

Implicit Sheaf Operators for Effective and Efficient Cellular Sheaf Learning

Gabriele Onorato^{1*}, Dario Loi¹, Cristian Apostol¹, Nathaniel D. Bastian², Francesco Restuccia¹

¹Northeastern University, United States ²US Army

Abstract

While Sheaf Neural Networks (SNNs) enrich n -node m -edge diffusion of f -dimensional node features with d -dimensional node and edge stalks, they require computing a sparse $nd \times nd$ sheaf Laplacian, whose computation is prohibitive on edge devices. This paper addresses this issue with *Implicit Sheaf Operators (ISOs)*, a type of sheaf topological operators that leverage the topological structure of cellular sheaves to decompose the global diffusion process into local operator-vector products. We mathematically prove that ISO eliminates the $\mathcal{O}(md^2 \log(md^2))$ sparse-coalescing term and reduces the space complexity by $\mathcal{O}(\min(d, f))$ with respect to traditional Neural Sheaf Diffusion (NSD). Our experiments indicate that ISO-based operators achieve latency reduction of up to $4.87\times$ and save up to $3.92\times$ peak GPU memory. On real graph classification data, ISO uses up to $1.24\times$ less energy and $3.08\times$ less on-device memory as measured on a Jetson Orin Nano. Finally, to bring ISO to practice, we introduce Sheaf Convolutional Networks (SCNs), an ISO-based SNN attaining up to 3.93% higher accuracy compared to NSD-based SNNs on node classification benchmarks.

Introduction

Graph Neural Networks (GNNs) model relational data by encoding pairwise relational structures (Duvenaud et al. 2015; Hamilton, Ying, and Leskovec 2017; Ying et al. 2018; Ferriol-Galmés et al. 2023; Gori, Monfardini, and Scarselli 2005; Scarselli et al. 2009; Wu et al. 2021). Designing GNNs requires *permutation-equivariance*, i.e., neural network layers that would ignore the ordering of nodes in a graph, thus achieving *information diffusion* (Kondor and Lafferty 2002) through the continuous Laplace operator (Chen et al. 2018; Chamberlain et al. 2021; Thorpe et al. 2022). On graphs, continuous diffusion becomes the discrete Graph Laplacian (Chung 1997), whose localized first-order approximations underpin Message Passing Neural Networks (MPNNs) (Gilmer et al. 2017; Veličković et al. 2018; Xu et al. 2019), including Graph Convolutional Networks (GCNs) (Kipf and Welling 2017). Sheaf Neural Networks (SNNs) augment nodes and edges with d -dimensional stalks and restriction maps to mitigate oversmoothing (Hansen and Gebhart 2020; Rosiak 2022).

Key Issue. Existing Neural Sheaf Diffusion (NSD) methods (Bodnar et al. 2022; Suk et al. 2022; Zaghen et al. 2024; Barbero et al. 2022b; Caralt et al. 2024) learn an n -node m -edge sheaf for f -dimensional node features by constructing an $nd \times nd$ block-sparse Sheaf Laplacian (SL) (Hansen and Ghrist 2019a; Barbero et al. 2022a) at each SNN layer. Its $\mathcal{O}(md^2 f)$ memory cost and high computational complexity create severe bottlenecks on edge devices (Hanks et al. 2025; Seely, Cupiał, and Jones 2026), especially for inductive tasks where the operator must be rebuilt for every input graph (Braithwaite et al. 2026). This shift toward block-matrix multiplications increases computing overhead and hinders scalable deployment, as the standard message-passing abstraction can no longer be used without substantial modification. Figure 1 shows that explicitly materializing the sheaf operator causes NSD to scale poorly with the number of edges, incurring up to $39.2\times$ higher runtime and $9.27\times$ greater memory usage than GCN.

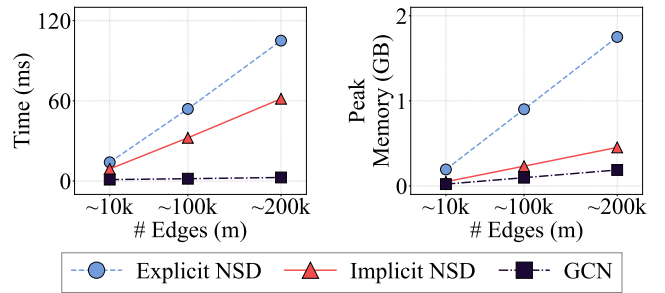


Figure 1: Compute time (**Left**) & peak memory (**Right**) usage of Bodnar et al. (2022)’s General NSD with $d = 8$ and $f = 16$ compared with Implicit Sheaf Operator (ISO) and GCN (Kipf and Welling 2017) with $h = 128$ on an *inductive inference* task on Erdős-Rényi graphs with $n = 1000$ nodes.

