reach a perfect 100.0/100.0 across all stalk dimensions, indicating that the polynomial spectral filtering combined with orthogonal or general restriction maps provides sufficient expressivity to fully separate the two classes. However, this advantage does not extend uniformly to all datasets: on ENZYMES, Diag-P performs

Table 16: *Sheaf* attention *predictor* × *restriction map ablation, graph classification.* Best restriction map per sheaf learner highlighted; best sheaf learner in **bold**.

Sheaf Predictor		MUTAG		PROTEINS		ENZYMES		NCI1	
		Macro F1	Micro F1	Macro F1	Micro F1	Macro F1	Micro F1	Macro F1	Micro F1
Att. NSD	Bundle	64.91	70.00	25.79	33.04	33.65	35.00	55.12	55.12
	General	71.51	75.00	24.32	32.14	5.56	20.00	35.44	35.44
	Bundle-Poly	76.19	80.00	24.32	32.14	12.17	20.00	62.08	62.08
	Gen-Poly	**100.00**	**100.00**	66.75	71.43	14.13	23.33	**72.30**	**72.30**
	SheafAN	94.88	95.00	52.37	52.68	12.35	21.67	48.47	48.47
Att. TE	Bundle	79.80	80.00	54.29	54.46	18.27	26.67	59.88	59.88
	General	41.33	60.00	63.25	63.39	8.01	21.67	58.82	58.82
	Bundle-Poly	59.60	65.00	72.57	75.89	21.29	21.67	62.08	62.08
	Gen-Poly	100.00	100.00	49.42	50.00	16.60	28.33	60.06	60.06
	SheafAN	78.02	80.00	75.24	76.79	17.05	28.33	–	–
Att. Ens.	Bundle	84.00	85.00	43.49	44.64	18.47	30.00	60.38	60.38
	General	73.33	75.00	64.22	66.96	5.56	20.00	63.49	63.49
	Bundle-Poly	73.33	75.00	62.53	63.39	22.13	23.33	19.44	19.44
	Gen-Poly	94.67	95.00	41.70	43.75	16.62	23.33	–	–
	SheafAN	94.88	95.00	28.63	34.82	15.35	20.00	19.18	19.18
Att. NE	Bundle	94.88	95.00	24.32	32.14	36.67	43.33	55.36	55.36
	General	73.33	75.00	25.25	32.14	17.36	30.00	28.09	28.09
	Bundle-Poly	94.88	95.00	49.60	67.86	2.98	8.33	63.02	63.02
	Gen-Poly	100.00	100.00	70.10	74.11	19.45	31.67	5.26	5.26
	SheafAN	71.51	75.00	63.04	63.39	31.74	35.00	62.08	62.08
Att. EE	Bundle	78.02	80.00	33.97	38.39	16.35	26.67	62.30	62.30
	General	94.88	95.00	24.32	32.14	19.35	25.00	60.44	60.44
	Bundle-Poly	94.88	95.00	63.25	63.39	21.27	23.33	19.18	19.18
	Gen-Poly	100.00	100.00	41.27	43.75	23.58	26.67	–	–
	SheafAN	37.50	60.00	70.05	71.43	18.49	26.67	11.76	11.76
Att. NT	Bundle	79.17	80.00	78.84	80.36	51.23	53.33	67.98	67.98
	General	79.17	80.00	46.27	46.43	13.17	26.67	24.45	24.45
	Bundle-Poly	84.65	85.00	**79.69**	**81.25**	**56.71**	**56.67**	–	–
	Gen-Poly	89.58	90.00	75.94	80.36	41.23	43.33	70.06	70.06
	SheafAN	84.96	85.00	76.95	79.46	46.52	50.00	64.30	64.30
Att. ET	Bundle	84.65	85.00	24.32	32.14	7.78	21.67	22.81	22.81
	General	79.17	80.00	38.92	41.96	8.67	23.33	47.85	47.85
	Bundle-Poly	89.90	90.00	24.32	32.14	24.06	25.00	64.74	64.74
	Gen-Poly	100.00	100.00	65.17	66.07	21.86	23.33	69.89	69.89
	SheafAN	94.88	95.00	24.32	32.14	13.36	20.00	62.08	62.08

poorly (peaking at only 25.6/30.0), well below the non-polynomial Diag (53.0/53.3) and Bundle (62.3/63.3 at $d=4$).

Third, SheafAN lags behind the Laplacian-based families on node classification, particularly on IMDB, where it peaks at 56.4/59.3 Macro/Micro-F1 versus 64.5/67.2 for Bundle. The gap is smaller but consistent on ACM and DBLP. However, SheafAN is more competitive on link prediction, where it achieves strong AUPR/AUROC on LastFM (97.9/97.6 at $d=5$) and near-saturated performance on MovieLens. This suggests that the attention-based adjacency aggregation of SheafAN is better suited for pairwise scoring tasks than for node-level classification.

Fourth, link prediction performance is largely insensitive to d. AUPR and AUROC are near-saturated across all configurations (e.g., 98.3–98.5 AUPR on LastFM, 99.5–99.7 on MovieLens), and vary by less than 0.3 percentage points within each family. This contrasts sharply with node and graph classification, where performance swings of 5–20 points across

stalk dimensions are common (e.g., General on MUTAG ranges from 74.9 at $d=5$ to 94.9 at $d=3$).

Finally, the non-polynomial families (Diag, Bundle, General) tend to be more stable across d than their polynomial counterparts. The polynomial families occasionally exhibit sharp performance drops, such as Diag-P on ENZYMES falling from 25.6 at $d=3$ to 14.0 at $d=4$, or Gen.-P on ACM degrading from 94.3 at $d=2$ to 90.6 at $d=5$. This instability may stem from the higher-order Chebyshev coefficients amplifying noise when the graph structure does not support long-range spectral patterns, and reinforces the recommendation to treat both d and the restriction map family as hyperparameters.

Table 17: Node classification: Macro-F1 / Micro-F1 (%) by restriction map family and stalk dimension d. Cyan = best in family, **bold** = global best per dataset.

	ACM				DBLP				IMDB			
Family	$d=2$	$d=3$	$d=4$	$d=5$	$d=2$	$d=3$	$d=4$	$d=5$	$d=2$	$d=3$	$d=4$	$d=5$
Diag	93.7/93.6	93.0/92.9	92.6/92.5	93.0/92.9	94.0/94.5	94.1/94.5	94.1/94.5	93.5/94.0	62.4/65.4	63.6/66.6	62.8/65.9	63.2/65.8
Bundle	92.6/92.5	91.2/91.1	91.0/90.8	92.1/92.1	93.8/94.3	93.7/94.2	93.2/93.7	93.4/93.9	64.4/**67.2**	**64.5**/66.6	63.9/66.5	64.2/66.4
General	91.8/91.7	91.7/91.7	91.9/91.8	92.1/92.0	93.8/94.3	94.0/94.4	94.7/95.1	94.0/94.4	63.5/65.6	62.5/65.5	62.4/65.4	63.0/65.8
Diag-P	93.3/93.2	93.1/93.0	94.2/94.1	93.5/93.4	94.9/95.3	94.7/95.1	94.9/95.3	94.6/95.0	61.5/64.7	61.6/64.6	61.1/64.1	60.2/63.6
Bundle-P	90.3/90.2	92.1/92.0	89.3/89.1	90.3/90.2	94.7/95.1	93.3/93.8	94.6/95.0	93.6/94.0	61.5/64.3	61.8/64.6	61.0/64.3	61.9/65.2
Gen.-P	**94.3/94.2**	93.6/93.5	93.1/93.1	90.6/90.5	**95.0/95.4**	94.4/94.8	94.6/95.0	94.6/95.0	62.4/65.8	62.5/64.9	62.3/65.3	60.9/64.6
SheafAN	88.9/89.0	89.7/89.7	88.8/88.8	90.9/90.9	90.9/91.6	91.1/91.9	92.4/93.0	87.3/88.3	56.1/60.2	56.2/59.2	55.9/59.2	56.4/59.3

Table 18: Graph classification: Macro-F1 / Micro-F1 (%) by restriction map family and stalk dimension d. Cyan = best in family, **bold** = global best per dataset.