Proposed Approach. To address the key issue described above, we propose ISOs, a new approach to effective and efficient sheaf learning (see Figure 2). ISO introduces the concept of *implicit sheaf operators*, which leverage the sheaf structure to decompose the process into local operator-vector products directly from restriction maps. While Hajij et al. (2025) formalized message passing (Gilmer et al. 2017) on related structures known as copresheaves over topological domains, we point out that extending this framework to

*Corresponding author.

cellular sheaves introduces substantial challenges compared with conventional GCNs (Kipf and Welling 2017). In a GCN, the Laplacian or adjacency matrix is fixed by the graph topology, and message passing therefore relies on scalar edge weights. Cellular sheaves, by contrast, associate each incidence relation with a learned $d \times d$ restriction map. As a result, the corresponding propagation operators must be constructed separately for each graph and recomputed at every gradient step. ISO addresses this key bottleneck by replacing the explicit sparse block-matrix assembly with local edge-wise batched matrix multiplications (in other words, gather-scatter message passing operations) computed directly over the restriction maps. Theorems 3 and 8 prove that ISO bypasses the instantiation of $nd \times nd$ sparse matrices and achieves up to $4.87\times$ less latency and up to $3.92\times$ less peak GPU memory. To bring ISO to practice, we introduce SHEAF CONVOLUTIONAL NETWORKS (SCNs), an alternative formulation of SNNs that leverages the sheaf adjacency operator and achieves up to 3.93% higher accuracy on the node classification benchmark introduced by Pei et al. (2020).

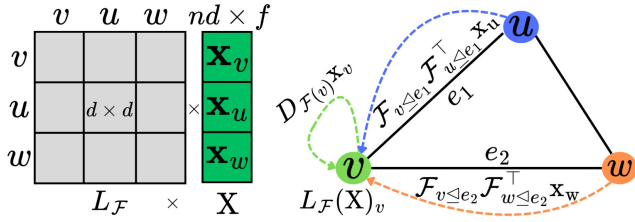


Figure 2: (Left) The Sheaf Laplacian (SL) $L_F \in \mathbb{R}^{nd \times nd}$ is explicitly computed before multiplying it with the sheaf node features X . (Right) The proposed ISOs reformulate the computation as a purely local process. By leveraging the topological features of cellular sheaves, ISO computes the exact action of $L_F X$ independently at each single node v , bypassing the global matrix completely.

Summary of Novel Contributions:

- **Implicit Sheaf Operators:** We introduce two ISOs, Implicit Sheaf Laplacian (ISL) and Implicit Sheaf Adjacency (ISA), which leverage the topological structure of cellular sheaves to decompose the global diffusion process into local operations directly from restriction maps. We prove ISL and ISA to be equivalent to explicit sheaf operators (Theorems 3 and 8, respectively).
- **Space & Time-Complexity Reduction:** Since ISL and ISA avoid the cost of sparse-matrix coalescing by never materializing the $nd \times nd$ block matrix, we prove that this approach eliminates the $\mathcal{O}(md^2 \log(md^2))$ sparse coalescing term (Theorems 4 and 9) and reduces space complexity by $\mathcal{O}(\min(d, f))$ (Theorems 5 and 10).
- **Memory and Power Efficiency:** Compared with explicit implementations, ISOs use $3.08\times$ less on-device memory and have up to $1.24\times$ lower energy consumption, as measured on a Jetson Orin Nano.
- **Experimental Latency Reduction:** On Erdős-Rényi random graphs (Lyzinski, Fishkind, and Priebe 2014), ISL and ISA provide a speedup of up to $4.87\times$, in the best

case, from 37.78ms to 7.76ms, and they reduce peak GPU memory by up to $3.92\times$ (1816MB vs 463MB).

- **Sheaf Convolution:** We introduce SHEAF CONVOLUTIONAL NETWORKS based on ISA, which we show achieve up to 3.93% higher accuracy on Pei et al. (2020) node classification benchmarks compared to NSD.

Related Work

From spectral to spatial GNNs. Early spectral approaches defined convolution filters in the eigenbasis of the graph Laplacian (Bruna et al. 2014), providing a principled analogue of Euclidean convolutions at the cost of a full $\mathcal{O}(n^2)$ eigendecomposition. Polynomial approximations such as ChebNet (Defferrard, Bresson, and Vandergheynst 2016) truncate the spectral filter to a K -th order Chebyshev expansion, while GCN (Kipf and Welling 2017) further restricts to first-order neighborhoods, thus yielding a localized spatial rule that scales linearly in the number of edges. This first-order localization shows GCN layers can be formulated as a *message-passing* step, an observation formalized by Gilmer et al. (2017) into a unified framework that subsumes a broad class of architectures: Graph Attention Network (GAT) (Veličković et al. 2018) introduces attention-weighted aggregation, GraphSAGE (Hamilton, Ying, and Leskovec 2017) extends the paradigm to the inductive setting by learning aggregation functions that generalize to unseen nodes and graphs, Graph Isomorphism Network (GIN) (Xu et al. 2019) introduces the concept of maximally powerful GNN via 1-dimensional Weisfeiler-Lehman (1-WL) expressivity, i.e., the ability for GNNs to distinguish non-isomorphic graphs, thus enabling the mapping of distinct topological structures to unique latent representations.

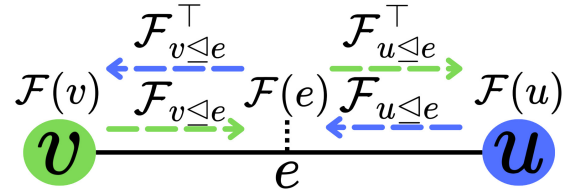


Figure 3: A *cellular sheaf* with stalks $\mathcal{F}(u), \mathcal{F}(v), \mathcal{F}(e)$ for nodes u, v and edge e . Vectors over stalks x_u, x_v are transported over the cellular sheaf via *restriction maps* using composition $\mathcal{F}_{v \leq e}^\top \mathcal{F}_{u \leq e}$.

Sheaf Neural Networks. As shown in Figure 3, cellular sheaves enrich graph topology by attaching a d -dimensional stalk to each node and edge, together with linear restriction maps between adjacent stalks (Curry 2014; Shepard 1985; Rosiak 2022). The resulting Sheaf Laplacian $L_F \in \mathbb{R}^{nd \times nd}$ generalizes the Graph Laplacian $L \in \mathbb{R}^{n \times n}$ and was first studied by Hansen and Ghrist (2019a), who later introduced the first SNN using a pre-defined sheaf structure (Hansen and Gebhart 2020). Bodnar et al. (2022) then proposed NSD, which learns the restriction maps end-to-end via a Multi-Layer Perceptron (MLP) on incident node features, thus adapting the sheaf structure to the task. Barbero et al. (2022a) introduced an efficient way to pre-instantiate the SL in transductive settings, while Suk et al. (2022) developed a second-order sheaf process called Neural Sheaf Propagation

(NSP) and Barbero et al. (2022b) introduced Sheaf Attention Networks (SheafANs), combining sheaf diffusion with multi-head attention (Vaswani et al. 2017). Zaghen et al. (2024) extended the diffusion operator beyond the linear regime by introducing a nonlinear activation inside the Laplacian update, applied on the coboundary operator. Beyond graphs, Hajij et al. (2025) recently formalized message passing on copresheaves over topological domains. While offering a dual algebraic perspective, copresheaf networks still rely on explicit high-dimensional block matrix representations. Consequently, all these implementations share the same *computational, memory and power bottleneck*. We address this limitation for the first time both theoretically and practically via ISOs, reducing peak GPU memory by up to $3.92\times$, inference time by up to $4.87\times$ and consuming up to $1.24\times$ less energy.

Preliminary Notions

Cellular Sheaves. Let us define an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with arbitrary edge orientations. A sheaf assigns a vector space $\mathcal{F}(u) \in \mathbb{R}^d$ to each node $u \in \mathcal{V}$ and $\mathcal{F}(e) \in \mathbb{R}^d$ to each edge $e \in \mathcal{E}$, called *stalks*, together with *restriction maps* $\mathcal{F}_{u \leq e} : \mathcal{F}(u) \rightarrow \mathcal{F}(e)$ for each incident pair $u \leq e$. Stacking vertex and edge stalks yields the 0- and 1-cochain spaces $C^0(\mathcal{G}, \mathcal{F}) := \bigoplus_{u \in \mathcal{V}} \mathcal{F}(u)$ and $C^1(\mathcal{G}, \mathcal{F}) := \bigoplus_{e \in \mathcal{E}} \mathcal{F}(e)$, respectively. Given a chosen orientation $e = (u \triangleleft v)$, the *coboundary map* $\delta : C^0(\mathcal{G}, \mathcal{F}) \rightarrow C^1(\mathcal{G}, \mathcal{F})$ acts edgewise as $\delta(\mathbf{x})_e := \mathcal{F}_{u \leq e} \mathbf{x}_u - \mathcal{F}_{v \leq e} \mathbf{x}_v$ and measures *disagreement* across edge-connected nodes (Hansen and Ghrist 2021). The *sheaf Laplacian* (Hansen and Ghrist 2019a) is defined as $L_{\mathcal{F}} := \delta^\top \delta$, and admits the node-wise form: $L_{\mathcal{F}}(\mathbf{x})_u = \sum_{u, v \leq e} \mathcal{F}_{u \leq e}^\top (\mathcal{F}_{u \leq e} \mathbf{x}_u - \mathcal{F}_{v \leq e} \mathbf{x}_v)$ and is positive semi-definite with block diagonal entries $L_{\mathcal{F}uu} = \sum_{u \leq e} \mathcal{F}_{u \leq e}^\top \mathcal{F}_{u \leq e}$ and off-diagonal blocks $L_{\mathcal{F}uv} = -\mathcal{F}_{u \leq e}^\top \mathcal{F}_{v \leq e}$. Its normalized version is $\Delta_{\mathcal{F}} := D_{\mathcal{F}}^{-1/2} L_{\mathcal{F}} D_{\mathcal{F}}^{-1/2}$, where $D_{\mathcal{F}}$ is the sheaf degree matrix defined nodewise as $D_{\mathcal{F}(u)} = \sum_{e: u \leq e} \mathcal{F}_{u \leq e}^\top \mathcal{F}_{u \leq e}$, which when $\mathcal{F}_{u \leq e} \in O(d)$ reduces to $D_{\mathcal{F}(u)} = I_d \cdot \deg(u)$. Moreover, we define the *sheaf adjacency* blockwise as $A_{\mathcal{F}u \leq v} := \mathcal{F}_{u \leq e}^\top \mathcal{F}_{v \leq e}$, with $L_{\mathcal{F}} = D_{\mathcal{F}} - A_{\mathcal{F}}$. Unless otherwise stated, we set all stalks to $\mathbb{R}^d$, so that $L_{\mathcal{F}}, A_{\mathcal{F}} \in \mathbb{R}^{nd \times nd}$. As such, when all restriction maps are the identity and $d = 1$, all the above operators reduce to traditional graph operators.