	MUTAG				PROTEINS				ENZYMES				NCI1			
Family	$d=2$	$d=3$	$d=4$	$d=5$	$d=2$	$d=3$	$d=4$	$d=5$	$d=2$	$d=3$	$d=4$	$d=5$	$d=2$	$d=3$	$d=4$	$d=5$
Diag	80.0/80.0	80.0/80.0	83.3/83.3	76.9/76.9	74.6/75.9	67.3/67.9	73.6/75.9	61.5/67.9	53.0/53.3	46.1/46.7	34.2/35.0	39.1/40.0	72.1/72.1	68.5/68.5	73.7/73.7	61.8/61.8
Bundle	78.0/80.0	84.7/85.0	94.9/95.0	84.7/85.0	74.8/76.8	75.8/77.7	78.8/80.4	54.4/54.5	50.8/50.0	51.2/53.3	**62.3/63.3**	51.0/51.7	62.6/62.6	57.6/57.6	68.0/68.0	60.4/60.4
General	89.0/90.0	94.9/95.0	89.9/90.0	74.9/75.0	71.3/72.3	64.2/74.1	78.7/81.2	64.2/67.0	49.9/51.7	47.7/50.0	50.7/51.7	46.9/46.7	63.2/63.2	64.4/64.4	67.5/67.5	62.1/62.1
Diag-P	95.7/95.7	95.7/95.7	96.0/96.0	**100.0/100.0**	51.8/67.9	78.9/**82.1**	71.6/74.1	63.2/74.1	22.7/28.3	25.6/30.0	14.0/21.7	14.9/23.3	60.2/60.2	62.1/62.1	62.1/62.1	61.4/61.4
Bundle-P	**100.0/100.0**	**100.0/100.0**	**100.0/100.0**	**100.0/100.0**	75.6/77.7	78.8/80.4	78.4/80.4	79.8/**82.1**	52.6/53.3	46.5/48.3	51.9/51.7	56.7/56.7	69.2/69.2	54.4/54.4	74.7/74.7	76.3/76.3
Gen.-P	**100.0/100.0**	**100.0/100.0**	**100.0/100.0**	**100.0/100.0**	75.9/80.4	**80.6/82.1**	78.7/81.2	77.5/79.5	34.4/36.7	41.6/43.3	41.7/43.3	48.1/48.3	**79.5/79.5**	72.9/72.9	71.5/71.5	73.0/73.0
SheafAN	89.9/90.0	94.9/95.0	94.9/95.0	94.9/95.0	75.8/78.6	76.9/79.5	77.2/79.5	76.7/80.4	46.6/48.3	49.3/51.7	41.6/41.7	37.3/40.0	72.7/72.7	71.9/71.9	73.6/73.6	75.1/75.1

Table 19: Link prediction: (AUPR / AUROC, %). by restriction map family and stalk dimension d. Cyan = best in family, **bold** = global best per dataset.

	LastFM				MovieLens				Amazon				YouTube			
Family	$d=2$	$d=3$	$d=4$	$d=5$	$d=2$	$d=3$	$d=4$	$d=5$	$d=2$	$d=3$	$d=4$	$d=5$	$d=2$	$d=3$	$d=4$	$d=5$
Diag	98.4/98.2	98.5/98.2	98.4/98.2	98.5/98.2	**99.7**/99.6	99.7/99.6	99.7/99.6	**99.7**/99.6	93.8/91.5	93.8/91.6	93.8/91.6	93.8/91.5	90.5/85.4	90.7/86.0	90.6/85.7	90.6/85.9
Bundle	98.5/98.2	98.4/98.3	98.5/98.2	98.5/98.2	99.7/**99.6**	99.7/99.6	99.7/99.6	99.7/99.6	93.7/91.3	93.8/91.6	93.7/91.5	93.8/91.6	90.1/84.4	89.3/83.2	90.4/85.4	90.7/85.9
General	98.5/98.3	98.5/98.3	98.5/98.3	98.5/98.3	99.7/99.6	99.7/**99.6**	99.7/**99.6**	99.7/**99.6**	93.8/91.6	93.8/91.6	93.8/91.6	93.8/91.7	90.7/86.0	90.1/84.9	**91.3/87.3**	**91.3/87.3**
Diag-P	98.3/97.9	98.4/98.0	98.3/98.0	98.4/98.1	99.6/99.5	99.7/99.6	99.6/99.6	99.7/99.6	93.2/90.6	93.3/90.5	93.5/90.9	93.3/90.8	90.1/85.3	90.5/85.8	90.7/86.5	90.6/85.4
Bundle-P	98.3/98.0	98.3/98.1	98.3/98.1	98.4/98.0	99.7/99.6	99.7/99.6	99.7/99.6	99.7/99.6	93.5/91.1	93.5/91.1	93.6/91.1	93.6/91.2	89.4/83.4	89.4/83.3	89.4/83.3	89.4/83.3
Gen.-P	98.3/98.1	98.3/98.0	98.3/98.1	**98.5/98.3**	99.7/99.6	99.7/99.6	99.7/99.6	99.7/99.6	**93.8/91.7**	93.8/91.5	93.8/91.6	93.8/91.4	89.2/83.0	-/-	-/-	-/-
SheafAN	97.7/97.5	97.7/97.3	97.8/97.2	97.9/97.6	99.5/99.4	99.5/99.4	99.6/99.5	99.5/99.4	91.9/88.8	92.0/88.7	92.2/89.1	92.1/89.2	87.0/80.5	87.1/80.6	87.0/80.5	86.1/79.6

C.2.8. Layer Depth Ablation Study

Tables 20–22 report the effect of varying the number of message passing layers $L \in \{2, 3, 4, 5, 6, 7, 8\}$ across all restriction map families, tasks, and datasets, alongside the homogeneous and heterogeneous GNN baselines. Each cell reports Macro-F1 / Micro-F1 for classification tasks and AUPR / AUROC for link prediction. Several patterns emerge.

The most striking observation is the *absence of catastrophic oversmoothing* in HetSheaf. On ACM, the best Macro-F1 across all sheaf families varies by at most 3.4 percentage points between the shallowest ($L=2$) and deepest ($L=8$) configuration (Gen.-P: 94.3/94.2 at $L=2$, 92.6/92.5 at $L=8$), and several families (Diag, General) achieve their peak performance at $L \geq 6$. In contrast, the homogeneous baselines suffer dramatic degradation at depth: on ACM, GAT drops from 92.2/92.1 at $L=3$ to 17.0/34.3 at $L=8$, GCN from 92.3/92.2 at $L=2$ to 32.8/40.9 at $L=8$, and GIN from 88.7/88.8 at $L=2$ to 41.3/43.7 at $L=8$. Even the heterogeneous baselines are not immune: HAN collapses from 93.6/93.6 at $L=5$ to 27.2/37.1 at $L=8$ on ACM, and HGT drops from 92.3/92.3 at $L=2$ to 40.1/46.0 at $L=8$. The only baseline that remains stable across depth is R-GCN, whose edge-type-specific parameterisation provides a form of oversmoothing resistance, though at a substantially higher parameter cost (Table 28).

On DBLP, the advantage of depth is more pronounced for sheaf models. Several families (General, Gen.-P) achieve their best results at $L \geq 4$, with Gen.-P peaking at 95.0/95.4 at $L=4$ and remaining above 93.9/94.3 through $L=8$. The baselines also benefit from moderate depth on DBLP (GIN peaks at 94.8/95.2 at $L=7$), but none match the best sheaf configurations. On IMDB, where all methods struggle, the sheaf families show a mild but consistent benefit from additional layers: Diag and Bundle both achieve their best results at $L \geq 6$ (Diag: 63.6/66.6 at $L=8$; Bundle: 64.5/66.6 at $L=6$), suggesting that deeper message passing helps capture the complex multi-label structure of this dataset. Notably, the IMDB baselines are extremely brittle: GCN and HAN oscillate between functional performance at odd layer counts and near-collapse at even ones, a pattern that is entirely absent from the sheaf models.

For graph classification, the depth–performance relationship is noisier, reflecting the smaller test sets and inductive evaluation protocol. On ENZYMES, deeper sheaf models consistently outperform shallower ones: Bundle achieves 62.3/63.3 at $L=8$ (its global best) versus 38.2/43.3 at $L=2$, and Diag reaches 53.0/53.3 at $L=8$ versus 31.0/40.0 at $L=2$. The baselines on ENZYMES are uniformly poor (GCN: 9–13, GAT: 6–14, SAGE: 6–9 Macro-F1 across all depths), and only HAN (46.1/45.0 at $L=6$) and HGT (48.3/48.3 at $L=5$) reach moderate performance. On NCI1, Gen.-P peaks at 79.5/79.5 at $L=7$, and Diag reaches 78.2/78.2 at $L=3$, both substantially above the homogeneous baselines (GCN: 64.2/65.0, GAT: 64.2/65.0 at best). The polynomial families exhibit greater variance across layers than the non-polynomial families: on NCI1, Diag-P ranges from 61.4 to 62.1, whereas Diag ranges from 61.8 to 78.2. This is consistent with the observation from the stalk dimension ablation that polynomial spectral filtering amplifies sensitivity to hyperparameter choices.

For link prediction (Table 22), all sheaf families are effectively invariant to depth. On LastFM, AUPR varies by at most 0.2 percentage points across layers for all non-polynomial families, and AUROC varies by at most 0.3 points. On MovieLens, performance is near-constant across all depths and families (AUPR $\geq$ 99.5 and AUROC $\geq$ 99.4 for every sheaf

configuration), substantially outperforming HAN which is essentially non-functional on this dataset (AUPR ≈ 63.0 regardless of depth). On Amazon and YouTube, the baselines are more competitive, with HGT achieving the strongest baseline AUPR on both datasets (96.2 and 95.0 respectively), but the sheaf models match or exceed these scores across all layer depths.

Overall, these results demonstrate that the sheaf Laplacian diffusion process provides a principled mechanism for deep message passing without the oversmoothing pathology that limits standard GNNs to shallow architectures. The restriction maps provide per-edge transport operators that preserve discriminative signal across layers, while the polynomial variants add spectral filtering that can exploit deeper receptive fields when the graph structure supports it. In practice, we recommend treating L as a hyperparameter tuned alongside d and the restriction map family, with the expectation that $L \in \{3, \ldots, 6\}$ providing a robust default range.

Table 20: Node classification: Macro-F1 / Micro-F1 (%) by model and number of layers L. Cyan = best in family, **bold** = global best per dataset.

	ACM							DBLP							IMDB						
	$L=2$	$L=3$	$L=4$	$L=5$	$L=6$	$L=7$	$L=8$	$L=2$	$L=3$	$L=4$	$L=5$	$L=6$	$L=7$	$L=8$	$L=2$	$L=3$	$L=4$	$L=5$	$L=6$	$L=7$	$L=8$
Homogeneous baselines																					
GCN	92.3/92.2	92.1/92.0	65.2/67.3	53.6/57.0	38.6/42.3	30.8/39.1	32.8/40.9	82.2/83.0	88.4/89.2	88.4/89.2	89.8/90.6	90.4/91.1	91.0/91.7	87.0/88.2	15.5/25.8	58.3/60.9	15.4/39.4	57.1/60.1	23.3/45.2	58.2/62.1	16.6/28.2
GAT	92.1/92.0	92.2/92.1	83.1/84.1	28.1/38.2	29.2/37.6	28.8/38.3	17.0/34.3	84.4/85.1	90.3/91.0	89.4/89.9	92.3/92.8	90.0/90.7	91.1/91.7	91.4/92.0	24.7/32.9	60.2/64.0	14.1/40.0	59.0/61.3	24.7/32.9	55.7/61.3	23.3/45.2
GIN	88.7/88.8	86.8/86.5	35.3/45.2	49.0/49.4	42.2/43.3	41.6/45.4	41.3/43.7	79.9/80.5	93.6/93.9	93.2/93.7	93.2/93.7	92.9/93.4	94.8/95.2	92.9/93.3	53.1/56.9	60.2/63.9	55.7/60.0	57.3/60.8	52.2/58.5	59.1/62.2	52.1/56.7
SAGE	91.4/91.3	91.8/91.7	92.0/91.9	89.1/89.0	90.6/90.5	92.2/92.1	85.1/85.0	83.6/84.6	88.0/89.0	90.1/90.7	88.4/88.9	89.8/90.5	88.9/89.7	87.7/88.5	58.2/62.4	58.6/61.2	59.5/61.8	60.6/63.8	55.6/57.6	59.2/63.4	58.1/63.6
Heterogeneous baselines																					
R-GCN	92.0/91.9	93.3/93.2	92.9/92.8	92.1/92.0	91.3/91.2	93.0/92.9	92.0/91.9	93.6/94.0	94.3/94.7	93.6/94.1	94.0/94.5	92.8/93.4	93.8/94.2	93.4/93.9	58.8/62.3	59.3/61.5	59.7/62.4	58.3/61.2	59.1/61.5	59.3/61.6	58.4/60.9
HAN	92.2/92.1	91.4/91.3	91.0/90.9	93.6/93.6	92.3/92.3	27.1/37.0	27.2/37.1	82.7/83.3	92.1/92.6	92.8/93.3	90.8/91.4	92.7/93.2	91.8/92.4	94.0/94.4	14.1/40.0	56.8/61.7	17.2/29.5	55.5/58.9	15.5/25.8	25.5/39.5	14.1/40.0
HGT	92.3/92.3	91.5/91.4	91.8/91.8	29.8/37.0	91.7/91.7	89.0/88.8	40.1/46.0	82.7/83.6	90.6/91.3	92.8/93.3	91.0/91.7	91.2/91.7	91.3/91.9	88.2/89.0	56.5/60.2	41.4/49.4	51.3/53.6	51.5/55.3	39.0/56.1	35.3/41.8	38.8/46.2
HetSheaf (ours)																					
Diag	93.0/92.9	92.6/92.4	93.0/92.9	92.6/92.5	93.7/93.6	92.9/92.9	92.5/92.5	84.5/85.2	94.1/94.5	93.5/94.0	93.5/93.9	93.5/94.0	94.0/94.5	93.8/94.2	59.1/63.0	62.2/65.2	62.2/65.0	62.5/65.4	62.9/65.4	63.2/66.5	63.6/66.6
Bundle	91.5/91.5	91.0/90.9	92.6/92.5	92.1/92.1	90.6/90.5	91.8/91.7	91.8/91.7	84.9/85.8	93.0/93.5	92.6/93.1	93.4/93.9	93.7/94.2	93.3/93.8	93.8/94.3	59.2/63.2	62.6/65.5	62.5/66.0	62.7/65.7	64.5/66.6	63.9/66.5	64.4/**67.2**
General	91.9/91.8	91.8/91.7	91.8/91.7	91.7/91.6	91.1/91.0	91.5/91.4	92.1/92.0	85.0/85.7	93.7/94.1	94.4/94.9	94.7/95.1	94.0/94.4	94.0/94.4	93.9/94.4	58.9/62.7	62.3/64.8	60.9/64.1	63.5/65.8	62.4/65.4	62.5/65.5	62.0/64.9
Diag-P	94.2/94.1	93.8/93.7	93.9/93.8	92.5/92.4	92.6/92.5	92.3/92.3	93.5/93.4	94.5/94.9	94.9/95.3	94.7/95.1	94.6/95.0	94.4/94.8	94.2/94.6	93.9/94.4	61.5/64.7	61.1/64.1	60.5/63.7	60.3/63.5	61.6/64.6	60.6/63.2	61.3/64.1
Bundle-P	89.0/88.9	90.2/90.1	89.3/89.1	92.1/92.0	90.3/90.2	89.6/89.5	90.3/90.2	93.3/93.8	93.6/94.0	94.7/95.1	91.4/92.0	93.4/93.9	78.6/79.5	91.8/92.4	60.4/64.6	61.8/64.2	61.9/65.2	61.3/64.0	59.9/64.3	61.5/64.3	60.0/64.1
Gen.-P	**94.3/94.2**	91.0/90.9	93.6/93.5	91.8/91.7	91.5/91.4	90.9/90.9	92.6/92.5	93.9/94.3	94.7/95.1	**95.0/95.4**	94.6/95.0	94.6/95.0	94.8/95.2	93.9/94.3	60.3/63.3	60.8/64.2	62.4/65.8	**62.5**/64.9	62.3/65.3	62.3/65.1	61.2/63.7
SheafAN	88.7/88.6	88.3/88.4	88.4/88.4	88.8/88.7	88.2/88.1	90.9/90.9	88.2/88.2	86.3/87.0	91.1/91.8	82.7/83.7	90.9/91.6	92.4/93.0	91.1/91.9	90.6/91.2	55.9/59.2	54.6/58.7	56.1/58.9	55.6/59.1	56.2/59.2	56.4/59.3	55.8/60.2

C.3. External Comparisons with Additional Heterogeneous GNNs

The main paper compares HetSheaf against a controlled set of strong homogeneous, heterogeneous, and sheaf-based baselines trained under a unified codebase and tuning protocol. This section extends this comparison in two ways: first, with a *matched-protocol* re-run of four recent state-of-the-art heterogeneous GNNs; and second, with *contextual* comparisons against a broader set of heterogeneous GNNs reported in recent surveys and benchmarks [45, 79, 44].

C.3.1. Matched-Protocol Comparison with Recent State-of-the-Art Heterogeneous GNNs

We re-ran HAGNN [79], HINormer [46], SeHGNN [73], and SlotGAT [78] on the HGB node classification benchmarks under a fully matched protocol. All baselines use the authors' official implementations, are trained on the same committed split as our models (canonical HGB 24/6/70), and are scored by the same metric code, with each run asserting agreement with the implementation's own `sklearn` metrics. For each baseline, we report the best validation-selected configuration from a 100-run random search. Table 23 reports the results:

Table 21: Graph classification: Macro-F1 / Micro-F1 (%) by model and number of layers L. Cyan = best in family, **bold** = global best per dataset.