Neural Sheaf Diffusion. Neural Sheaf Diffusion (NSD) (Bodnar et al. 2022) learns the sheaf structure from data. Node features $\mathbf{x} \in \mathbb{R}^f$ are projected onto d -dimensional stalk signals and stacked into a 0-cochain $\mathbf{X} \in \mathbb{R}^{nd \times f}$ with f feature channels. The restriction maps at layer k are predicted from incident node features $e = (u, v)$ via a learnable linear function $\mathcal{F}_{u \leq e}^{(k)} = \Phi^{(k)}(\mathbf{x}_u, \mathbf{x}_v) = \text{MLP}(\mathbf{x}_u \parallel \mathbf{x}_v)$, where $\parallel$ denotes concatenation. These maps can produce diagonal, bundle (also called orthogonal, or $O(d)$), or general $d \times d$ matrices, which are used to construct $\Delta_{\mathcal{F}^{(k)}}$ and perform the diffusion step:

$$\mathbf{Z}^{(k)} = (\mathbf{I}_n \otimes \mathbf{W}_1^{(k)}) \mathbf{X}^{(k)} \mathbf{W}_2^{(k)}, \quad (1)$$

$$\mathbf{X}^{(k+1)} = (1 + \varepsilon^{(k)}) \odot \mathbf{X}^{(k)} - \sigma(\Delta_{\mathcal{F}^{(k)}} \mathbf{Z}^{(k)}). \quad (2)$$

where $\mathbf{W}_1^{(k)}$ and $\mathbf{W}_2^{(k)}$ are trainable weight matrices. As shown by Bodnar et al. (2022), this process converges to the *global section* of the graph (Hansen and Ghrist 2019b), defined as $H^0(\mathcal{G}; \mathcal{F}) := \{\mathbf{x} \in C^0(\mathcal{G}, \mathcal{F}) : \mathcal{F}_{v \leq e} \mathbf{x}_v = \mathcal{F}_{u \leq e} \mathbf{x}_u\}$. This can be seen as a consequence of the *heat* equation over the sheaf via its Laplacian $\Delta_{\mathcal{F}}$. Equivalently, we can define $H^0(\mathcal{G}; \mathcal{F}) := \ker(\delta) = \ker(L_{\mathcal{F}})$, thus we can interpret this as a *convergence to zero disagreement* across nodes, yielding $\delta(\mathbf{x})_e = 0 \forall e$.

Implicit Sheaf Operators (ISO)

While sheaf-based models mitigate oversmoothing (Hansen and Gebhart 2020; Bodnar et al. 2022), their explicit implementation is costly because it requires construction of the $nd \times nd$ sheaf Laplacian $\Delta_{\mathcal{F}}$ or adjacency matrix $\hat{\mathbf{A}}_{\mathcal{F}}$. For a graph with n nodes, m edges and stalk dimension d , this requires $O(md^2)$ storage and an $O(md^2 \log(md^2))$ sparse-coalescing step, creating a major bottleneck. We address this limitation with ISO, which replace explicit assembly with local edge-wise Batched Matrix Multiplication (BMM) over the restriction maps. This yields the following gains over the explicit implementation:

1. **Sparse Index Storage:** By avoiding the construction of an $nd \times nd$ block matrix, ISOs eliminate all sparse index overhead, thus saving $O(md^2)$ space.
2. **Sparse Coalescing:** Explicit sparse-block assembly requires sorting and merging $O(md^2)$ index entries, incurring $O(md^2 \log(md^2))$ time complexity. ISOs eliminate this bottleneck by operating directly on restriction maps through gather–scatter message passing operations.
3. **Transport-Map Formation:** Explicit adjacency construction forms $d \times d$ transport maps $\mathcal{F}_{v \leq e}^\top \mathcal{F}_{u \leq e}$ at a cost of $O(md^3)$. ISA eliminates this step through a two-BMM decomposition, with a time complexity of $O(md^2 f)$.

Table 1: Per-call cost of shared primitives by sheaf family.

Primitive	$O(d)$	Diagonal	General
DEGREE	$O(m)$	$O(md)$	$O(md^3)$
DEGREEINV SORT	$O(n)$	$O(nd)$	$O(nd^3)$
Per node BMM	$O(ndf)$	$O(ndf)$	$O(nd^2 f)$
Per edge BMM	$O(md^2 f)$	$O(mdf)$	$O(md^2 f)$