	MUTAG							PROTEINS						
	$L=2$	$L=3$	$L=4$	$L=5$	$L=6$	$L=7$	$L=8$	$L=2$	$L=3$	$L=4$	$L=5$	$L=6$	$L=7$	$L=8$
Homogeneous baselines														
GCN	89.6/90.0	89.6/90.0	89.6/90.0	89.6/90.0	89.6/90.0	89.6/90.0	89.6/90.0	76.4/80.4	71.8/77.7	69.5/71.4	75.8/78.6	74.1/76.8	69.0/70.5	75.8/78.6
GAT	89.6/90.0	89.6/90.0	89.6/90.0	89.6/90.0	89.6/90.0	58.3/60.0	84.0/85.0	56.2/56.2	75.8/78.6	71.8/77.7	48.8/70.5	71.8/77.7	74.6/77.7	52.2/52.7
GIN	89.6/90.0	94.9/95.0	89.9/90.0	85.0/85.0	85.0/85.0	89.6/90.0	84.7/85.0	70.3/71.4	51.3/71.4	64.1/69.6	70.6/72.3	–	66.8/67.9	62.7/68.8
SAGE	60.0/70.0	60.0/70.0	89.6/90.0	89.6/90.0	89.6/90.0	85.0/85.0	89.6/90.0	69.9/78.6	71.8/77.7	64.7/75.9	64.7/75.9	64.7/75.9	71.8/77.7	74.7/78.6
Heterogeneous baselines														
R-GCN	73.3/75.0	84.7/85.0	79.2/80.0	85.0/85.0	79.2/80.0	84.7/85.0	84.7/85.0	74.6/77.7	74.7/78.6	71.8/77.7	74.1/76.8	71.8/77.7	71.8/77.7	75.8/78.6
HAN	71.5/75.0	74.4/75.0	74.4/75.0	64.2/65.0	67.0/70.0	82.9/85.0	73.3/75.0	–	79.0/83.0	76.8/80.4	72.9/75.9	77.1/80.4	75.8/77.7	74.6/75.9
HGT	68.8/70.0	84.7/85.0	74.4/75.0	74.9/75.0	84.7/85.0	79.2/80.0	74.4/75.0	74.4/76.8	71.3/74.1	77.6/81.2	24.3/32.1	61.6/61.6	65.0/65.2	24.2/30.4
HetSheaf (ours)														
Diag	80.0/80.0	80.0/80.0	80.0/80.0	83.3/83.3	80.0/80.0	80.0/80.0	80.0/80.0	73.6/75.9	68.2/68.2	70.7/75.9	62.2/62.5	68.9/72.3	74.6/75.9	67.3/70.5
Bundle	79.8/80.0	84.7/85.0	84.7/85.0	94.9/95.0	84.7/85.0	84.7/85.0	79.2/80.0	78.8/80.4	74.8/75.9	75.5/76.8	50.7/50.9	75.8/77.7	74.7/76.8	69.8/70.5
General	79.8/80.0	76.2/80.0	74.9/75.0	89.9/90.0	73.3/75.0	94.9/95.0	78.0/80.0	33.3/37.5	38.0/40.2	78.7/81.2	64.1/66.1	63.2/63.4	71.3/74.1	64.2/67.0
Diag-P	89.9/90.0	96.0/96.0	95.7/95.7	**100.0/100.0**	94.9/95.0	95.7/95.7	96.0/96.0	40.6/42.9	63.2/74.1	71.6/74.1	41.3/67.9	76.4/80.4	55.4/55.4	78.9/**82.1**
Bundle-P	**100.0/100.0**	94.9/95.0	**100.0/100.0**	**100.0/100.0**	94.9/95.0	**100.0/100.0**	94.9/95.0	74.9/77.7	79.8/**82.1**	76.9/79.5	79.7/81.2	78.8/80.4	78.4/80.4	71.4/78.6
Gen.-P	**100.0/100.0**	**100.0/100.0**	**100.0/100.0**	**100.0/100.0**	**100.0/100.0**	**100.0/100.0**	**100.0/100.0**	77.8/80.4	78.1/80.4	76.7/77.7	73.9/75.9	78.7/81.2	75.8/78.6	**80.6/82.1**
SheafAN	89.0/90.0	79.2/80.0	84.0/85.0	94.9/95.0	89.9/90.0	94.9/95.0	94.9/95.0	73.8/78.6	74.7/78.6	77.2/80.4	76.4/78.6	76.7/78.6	75.6/77.7	75.2/76.8

	ENZYMES							NCI1						
	$L=2$	$L=3$	$L=4$	$L=5$	$L=6$	$L=7$	$L=8$	$L=2$	$L=3$	$L=4$	$L=5$	$L=6$	$L=7$	$L=8$
Homogeneous baselines														
GCN	11.2/25.0	12.7/28.3	9.3/23.3	11.2/15.0	10.9/15.0	11.9/18.3	9.8/11.7	63.2/63.5	64.2/65.0	–	63.2/63.5	64.2/65.0	63.2/63.5	64.2/65.0
GAT	14.2/16.7	5.6/20.0	12.0/18.3	9.7/20.0	5.9/21.7	5.9/21.7	6.9/18.3	64.2/65.0	62.0/62.0	43.1/50.4	35.5/55.0	63.2/63.5	61.3/65.0	35.5/55.0
GIN	9.6/21.7	8.1/11.7	9.1/21.7	13.4/18.3	10.1/18.3	16.3/21.7	20.2/28.3	64.2/65.0	60.9/63.5	64.2/65.0	63.0/64.2	62.0/62.0	63.2/64.7	55.7/57.4
SAGE	5.6/8.3	6.0/21.7	8.3/20.0	5.6/20.0	8.5/18.3	5.6/20.0	5.9/21.7	62.0/62.0	63.6/63.7	62.0/62.0	60.9/61.6	63.7/65.5	62.9/63.0	62.0/62.0
Heterogeneous baselines														
R-GCN	7.9/16.7	9.0/16.7	10.4/16.7	9.7/20.0	8.6/20.0	6.8/10.0	5.9/21.7	64.8/65.7	63.9/64.5	62.0/62.0	64.2/65.0	63.2/63.5	58.6/64.7	63.2/63.5
HAN	26.6/28.3	41.6/41.7	–	31.6/33.3	46.1/45.0	38.3/38.3	19.1/25.0	–	74.7/75.2	76.1/76.2	78.0/78.1	75.4/75.4	76.4/76.4	75.1/75.2
HGT	–	40.1/40.0	36.1/36.7	48.3/48.3	39.4/43.3	9.3/18.3	13.4/28.3	71.6/71.8	67.8/68.1	79.3/79.3	58.7/63.7	65.0/65.0	66.3/67.9	–
HetSheaf (ours)														
Diag	31.0/40.0	46.1/46.7	34.2/35.0	19.3/28.3	45.2/45.0	48.3/50.0	53.0/53.3	72.1/72.1	78.2/78.2	73.7/73.7	74.5/74.5	68.5/68.5	76.8/76.8	61.8/61.8
Bundle	38.2/43.3	50.2/53.3	38.6/41.7	36.6/36.7	51.2/53.3	22.4/30.0	**62.3/63.3**	68.0/68.0	62.1/62.1	59.9/59.9	62.8/62.8	53.2/53.2	60.4/60.4	62.3/62.3
General	31.9/36.7	47.7/50.0	46.9/46.7	43.5/45.0	49.9/51.7	17.8/28.3	50.7/51.7	61.7/61.7	48.7/48.7	64.4/64.4	28.1/28.1	63.2/63.2	67.5/67.5	62.1/62.1
Diag-P	14.4/21.7	10.1/21.7	11.3/23.3	20.6/23.3	14.9/21.7	25.6/30.0	22.7/28.3	–	61.4/61.4	62.1/62.1	49.1/49.1	23.0/23.0	46.6/46.6	62.1/62.1
Bundle-P	46.5/48.3	56.7/56.7	52.6/53.3	51.9/51.7	32.4/35.0	50.0/50.0	34.6/35.0	69.2/69.2	63.9/63.9	76.3/76.3	68.6/68.6	74.2/74.2	73.0/73.0	62.1/62.1
Gen.-P	41.2/45.0	41.7/43.3	48.1/48.3	45.5/46.7	37.1/41.7	38.0/41.7	34.4/36.7	67.2/67.2	72.3/72.3	73.0/73.0	72.9/72.9	69.9/69.9	**79.5/79.5**	62.1/62.1
SheafAN	27.0/30.0	34.9/36.7	46.5/50.0	37.7/41.7	38.6/41.7	49.3/51.7	30.9/35.0	62.1/62.1	68.2/68.2	75.1/75.1	69.7/69.7	66.5/66.5	75.1/75.1	73.6/73.6

HETSHEAF obtains the best Macro and Micro F1 on ACM and DBLP, while on IMDB SeHGNN and HINormer are stronger, with HETSHEAF remaining competitive despite not relying on meta-path or transformer machinery.

C.3.2. CONTEXTUAL COMPARISONS FROM THE LITERATURE

We additionally collect results reported in the literature for a broader set of heterogeneous GNNs. Since these baselines were not re-run under a single implementation and, in some cases, use different train/validation/test splits, these comparisons should be interpreted as contextual rather than primary.

Among the additional baselines, HAGNN [79] is the strongest competitor on DBLP and IMDB under the HGB protocol. Although presented as a hybrid framework, it still relies on a fused meta-path graph in its first aggregation stage and therefore inherits core limitations of meta-path-based methods: it requires expert-selected meta-paths, restricts intratype aggregation to those manually specified relational templates, and remains bounded

Table 22: Link prediction: AUPR / AUROC (%) by model and number of layers L. Cyan = best in family, **bold** = global best per dataset.