Shared primitives. Both ISL and ISA rely on two shared subroutines and two shared kernels; all costs depend on the sheaf family as summarized in Table 1. **DEGREE** computes the per-node block $D_{\mathcal{F}(v)} = \sum_{e: v \leq e} \mathcal{F}_{v \leq e}^\top \mathcal{F}_{v \leq e}$. For diagonal restriction maps, this can be computed elementwise as $D_{\mathcal{F}(v)} = \sum_{e: v \leq e} \mathcal{F}_{v \leq e}^2$, whereas for bundle restriction maps, the degree block simplifies directly to $D_{\mathcal{F}(v)} = \deg(v) I_d$. **DEGREEINV SORT** calculates $D_{\mathcal{F}(v)}^{-1/2}$, and benefits from similar optimizations in the diagonal and bundle case. The *per-node BMM* applies $D_{\mathcal{F}(v)}^{-1/2}$ to each node’s feature slice. The *per-edge BMM* is the two-BMM gather–scatter:

$$\text{TwoBMM}(\mathcal{F}, \mathbf{X})_v := \sum_{u, v \leq e} \mathcal{F}_{v \leq e}^\top \mathcal{F}_{u \leq e} \mathbf{x}_u, \quad (3)$$

which avoids forming the transport maps $\mathcal{F}_{v \leq e}^\top \mathcal{F}_{u \leq e}$ explicitly, at a cost of $\mathcal{O}(md^2 f)$.

Assumption 1. All sparse tensors are stored in the Coordinate Format (COO) as in Bodnar et al. (2022).

Assumption 2. All graphs are connected, such that $n \leq 2m$.

Theorem 1. Constructing and applying the explicit normalized sheaf Laplacian $\Delta_{\mathcal{F}} \mathbf{X}$ costs:

- **General:** $\mathcal{O}(md^3 + md^2 \log(md^2) + md^2 f)$;
- **Bundle:** $\mathcal{O}(md^3 + md^2 \log(md^2) + md^2 f)$;
- **Diagonal:** $\mathcal{O}(md \log(md) + md f)$.

Proof sketch. Each cost term maps to one pipeline stage. Forming the $2m$ transport maps $\mathcal{F}_{v \leq e}^\top \mathcal{F}_{u \leq e}$ costs $\mathcal{O}(md^3)$ in the general and bundle case, $\mathcal{O}(md)$ in the diagonal. Forming the sparse tensors sorts the nonzero entries of the operator, which are $\mathcal{O}(md^2)$ in the general and bundle case, $\mathcal{O}(md)$ in the diagonal case, sorting therefore introduces the log term. Applying the operator requires Sparse Matrix Multiplication, which has cost $\mathcal{O}(md^2 f)$ in the general and bundle case, $\mathcal{O}(md)$ in the diagonal case (Full proofs for every sketch are available in the Appendix).

Theorem 2. The peak auxiliary memory of the explicit sheaf Laplacian is:

- **General / Bundle:** $\mathcal{O}(md^2 f)$;
- **Diagonal:** $\mathcal{O}(md f)$.

Proof sketch. The peak allocation is given by the autograd storage buffer of the operator applied on the 0-cochain, whose size is itself dependent on the count of nonzero elements ($\mathcal{O}(md^2)$ for $\mathcal{F}_{u \leq e}$ bundle/general, $\mathcal{O}(md)$ in the diagonal case), and the feature dimension f .

Implicit Sheaf Laplacian

Neural Sheaf Diffusion (NSD) updates node features by multiplying $\Delta_{\mathcal{F}} \in \mathbb{R}^{nd \times nd}$ with a flattened 0-cochain feature matrix $\mathbf{X} \in \mathbb{R}^{nd \times f}$, introducing a scale dependency on both the graph size n and the stalk dimension d . The key insight behind ISL is that the global operator $\Delta_{\mathcal{F}}$ acting on the entire 0-cochain space $\mathbf{X}$ can be decoupled into independent, node-centric computations. As such ISL exploits the local topology of the sheaf: the action of the Laplacian on a given node v strictly depends only on its local stalk $\mathcal{F}(v)$ and the restriction maps $\mathcal{F}_{v \leq e}$ of its incident edges. Given incident nodes $e = (u, v)$, the product between the Laplacian and the 0-cochain $\Delta_{\mathcal{F}} \mathbf{X}$ admits the nodewise decomposition:

$$(L_{\mathcal{F}} \mathbf{X})_v = D_{\mathcal{F}(v)} \mathbf{x}_v - \text{TwoBMM}(\mathcal{F}, \mathbf{X})_v. \quad (4)$$

The normalized version absorbs $D_{\mathcal{F}}^{-1/2}$ into the node features before and after the gather-scatter steps:

$$\begin{aligned} \tilde{\mathbf{X}} &= D_{\mathcal{F}}^{-1/2} \mathbf{X}, \\ (\Delta_{\mathcal{F}} \mathbf{X})_v &= D_{\mathcal{F}(v)}^{-1/2} \left(D_{\mathcal{F}(v)} \tilde{\mathbf{x}}_v - \text{TwoBMM}(\mathcal{F}, \tilde{\mathbf{X}})_v \right). \end{aligned} \quad (5)$$

This is mathematically equivalent to computing $L_{\mathcal{F}} = \delta^\top \delta$, normalizing via $\Delta_{\mathcal{F}} := D_{\mathcal{F}}^{-1/2} L_{\mathcal{F}} D_{\mathcal{F}}^{-1/2}$, and multiplying $\Delta_{\mathcal{F}} \mathbf{X}$, while avoiding all intermediate sparse structures, the procedure is detailed in Algorithm 1.

Algorithm 1: Implicit Sheaf Laplacian (ISL)

Input: Features $\mathbf{X} \in \mathbb{R}^{nd \times f}$

Parameter: Maps $\mathcal{F} \in \mathbb{R}^{2m \times d \times d}$, edge index (row, col), reverse-edge index $\mathbf{r}$

Output: out = $\Delta_{\mathcal{F}} \mathbf{X}$

```

1:  $D_v \leftarrow \text{DEGREE}(\mathcal{F}, \text{row})$ 
2: if normalized then
3:    $D_v^{-1/2} \leftarrow \text{DEGREEINV SORT}(D_v)$ 
4:    $\mathbf{X}[v] \leftarrow D_v^{-1/2} \mathbf{X}[v], \quad \forall v \quad \triangleright$  per-node BMM pre-scale
5: end if
6:  $\text{deg\_term}[v] \leftarrow D_v \mathbf{X}[v], \quad \forall v$ 
7:  $F_{\text{other}} \leftarrow \mathcal{F}[\mathbf{r}] \quad \triangleright$  destination maps  $\mathcal{F}_{u \leq e}$ 
8:  $\text{step}_1 \leftarrow F_{\text{other}} \mathbf{X}[\text{col}] \quad \triangleright$  BMM 1 per edge  $e: \mathcal{F}_{u \leq e} \mathbf{x}_u$ 
9:  $\text{step}_2 \leftarrow \mathcal{F}^\top \text{step}_1 \quad \triangleright$  BMM 2 per edge  $e: \mathcal{F}_{v \leq e}^\top(\cdot)$ 
10:  $\text{adj\_term} \leftarrow \text{ScatterAdd}(\text{step}_2, \text{row})$ 
11:  $\text{out} \leftarrow \text{deg\_term} - \text{adj\_term} \quad \triangleright$  Laplacian: degree minus adjacency
12: if normalized then
13:    $\text{out}[v] \leftarrow D_v^{-1/2} \text{out}[v], \quad \forall v \quad \triangleright$  per-node BMM post-scale
14: end if
```

Theorem 3. For any cellular sheaf $\mathcal{F}$ on $\mathcal{G}$, with uniform stalk dimension d , square $d \times d$ restriction maps and 0-cochain $\mathbf{X} \in \mathbb{R}^{nd \times f}$, Algorithm 1 computes $\Delta_{\mathcal{F}} \mathbf{X}$ exactly via Eq. 4 and 6 without materializing the $nd \times nd$ sparse matrix.

Proof sketch. The block row v of $L_{\mathcal{F}}$ contains only $D_{\mathcal{F}}(v)$ and $-\mathcal{F}_{v \leq e}^\top \mathcal{F}_{u \leq e}$ over $u, v \leq e$. Therefore $(L_{\mathcal{F}} \mathbf{X})_v = D_{\mathcal{F}}(v) \mathbf{x}_v - \text{TwoBMM}(\mathcal{F}, \mathbf{X})_v$, by expanding the block product of $(L_{\mathcal{F}} \mathbf{X})_v$ and re-parenthesizing each summand as $\mathcal{F}_{v \leq e}^\top (\mathcal{F}_{u \leq e} \mathbf{x}_u)$, exactly what is computed by lines 6 – 11 of Algorithm 1. Since $D_{\mathcal{F}}^{-1/2}$ is block-diagonal, the normalization factors are distributed into the nodewise pre-scale (line 4) and post-scale (line 13).

Theorem 4. Algorithm 1 computes $\Delta_{\mathcal{F}} \mathbf{X}$ in:

- **General:** $\mathcal{O}(m \log m + md^3 + md^2 f)$;
- **Bundle:** $\mathcal{O}(m \log m + md^2 f)$;
- **Diagonal:** $\mathcal{O}(m \log m + md f)$.

This eliminates the $\mathcal{O}(md^2 \log(md^2))$ coalescing and $\mathcal{O}(md^3)$ transport-map costs of the explicit path (Thm. 1).

Proof sketch. The reverse-edge index $\mathbf{r}$ is built through sorting ($\mathcal{O}(m \log m)$) once per topology. Computing degree normalization in lines 1 – 5 and 12 – 14 of Alg 1 incurs per node costs of $\mathcal{O}(nd^3)/\mathcal{O}(n)/\mathcal{O}(nd)$ (general / bundle / diagonal), plus two batched matrix multiplies with cost $\mathcal{O}(md^2 f)$ ($\mathcal{O}(md f)$ in the diagonal case). Due to assumption 2, we disregard the per-node costs since they are dominated by edge-dependent terms in all restriction map families.

Theorem 5. The peak auxiliary memory of Algorithm 1 is:

- **General / Bundle:** $\mathcal{O}(md^2 + md f)$;
- **Diagonal:** $\mathcal{O}(md f)$;

Proof sketch. Differently from Theorem 2, no sparse operator exists; the only indices are three arrays of length $2m$. The peak memory usage is given by the autograd storage of the TwoBMM: the gathered restriction maps $\mathcal{O}(md^2)$ and the per-edge feature intermediates $\mathcal{O}(mdf)$. We disregard per-node buffers of order $\mathcal{O}(nd^2)$ or $\mathcal{O}(ndf)$ due to assumption 2.