	LastFM							MovieLens						
	$L=2$	$L=3$	$L=4$	$L=5$	$L=6$	$L=7$	$L=8$	$L=2$	$L=3$	$L=4$	$L=5$	$L=6$	$L=7$	$L=8$
Homogeneous baselines														
GCN	96.7/96.5	96.5/96.0	96.9/96.8	96.8/96.8	97.1/97.0	97.1/96.9	97.0/97.0	99.5/99.5	99.6/99.6	99.6/99.6	99.6/99.6	99.6/99.6	99.6/99.6	99.6/99.6
GAT	62.7/50.4	62.7/50.4	62.7/50.4	63.1/51.1	62.7/50.4	62.7/50.4	62.9/50.7	97.0/97.4	97.6/97.9	96.8/97.3	96.8/97.3	96.8/97.2	96.8/97.2	96.8/97.2
GIN	97.4/96.3	98.1/97.6	98.0/97.7	97.9/97.7	96.7/96.6	97.0/96.9	96.9/96.5	99.6/99.5	99.6/99.6	99.6/99.6	99.6/99.5	99.6/99.6	99.3/99.3	99.5/99.5
SAGE	92.7/89.2	88.0/86.4	83.5/80.1	95.1/92.7	90.0/86.4	88.5/86.5	88.0/86.6	98.8/98.8	98.8/98.8	99.1/99.0	99.2/99.1	99.1/99.0	99.2/99.0	99.2/99.1
Heterogeneous baselines														
R-GCN	97.8/97.8	97.7/97.7	97.8/97.8	98.0/97.9	97.9/97.9	97.8/97.8	97.8/97.9	99.4/99.4	99.4/99.4	99.4/99.4	99.3/99.3	99.3/99.3	99.2/99.2	99.3/99.3
HAN	75.8/73.4	86.2/83.0	76.7/74.2	88.3/83.3	86.6/81.8	87.5/82.8	87.2/82.7	63.0/51.1	63.0/51.1	63.0/51.1	63.6/52.2	63.1/51.2	63.1/51.3	63.1/51.4
HGT	83.8/71.3	86.1/74.1	85.8/74.5	85.9/74.4	86.0/74.5	86.1/74.5	85.9/74.4	68.9/61.7	70.8/63.6	63.8/52.6	77.7/74.0	67.6/58.9	63.0/51.1	63.0/51.0
HetSheaf (ours)														
Diag	98.4/98.1	98.4/98.2	98.4/98.2	**98.5**/98.2	98.5/98.2	98.4/98.2	98.4/98.2	**99.7**/99.6	**99.7**/99.6	**99.7**/99.6	**99.7**/99.6	**99.7**/99.6	**99.7**/99.6	**99.7**/99.6
Bundle	98.4/98.3	98.5/98.2	98.5/98.2	98.5/98.2	**98.5/98.3**	98.4/98.1	98.4/98.2	99.7/**99.6**	99.7/99.6	99.7/99.6	99.7/99.6	99.7/99.6	99.7/99.6	99.7/99.6
General	98.5/98.2	98.5/**98.3**	**98.5/98.3**	98.5/**98.3**	98.5/**98.3**	98.4/98.2	98.4/98.3	99.7/99.6	99.7/99.6	99.7/99.6	99.7/99.6	99.7/**99.6**	99.7/99.6	99.7/99.6
Diag-P	98.4/98.1	98.4/98.0	98.3/98.0	98.3/98.0	98.4/98.0	98.3/98.1	98.2/97.9	99.6/99.5	99.6/99.6	99.6/99.6	99.7/99.6	99.7/99.6	99.7/99.6	99.7/99.6
Bundle-P	98.3/97.8	98.3/98.0	98.4/98.1	98.3/98.1	98.2/98.0	98.2/97.9	98.2/97.8	99.7/99.6	99.7/99.6	99.7/99.6	99.7/99.6	99.7/99.6	99.7/99.6	99.7/99.6
Gen.-P	98.3/98.0	98.3/98.0	**98.5/98.3**	98.3/98.1	98.2/97.9	98.4/98.1	98.3/98.0	99.7/99.6	99.7/99.6	99.7/99.6	99.7/99.6	99.7/99.6	99.7/99.6	99.7/99.6
SheafAN	97.7/97.2	97.7/97.3	97.6/97.1	97.9/97.6	97.8/97.2	97.7/97.5	97.7/97.3	99.5/99.4	99.5/99.4	99.5/99.4	99.5/99.4	99.5/99.4	99.5/99.5	99.6/99.5

	Amazon							YouTube						
	$L=2$	$L=3$	$L=4$	$L=5$	$L=6$	$L=7$	$L=8$	$L=2$	$L=3$	$L=4$	$L=5$	$L=6$	$L=7$	$L=8$
Homogeneous baselines														
GCN	92.3/89.5	92.1/89.5	92.4/89.6	92.1/89.6	92.3/89.6	92.4/89.7	92.2/89.6	89.3/83.3	89.3/83.2	89.3/83.2	89.3/83.3	89.3/83.3	89.3/83.3	89.3/83.2
GAT	88.6/86.0	88.7/86.0	88.1/85.5	88.6/86.1	80.3/74.0	88.6/86.0	85.4/82.1	62.5/50.0	62.5/50.0	66.8/58.4	62.5/50.0	62.7/50.4	62.5/50.0	62.5/50.0
GIN	92.1/89.6	92.2/89.3	92.3/89.5	92.2/89.5	92.2/89.2	92.2/89.2	–	88.6/83.0	87.6/82.1	85.8/81.3	88.0/82.2	87.0/81.0	87.3/82.1	85.4/79.7
SAGE	89.3/86.2	90.0/87.1	90.8/88.0	90.3/87.3	89.9/87.0	90.4/87.5	90.4/87.5	71.4/66.2	81.4/75.6	84.5/80.2	83.1/78.9	78.5/75.9	79.2/76.1	85.6/79.2
Heterogeneous baselines														
R-GCN	93.4/91.4	92.9/91.0	92.8/91.1	92.8/90.9	87.2/83.3	92.1/90.1	88.6/86.0	91.7/88.3	91.6/88.2	91.7/88.2	91.5/88.2	91.4/88.1	91.4/88.0	91.4/88.1
HAN	94.0/92.3	94.9/93.4	94.3/92.6	91.3/88.7	94.1/92.3	95.2/93.6	94.9/93.1	90.3/87.0	91.1/86.8	93.5/90.1	92.2/88.3	90.0/85.8	90.0/85.2	89.1/84.4
HGT	96.0/95.3	96.1/95.6	95.6/95.3	95.9/95.4	96.1/95.7	96.2/95.6	95.9/95.5	94.2/91.5	94.6/91.8	95.0/92.5	94.5/91.6	94.2/91.6	94.6/91.7	94.1/91.6
HetSheaf (ours)														
Diag	**93.9**/91.6	93.8/91.5	93.8/91.6	93.8/91.5	93.8/91.5	93.9/91.6	**93.9**/91.6	89.5/83.5	90.6/85.4	90.7/86.0	90.4/85.5	90.5/85.7	90.5/85.4	89.4/83.4
Bundle	93.6/91.3	93.8/91.6	93.8/91.5	93.7/91.5	93.8/91.5	93.8/91.6	93.7/91.4	**90.7**/85.9	90.3/84.9	89.3/83.2	90.1/84.4	89.5/83.4	89.2/83.2	89.3/83.4
General	93.9/**91.7**	93.9/91.6	93.8/91.6	93.8/91.6	93.8/91.5	93.8/91.5	93.8/91.6	91.1/86.8	90.7/86.0	**91.3/87.3**	90.4/85.4	**91.3/87.3**	90.6/86.0	90.4/85.6
Diag-P	93.5/90.9	93.3/90.7	93.1/90.5	93.1/90.3	93.2/90.6	93.3/90.5	93.3/90.8	90.6/85.4	90.5/85.2	90.6/85.8	90.7/86.5	89.4/83.3	90.1/85.3	89.4/83.2
Bundle-P	93.5/91.1	93.6/90.9	93.5/91.1	93.5/91.1	93.5/91.1	93.6/91.1	93.6/91.2	89.4/83.3	89.4/83.3	89.4/83.3	89.4/83.4	89.4/83.2	89.4/83.3	89.4/83.3
Gen.-P	93.0/90.3	93.8/91.4	**93.8/91.7**	93.8/91.5	93.6/91.1	93.6/91.3	92.9/90.3	–	–	–	–	–	89.2/83.0	–
SheafAN	92.0/88.7	92.1/88.9	91.7/88.4	91.8/88.6	92.2/89.1	91.9/88.7	92.0/89.2	86.9/80.4	87.0/80.5	86.7/80.5	86.8/80.2	87.0/80.5	86.6/80.2	87.1/80.6

Table 23: *Matched-protocol node classification* against recent state-of-the-art heterogeneous GNNs (Macro-F1 / Micro-F1). Protocol as described above; colouring as in Table 2. Our seed-averaged values are in Table 2.