Sheaf Convolutional Network

As shown in the derivation of ISL, the Sheaf Laplacian can be decomposed as $L_{\mathcal{F}} = D_{\mathcal{F}} - A_{\mathcal{F}}$, by retaining only the adjacency term $A_{\mathcal{F}}$, we can intuitively define a *sheaf convolution* operator. This mirrors the transition from Laplacian diffusion to the propagation rule of GCN (Kipf and Welling 2017). Furthermore, this adjacency-based formulation was previously identified as a discretization of the sheaf heat equation by Bodnar et al. (2022), and has recently been shown by Bourgerie, Girdzijauskas, and Fodor (2026) to counteract the vanishing gradient problem, thereby enabling the construction of deeper SNNs.

Given a cellular sheaf $\mathcal{F}$ over a graph $\mathcal{G}$, define the *augmented sheaf adjacency* $\mathbf{A}_{\mathcal{F}}^{(k)} = (A_{\mathcal{F}}^{(k)} + I_{nd})$, the augmented sheaf degree $\tilde{D}_{\mathcal{F}}^{(k)}(v) = D_{\mathcal{F}}^{(k)}(v) + I_d$ and its symmetric normalization $\hat{\mathbf{A}}_{\mathcal{F}}^{(k)} = \tilde{D}_{\mathcal{F}}^{-1/2} \mathbf{A}_{\mathcal{F}}^{(k)} \tilde{D}_{\mathcal{F}}^{-1/2}$, following the renormalization of Kipf and Welling (2017). The Sheaf Convolutional Network (SCN) layer is then given by the following diffusion step:

$$\mathbf{Z}^{(k)} = (\mathbf{I}_n \otimes \mathbf{W}_1^{(k)}) \mathbf{X}^{(k)} \mathbf{W}_2^{(k)}, \quad (7)$$

$$\mathbf{X}^{(k+1)} = (1 + \varepsilon^{(k)}) \odot \mathbf{X}^{(k)} + \sigma(\hat{\mathbf{A}}_{\mathcal{F}^{(k)}} \mathbf{Z}^{(k)}). \quad (8)$$

Where $\mathbf{W}_1^{(k)}$ and $\mathbf{W}_2^{(k)}$ act on the stalk and feature dimensions respectively, and $\varepsilon^{(k)}$ is a learnable residual scale. The adjacency operator aggregates transported neighbor representations additively rather than differencing them against the node’s own transported feature, reaching up to 3.93% higher node-classification accuracy on Pei et al. (2020) benchmark compared to NSD. (See Table 3).

Theorem 6. *Constructing and applying the explicit normalized augmented sheaf adjacency $\hat{\mathbf{A}}_{\mathcal{F}} \mathbf{X}$ costs:*

- **General:** $\mathcal{O}(md^3 + md^2 \log(md^2) + md^2 f)$;
- **Bundle:** $\mathcal{O}(md^3 + md^2 \log(md^2) + md^2 f)$;
- **Diagonal:** $\mathcal{O}(md \log(md) + mdf)$.

Proof sketch. The explicit adjacency algorithm differs from Theorem 1 only in sign, and in n normalized self-loop blocks $\tilde{D}_{\mathcal{F}}(v)^{-1}$, which are absorbed, as $\mathcal{O}(nd^2) \leq \mathcal{O}(md^2)$ by assumption 2.

Theorem 7. *The peak auxiliary memory of the explicit augmented sheaf adjacency is:*

- **General / Bundle:** $\mathcal{O}(md^2 f)$;
- **Diagonal:** $\mathcal{O}(mdf)$;

Proof sketch. The proof follows the same argument as Theorem 2. In addition, n explicitly materialized self-loop blocks are present, resulting in a $\mathcal{O}(nd^2)$ term in the general and bundle case, and a $\mathcal{O}(nd)$ term in the diagonal case. Due to assumption 2 these node-wise terms are absorbed.

Algorithm 2: Implicit Sheaf Adjacency (ISA)

Input: Features $\mathbf{X} \in \mathbb{R}^{nd \times f}$

Parameter: Maps $\mathcal{F} \in \mathbb{R}^{2m \times d \times d}$, edge index (row, col), reverse-edge index $\mathbf{r}$