	ACM		DBLP		IMDB	
Model	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
HAGNN	**92.17**±0.35	**92.11**±0.33	**94.27**±0.56	**94.69**±0.51	**63.32**±0.94	**67.60**±0.39
SeHGNN	**92.95**±0.38	**92.87**±0.38	93.81±0.29	94.26±0.28	**66.87**±0.11	**69.52**±0.17
SlotGAT	91.73±0.48	91.62±0.49	**94.49**±0.37	**94.89**±0.37	61.59±0.59	66.27±0.55
Sheaf-NSD	92.07±2.09	91.90±2.09	92.91±1.21	93.15±1.18	61.94±0.34	65.10±0.20
HetSheaf (ours)	**94.30**±0.24	**94.22**±0.24	**94.93**±0.36	**95.27**±0.34	**64.51**±0.68	**66.57**±0.17

by how well the chosen meta-paths approximate the true semantics of the graph. By contrast, HetSheaf dispenses with meta-path engineering and models inter-type interactions directly on the original heterogeneous topology through learned restriction maps. As shown in Table 24, this yields competitive or superior performance without any hand-designed meta-path machinery.

Tables 24 and 25 collect these broader comparisons for node classification and link prediction, respectively.

Table 24: *Node classification against results reported in the literature* (Macro-F1 / Micro-F1); mean ± std as given by each source, colouring as in Table 2, "–" unavailable. Sources: † Lv et al. [45] and ‡ Zhu et al. [79], both HGB split (24/6/70 %); § Lin et al. [44], 20 % train split, so *not* directly comparable and reported without std.

	DBLP		IMDB		ACM	
Model	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
GTN‡	65.21±1.16	66.23±1.23	59.26±0.84	64.07±0.65	91.31±0.70†	91.20±0.71†
HetSANN‡	84.08±1.01	84.96±0.88	49.25±0.57	57.47±1.12	90.02±0.35†	89.91±0.37†
HetGNN‡	92.77±0.24	93.23±0.23	47.87±0.33	50.83±0.26	85.91±0.25†	86.05±0.25†
CKD‡	92.52±0.23	92.80±0.22	60.30±0.77	65.98±0.87	–	–
HGNN-AC‡	92.97±0.72	93.43±0.96	56.63±0.81	63.85±0.85	–	–
RSHN†	93.34±0.58	93.81±0.55	59.85±3.21	64.22±1.03	90.50±1.51	90.32±1.54
MAGNN‡	93.16±0.38	93.65±0.34	56.92±1.76	65.11±0.59	90.88±0.64†	90.77±0.65†
R-HGNN‡	93.31±0.42	93.80±0.39	61.39±0.51	66.03±0.65	–	–
AMHGNN‡	93.71±0.79	94.08±0.76	63.38±0.66	67.29±0.64	–	–
SimpleHGN‡	93.81±0.54	94.26±0.49	63.53±1.66	67.42±0.42	**93.42**±0.44†	**93.35**±0.45†
BPHGNN‡	93.89±0.43	94.50±0.47	64.01±0.56	**67.77**±0.70	–	–
HINormer‡	**94.57**±0.21	**94.94**±0.23	**64.65**±0.53	**67.83**±0.34	–	–
HAGNN‡	**95.06**±0.31	**95.40**±0.30	**65.57**±0.49	**68.62**±0.23	–	–
MRGCN§	89.53	90.51	45.62	47.78	87.64	87.52
SSDCM§	89.44	89.94	49.45	59.16	87.72	87.65
MHGCN§	90.90	92.17	50.54	64.20	88.96	89.19
HGATE§	90.25	91.15	41.99	52.10	88.73	88.70
MGDCR§	92.04	92.43	45.03	59.05	89.03	88.87
HMGE§	91.56	92.35	32.93	57.35	90.80	90.66
AMOGCN§	92.02	92.46	50.92	63.60	91.72	91.39
HGNN-AR^{2}§	92.54	93.02	52.51	61.36	**92.42**	**92.36**
Sheaf-NSD	92.91±1.21	93.15±1.18	61.94±0.34	65.10±0.20	92.07±2.09	91.90±2.09
HetSheaf (ours)	**94.93**±0.36	**95.27**±0.34	**64.51**±0.68	66.57±0.17	**94.30**±0.24	**94.22**±0.24

C.4. Implementation Details

We now collect the implementation details underlying all experiments. We begin with the hardware and software environment, then describe the downstream decoders and loss functions used for the three tasks, followed by the parameterisation of restriction maps and the full family of heterogeneous sheaf predictors. We then report the computational complexity

Table 25: *Link prediction against results reported in the literature* (AUROC); mean ± std as given by each source, colouring as in Table 2, "OOM" out of memory, "–" unavailable. Sources: † Lv et al. [45]; ‡ Zhu et al. [79].

Model	Amazon	LastFM	PubMed
MAGNN‡	–	56.81±0.05	OOM
HetGNN†	77.74±0.24	62.09±0.01	73.63±0.01
GATNE†	77.39±0.50	66.87±0.16	63.39±0.65
AutoAC‡	–	66.64±0.26	82.95±0.38
SimpleHGN†	**93.40±0.62**	67.16±0.37‡	83.39±0.39‡
HAGNN‡	–	**67.33±0.10**	**84.00±0.63**
Sheaf-NSD	**90.02±0.25**	**96.58±0.17**	**93.69±0.16**
HetSheaf (ours)	**91.71±0.37**	**98.33±0.35**	**93.82±0.68**

and overhead, and detail the heterogeneous extensions of Sheaf Attention Networks and Polynomial Neural Sheaf Diffusion used in our experiments.

C.4.1. Hardware and Software Setup

All experiments were conducted on an AWS EC2 g6.xlarge instance running Ubuntu 24.04 LTS, equipped with an NVIDIA L4 GPU (CUDA 12.8, cuDNN 9.1). The codebase is implemented in Python 3.13.2 using PyTorch 2.8 and PyTorch Geometric 2.7 [21, 22], with dependency management handled by `uv`[10]. Hyperparameter searches were performed via Weights & Biases[11] sweeps. These choices were kept fixed across models to ensure that empirical differences reflect modelling choices rather than software-stack variation.

C.4.2. Downstream Decoders and Loss Functions

The sheaf encoder produces node-level hidden states that are then consumed by a task-specific decoder. Since node classification, link prediction, and graph classification place different demands on the output space, we use standard downstream decoders tailored to each task while leaving the sheaf backbone unchanged.

Node classification. After the Sheaf GNN processes the graph, the final node embeddings are passed to an MLP to produce classification logits. For multiclass tasks, we use a softmax activation with cross-entropy loss; for multilabel settings, we use a sigmoid activation with binary cross-entropy loss.

Link prediction. For each node pair u, v and relation type r, the model outputs the score

$$P(u \sim_r v) = Decoder(\mathbf{H}_u, \mathbf{H}_v) \tag{75}$$

which gives the probability that u and v are connected by a relation of type r. Since this is a binary prediction problem, the loss is binary cross-entropy.

10. https://docs.astral.sh/uv/
11. https://wandb.ai

The decoder can take several forms. A dot-product decoder first computes the inner product and then applies a sigmoid:

$$Decoder(\mathbf{H}_u, \mathbf{H}_v) = \sigma(\langle \mathbf{H}_u, \mathbf{H}_v \rangle). \tag{76}$$

DistMult [71] instead computes the bilinear score

$$Decoder(\mathbf{H}_u, \mathbf{H}_v) = \sigma\Big(\mathbf{H}_u^{\intercal} \mathbf{W}_{\psi(u,v)} \mathbf{H}_v\Big), \tag{77}$$

where $\mathbf{W}_{\psi(u,v)} \in \mathbb{R}^{d_L \times d_L}$ is a learnable weight matrix for each edge type. Finally, a concat decoder concatenates the two embeddings, applies a one-layer MLP, and then a sigmoid:

$$Decoder(\mathbf{H}_u, \mathbf{H}_v) = \sigma(\mathrm{MLP}(\mathbf{H}_u \| \mathbf{H}_v)). \tag{78}$$

Following Lv et al. [45], we treat the choice of decoder as a hyperparameter.

C.4.3. Restriction Map Parametrisations

Each restriction map is represented as a $d \times d$ block matrix and parametrised by the matrix-valued function $\Phi(\mathbf{x}_u, \mathbf{x}_v, \phi(u), \phi(v), \psi(e))$. We briefly describe how each family is constructed.

Diagonal restriction maps. The predictor outputs d values, which define the diagonal entries of the restriction map.

General restriction maps. The predictor outputs d^2 values, which are reshaped into a full $d \times d$ matrix.

Orthogonal restriction maps. For orthogonal restriction maps $\mathcal{F}_{\trianglelefteq \cdot} \in \mathrm{O}(d)$, we consider two parameterisations. The first uses Householder products: the MLP outputs $\lfloor \frac{d(d-1)}{2} \rfloor$ parameters, which are converted by the Torch Householder library [51] into a product of Householder reflections covering the full orthogonal group $\mathrm{O}(d)$. The second uses the Cayley transform: the MLP outputs d^2 values forming a matrix $\mathcal{F}'$, from which the restriction map is obtained as

$$\mathcal{F}_{\trianglelefteq \cdot} = (\mathbf{I}_d - \mathbf{S})(\mathbf{I}_d + \mathbf{S})^{-1}, \qquad \mathbf{S} = \tfrac{1}{2}(\mathcal{F}' - \mathcal{F}'^{\top}). \tag{79}$$

Since the Cayley transform maps skew-symmetric matrices to $\mathrm{SO}(d)$, it can represent rotations but not reflections. In practice, the Householder and Cayley parameterisations provide different geometric biases, and the latter is the default for the attention predictors and SheafAN variant following Barbero et al. [4].

C.4.4. Full Description of Heterogeneous Sheaf Predictors

We next give the complete definition of the heterogeneous predictor family used in the experiments. These variants differ in two ways: first, in *which* typed signals are provided to the predictor (node features, node types, edge types, or combinations thereof), and second, in *how* this information is routed (shared predictor versus edge-type-specific ensemble). This family allows us to disentangle the contribution of feature-level conditioning from that of pure structural-type information.

Let $\mathcal{G}$ be a heterogeneous graph with node features $\mathbf{x}_u \in \mathbb{R}^f$, one-hot node-type vector $\mathbf{e}_{\phi(u)}$, and one-hot edge-type vector $\mathbf{e}_{\psi(e)}$ for edge $e := (u, v)$ and $\mathbf{z}_u = \mathcal{P}_{\phi(u)}(\mathbf{x}_u)$, $\mathbf{z}_v =$

Table 26: Summary of Heterogeneous Sheaf Predictors and their inputs.

Predictor Name	Acronym Meaning	Inputs Used
Sheaf-NSD	**N**eural **S**heaf **D**iffusion	$\mathbf{z}$
HetSheaf-TE	**T**ype **E**mbeddings	$\mathbf{z}, \phi, \psi$
HetSheaf-NE	**N**ode **E**mbeddings	$\mathbf{z}, \phi$
HetSheaf-EE	**E**dge **E**mbeddings	$\mathbf{z}, \psi$
HetSheaf-types	All **types** only	ϕ, ψ
HetSheaf-NT	**N**ode **T**ypes only	ϕ
HetSheaf-ET	**E**dge **T**ypes only	ψ
HetSheaf-ensemble	**Ensemble** of MLPs	$\mathbf{z}$ (via $\mathrm{MLP}_{\psi(e)}$)

$\mathcal{P}_{\phi(v)}(\mathbf{x}_v)$ the projections in the shared channel. All predictors below are compatible with diagonal, orthogonal, and general restriction-map types.

The predictors are broadly categorised by whether they append type embeddings to node features, discard features in favour of pure type signals, or use specialised routing through edge-type-specific experts.

Sheaf-NSD.

$$\mathcal{F}_{u \trianglelefteq e} \coloneqq \mathrm{MLP}(\mathbf{z}_u \| \mathbf{z}_v). \tag{80}$$

HetSheaf-TE.

$$\mathcal{F}_{u \trianglelefteq e} \coloneqq \mathrm{MLP}\Big(\mathbf{z}_u \| \mathbf{z}_v \| \mathbf{e}_{\phi(u)} \| \mathbf{e}_{\phi(v)} \| \mathbf{e}_{\psi(e)}\Big). \tag{81}$$

HetSheaf-NE.

$$\mathcal{F}_{u \trianglelefteq e} \coloneqq \mathrm{MLP}\Big(\mathbf{z}_u \| \mathbf{z}_v \| \mathbf{e}_{\phi(u)} \| \mathbf{e}_{\phi(v)}\Big). \tag{82}$$

HetSheaf-EE.

$$\mathcal{F}_{u \trianglelefteq e} \coloneqq \mathrm{MLP}\Big(\mathbf{z}_u \| \mathbf{z}_v \| \mathbf{e}_{\psi(e)}\Big). \tag{83}$$

HetSheaf-types.

$$\mathcal{F}_{u \trianglelefteq e} \coloneqq \mathrm{MLP}\Big(\mathbf{e}_{\phi(u)} \| \mathbf{e}_{\phi(v)} \| \mathbf{e}_{\psi(e)}\Big). \tag{84}$$

HetSheaf-NT.

$$\mathcal{F}_{u \trianglelefteq e} \coloneqq \mathrm{MLP}\Big(\mathbf{e}_{\phi(u)} \| \mathbf{e}_{\phi(v)}\Big). \tag{85}$$

HetSheaf-ET.

$$\mathcal{F}_{u \trianglelefteq e} \coloneqq \mathrm{MLP}\Big(\mathbf{e}_{\psi(e)}\Big). \tag{86}$$

HetSheaf-ensemble.

$$\mathcal{F}_{u \trianglelefteq e} \coloneqq \mathrm{MLP}_{\psi(e)}(\mathbf{z}_u \| \mathbf{z}_v), \tag{87}$$

where $\psi(e)$ selects the edge-type-specific predictor. This yields strict expressivity gains over a single shared MLP at the cost of parameters that scale linearly with the number of edge types.

C.4.5. HetSheaf Computational Complexity and Overhead

We derive the per-layer computational complexity of each HetSheaf predictor following the analysis framework of Bodnar et al. [6]. The aim is twofold: first, to make explicit how type conditioning changes the asymptotic cost relative to vanilla NSD; and second, to quantify the practical overhead of the sheaf-based approach relative to strong non-sheaf baselines.

Consider a heterogeneous graph with n nodes, m edges, s node types, and t edge types. Let d denote the stalk dimension and f the number of feature channels, so that $c = d \times f$. We distinguish between diagonal restriction maps, which require only d parameters per map, and non-diagonal maps, which require d^2 parameters and may incur an additional $\mathcal{O}(d^3)$ orthogonalisation cost in the orthogonal case. Table 27 summarises the resulting per-layer complexities.

Table 27: **Per-layer computational complexities of heterogeneous sheaf predictors and baselines.** Notation: n nodes, m edges, s node types, t edge types, stalk dimension d, channels f, $c = df$ (or $c = f$ for non-sheaf models), attention heads h. Diagonal restriction maps are denoted (diag); general or orthogonal maps are denoted (non-diag).

Model	**Computational complexity**
Non-sheaf baselines	
GCN	$\mathcal{O}(nc^2 + mc)$
GAT	$\mathcal{O}(nc^2h + mch)$
HAN	$\mathcal{O}(tnc^2h + tmch)$
HetSheaf predictors (diag)	
NSD	$\mathcal{O}(nc^2 + mdc)$
TE	$\mathcal{O}(nc^2 + md(c + s + t))$
NE	$\mathcal{O}(nc^2 + md(c + s))$
EE	$\mathcal{O}(nc^2 + md(c + t))$
types	$\mathcal{O}(nc^2 + md(s + t))$
NT	$\mathcal{O}(nc^2 + mds)$
ET	$\mathcal{O}(nc^2 + mdt)$
ensemble	$\mathcal{O}(nc^2 + mdct)$
HetSheaf predictors (non-diag)	
NSD	$\mathcal{O}(n(c^2 + d^3) + md^2(c + d))$
TE	$\mathcal{O}(n(c^2 + d^3) + md^2(c + s + t + d))$
NE	$\mathcal{O}(n(c^2 + d^3) + md^2(c + s + d))$
EE	$\mathcal{O}(n(c^2 + d^3) + md^2(c + t + d))$
types	$\mathcal{O}(n(c^2 + d^3) + md^2(s + t))$
NT	$\mathcal{O}(n(c^2 + d^3) + md^2s)$
ET	$\mathcal{O}(n(c^2 + d^3) + md^2t)$
ensemble	$\mathcal{O}(n(c^2 + d^3) + md^2(c + dt))$

Several observations follow. First, the additive cost introduced by type-aware predictors relative to NSD is governed by the number of appended type indicators, and is therefore typically negligible since heterogeneous benchmarks usually satisfy $s, t \ll c$. Second, diagonal

maps scale linearly in d per edge, while general or orthogonal maps scale quadratically in d because of the full matrix-valued transport. Third, the ensemble predictor is the only variant whose cost scales multiplicatively with the number of edge types, reflecting the use of independent edge-type-specific MLPs. Even so, for the small stalk dimensions and modest type cardinalities used in this work, the practical overhead remains limited. Finally, regarding scalability with the graph size: for fixed stalk dimension d and channel size f, every diagonal HetSheaf predictor in Table 27 runs in $\mathcal{O}(nc^2 + md(c+s+t)) = \mathcal{O}(n+m)$ time per layer, where n and m are the numbers of nodes and edges and s and t the numbers of node and edge types. The type conditioning we introduce therefore enters only as an *additive edge-wise cost* governed by $s+t$, rather than as a factor that grows with the graph size. While the largest benchmark evaluated in this work contains roughly 63K nodes, this linear scaling makes the diagonal variants natural candidates for much larger graphs; an empirical validation at million-node scale is left to future work.

To complement the asymptotic analysis, Table 28 reports parameter counts and wall-clock costs on the node classification benchmarks. Despite not yet using the most heavily optimised PyG kernels, the sheaf-based models remain substantially smaller than the strongest heterogeneous baselines in terms of parameters, while often matching or surpassing them in predictive performance.

Table 28: **HetSheaf computational overhead on node classification tasks.** The highest performing heterogeneous baseline architectures are highlighted with the highest performing HetSheaf model shown in **bold**.

	ACM		DBLP		IMDB	
	Runtime (s)	# params	Runtime (s)	# params	Runtime (s)	# params
R-GCN	22.1	43M	30.3	87.8M	25.9	72.1M
HAN	28.9	2.2M	18.5	1.9M	15.8	3.4M
HGT	1025.8	7.2M	648.3	6.2M	939.9	10.5M
Sheaf-NSD	47.6	155K	73.3	1.7M	40.5	1.6M
HetSheaf-TE (ours)	44.7	155K	**79.1**	**1.7M**	40.1	1.6M
HetSheaf-ensemble (ours)	55.7	174K	83.2	1.8M	44.9	1.9M
HetSheaf-NE (ours)	47.3	155K	89.6	1.7M	12.0	1.1M
HetSheaf-EE (ours)	**49.6**	**155K**	85.9	1.7M	41.8	1.6M
HetSheaf-NT (ours)	42.0	154K	82.4	1.7M	**38.6**	**1.6M**
HetSheaf-ET (ours)	45.1	154K	80.5	1.7M	39.8	1.6M

C.4.6. Heterogeneous Sheaf Attention Networks

Sheaf Attention Networks (SheafANs) [4] replace Laplacian-based propagation with attention-weighted aggregation over sheaf transports. Since HetSheaf is intended to be model-agnostic, it is important to verify that the inferred heterogeneous sheaf can be consumed not only by NSD-style diffusion layers but also by attention-based sheaf architectures. This subsection, therefore, describes how SheafAN is extended to the heterogeneous setting.

Their geometric backbone is the sheaf adjacency operator $\hat{A}_{\mathcal{F}} \in \mathbb{R}^{nd\times nd}$, whose off-diagonal blocks are defined as

$$P_{ij} = \mathcal{F}_{i\trianglelefteq e}^{\top}\mathcal{F}_{j\trianglelefteq e} \in \mathbb{R}^{d\times d}, \tag{88}$$

for $e = (i, j)$. In the original homogeneous formulation, the scalar attention coefficients are

$$\Lambda_{ij} = \frac{\exp\Big(\text{LeakyReLU}\Big(\mathbf{a}^{\top}[\mathbf{W}\mathbf{x}_i \| \mathbf{W}\mathbf{x}_j]\Big)\Big)}{\sum_{k\in\mathcal{N}_i}\exp(\text{LeakyReLU}(\mathbf{a}^{\top}[\mathbf{W}\mathbf{x}_i \| \mathbf{W}\mathbf{x}_k]))}. \tag{89}$$

To adapt SheafAN to heterogeneous graphs, we consider the channel projection $\mathbf{z}_* = \mathcal{P}_{\phi(*)}(\mathbf{x}_*)$ and augment the attention input with node- and edge-type embeddings:

$$\Lambda_{ij} = \frac{\exp\Big(\text{LeakyReLU}\Big(\mathbf{a}^{\top}[\mathbf{W}\mathbf{z}_i \| \mathbf{W}\mathbf{z}_j \| \mathbf{e}_{\phi(i)} \| \mathbf{e}_{\phi(j)} \| \mathbf{e}_{\psi((i,j))}]\Big)\Big)}{\sum_{k\in\mathcal{N}_i}\exp\Big(\text{LeakyReLU}\Big(\mathbf{a}^{\top}[\mathbf{W}\mathbf{z}_i \| \mathbf{W}\mathbf{z}_k \| \mathbf{e}_{\phi(i)} \| \mathbf{e}_{\phi(k)} \| \mathbf{e}_{\psi((i,k))}]\Big)\Big)}. \tag{90}$$

At the same time, the transport maps P_{ij} are computed from the heterogeneous restriction maps inferred by HetSheaf. Hence, both geometric transport and attention become aware of the typed structure of the graph, while the overall attentive sheaf architecture itself remains unchanged.

C.4.7. Heterogeneous Polynomial Neural Sheaf Diffusion

Polynomial Neural Sheaf Diffusion (PolyNSD) [7] extends first-order sheaf diffusion by replacing repeated local propagation with a learnable spectral polynomial of the sheaf Laplacian. Since PolyNSD is structurally quite different from NSD, its heterogeneous extension provides a strong demonstration of the model-agnosticity of HetSheaf.

For a sheaf Laplacian $L_{\mathcal{F}}$ with spectral decomposition $L_{\mathcal{F}} = U\Lambda U^{\top}$, a polynomial filter $p(L_{\mathcal{F}}) = \sum_{k=0}^{K} c_k L_{\mathcal{F}}^k$ acts diagonally in the sheaf Fourier basis, scaling the i-th mode by $p(\lambda_i)$. PolyNSD uses a Chebyshev parameterisation on a rescaled operator

$$\tilde{L}_{\mathcal{F}} = \frac{2}{\lambda_{\max}} L_{\mathcal{F}} - \mathbf{I}, \tag{91}$$

so that the polynomial remains numerically stable on $[-1, 1]$. The filter is then written as

$$p(\tilde{L}_{\mathcal{F}}) = \sum_{k=0}^{K} \alpha_k T_k(\tilde{L}_{\mathcal{F}}), \qquad \{\alpha_k\} = \text{softmax}(\boldsymbol{\eta}) \tag{92}$$

with the Chebyshev basis evaluated through the usual three-term recurrence.

Our framework extends PolyNSD to heterogeneous graphs without altering this spectral machinery. The only difference is that the Laplacian is now built from a heterogeneous sheaf whose restriction maps are predicted by

$$\mathcal{F}_{u\trianglelefteq e:=(u,v)} = \mathbf{\Phi}(\mathbf{x}_u, \mathbf{x}_v, \phi(u), \phi(v), \psi(e)) \tag{93}$$

The resulting heterogeneous sheaf Laplacian $L_{\mathcal{F}}$ is then used directly inside the polynomial filter. In this way, the same higher-order spectral diffusion mechanism extends naturally to the heterogeneous setting, again illustrating that once heterogeneity is absorbed into the inferred sheaf, the downstream sheaf architecture itself does not require ad hoc redesign.

Appendix D. Limitations

While HetSheaf provides a robust topological framework for learning on heterogeneous graphs, it possesses a few notable limitations. First, the computational complexity of the learned restriction maps scales quadratically with their dimension d when using general or bundle parametrisations. Furthermore, these stalks have a fixed dimension d across the entire network for all node and edge types, which may restrict the model's capacity to fully capture the varying complexities of a heterogeneous environment, constituting a well-known limitation of the Sheaf Laplacian. Second, the theoretical guarantees of the SheafPool gauge canonicalisation rely on the assumption that node embeddings possess full row rank, while in practice, network layers may induce rank-deficient stalks, necessitating a regularisation term ϵI_d. Consequently, the whitening operation acts as a stable approximation rather than an exact $O(d)$ canonicalisation.

D.1. Broader Impact

HetSheaf and SheafPool provide a general framework for learning on heterogeneous relational data by representing type-dependent structure through cellular sheaves rather than through increasingly specialised architectures. This may have positive impact in domains where heterogeneous graphs naturally arise, including biology, drug discovery, recommender systems, social networks, and communication networks. In these settings, modelling different node and edge types through type-aware local spaces may lead to more accurate and parameter-efficient systems, reducing the computational cost of training and deployment. At the same time, heterogeneous graph learning methods can also have negative societal impacts when applied to sensitive relational data. In recommender systems or social networks, improved relational modelling may amplify existing biases, reinforce filter bubbles, or enable more effective profiling of users. In biomedical or decision-support settings, inaccurate predictions or biased training data may lead to unreliable conclusions if the model is used without domain-specific validation. Moreover, because heterogeneous graphs often encode rich relational information, privacy risks may arise if sensitive node or edge attributes are present. Our work is primarily methodological and does not target a specific high-stakes deployment. Nevertheless, practical use of HetSheaf should be accompanied by dataset-specific fairness, privacy, robustness, and interpretability analyses. In particular, when applied to human-centred domains, users should carefully assess whether type information introduces or amplifies protected-attribute correlations, and whether the resulting predictions are reliable across different subgroups.