Output: out = $\hat{\mathbf{A}}_{\mathcal{F}} \mathbf{X}$

```

1:  $D_v \leftarrow \text{DEGREE}(\mathcal{F}, \text{row})$ 
2: if normalized then
3:    $D_v \leftarrow D_v + \mathbf{I}_d$  ▷ degree augmentation
4:    $D_v^{-1/2} \leftarrow \text{DEGREEINV\textbf{S}QRT}(D_v)$ 
5:    $\mathbf{X}[v] \leftarrow D_v^{-1/2} \mathbf{X}[v], \quad \forall v$  ▷ per-node BMM pre-scale
6: end if
7: out  $\leftarrow \mathbf{X}$  ▷ self-loop on (pre-scaled)  $\mathbf{X}$ 
8:  $F_{\text{other}} \leftarrow \mathcal{F}[\mathbf{r}]$  ▷ destination maps  $\mathcal{F}_{u \trianglelefteq e}$ 
9:  $\text{step}_1 \leftarrow F_{\text{other}} \mathbf{X}[\text{col}]$  ▷ BMM 1 per edge  $e: \mathcal{F}_{u \trianglelefteq e} \mathbf{x}_u$ 
10:  $\text{step}_2 \leftarrow \mathcal{F}^\top \text{step}_1$  ▷ BMM 2 per edge  $e: \mathcal{F}_{v \trianglelefteq e}^\top(\cdot)$ 
11: adj_term  $\leftarrow \text{ScatterAdd}(\text{step}_2, \text{row})$ 
12: out  $\leftarrow \text{out} + \text{adj\_term}$  ▷ self-loop plus neighbor aggregation
13: if normalized then
14:   out[v]  $\leftarrow D_v^{-1/2} \text{out}[v], \quad \forall v$  ▷ per-node BMM post-scale
15: end if
```

Implicit Sheaf Adjacency

Similarly to ISL, the product $\hat{\mathbf{A}}_{\mathcal{F}} \mathbf{X}$ of SCN can be reformulated as an implicit operator, avoiding the materialization of the $nd \times nd$ augmented adjacency matrix entirely. The structural difference from the Laplacian case is twofold: the adjacency term is *accumulated* rather than subtracted, and the degree term $D_{\mathcal{F}(v)} \mathbf{x}_v$ is replaced by the self-loop $\mathbf{x}_v$. Given a flattened 0-cochain feature matrix $\mathbf{X} \in \mathbb{R}^{nd \times f}$, the product with the augmented adjacency $\mathbf{A}_{\mathcal{F}} = A_{\mathcal{F}} + I_{nd}$ admits the following nodewise decomposition:

$$(\mathbf{A}_{\mathcal{F}} \mathbf{X})_v = \mathbf{x}_v + \text{TwoBMM}(\mathcal{F}, \mathbf{X})_v, \quad (9)$$

and the symmetrically normalized version via $\tilde{\mathbf{X}} = \tilde{D}_{\mathcal{F}}^{-1/2} \mathbf{X}$:

$$(\hat{\mathbf{A}}_{\mathcal{F}} \mathbf{X})_v = \tilde{D}_{\mathcal{F}(v)}^{-1/2} (\tilde{\mathbf{x}}_v + \text{TwoBMM}(\mathcal{F}, \tilde{\mathbf{X}})_v). \quad (10)$$

The complete procedure is given in Algorithm 2.

Theorem 8. *For any cellular sheaf $\mathcal{F}$ on $\mathcal{G}$, with uniform stalk dimension d , square $d \times d$ restriction maps and 0-cochain $\mathbf{X} \in \mathbb{R}^{nd \times f}$, Algorithm 2 computes $\hat{\mathbf{A}}_{\mathcal{F}} \mathbf{X}$ exactly via Eq. 9 and 10 without materializing an $nd \times nd$ matrix.*

Proof sketch. The proof follows a similar trace to Theorem 3 with some substitutions: the degree term becomes the self-loop copy out $\leftarrow \mathbf{X}$ (line 7), and the scattered aggregation is added rather than subtracted.

Theorem 9. *Algorithm 2 computes $\hat{\mathbf{A}}_{\mathcal{F}} \mathbf{X}$ in:*

- **General:** $\mathcal{O}(m \log m + md^3 + md^2 f)$;
- **Bundle:** $\mathcal{O}(m \log m + md^2 f)$;
- **Diagonal:** $\mathcal{O}(m \log m + mdf)$;

Table 2: Inductive forward-pass time (ms) and peak memory (MB) on Erdős-Rényi graphs with $n=1000$, $m=189,810$ and $f=16$. Explicit optimized vs. Implicit Sheaf Operators formulation. Mean and std over 3 runs. Best values in **bold**; $\times$ denotes speedup (time) or reduction ratio (memory).

Family	d	Forward Time (ms)			Peak Memory (MB)		
		Explicit	ISO	$\times$	Explicit	ISO	$\times$
<i>Sheaf Convolution</i>							
Diagonal	2	3.62 $\pm_{.01}$	3.26 $\pm_{.02}$	1.11 $\times$	105.8 $\pm_{.3}$	97.8 $\pm_{.1}$	1.08 $\times$
	4	7.49 $\pm_{.02}$	7.33 $\pm_{.02}$	1.02 $\times$	208.8 $\pm_{.5}$	191.3 $\pm_{.4}$	1.09 $\times$
	8	14.80 $\pm_{.04}$	13.63 $\pm_{.03}$	1.09 $\times$	418.0 $\pm_{1.2}$	382.9 $\pm_{.8}$	1.09 $\times$
Bundle	2	9.12 $\pm_{.03}$	9.14 $\pm_{.03}$	1.00 $\times$	117.1 $\pm_{.2}$	128.5 $\pm_{.7}$	0.91 $\times$
	4	24.73 $\pm_{.15}$	20.64 $\pm_{.15}$	1.20 $\times$	451.7 $\pm_{2.6}$	356.8 $\pm_{1.8}$	1.27 $\times$
	8	94.96 $\pm_{.66}$	58.04 $\pm_{.19}$	1.64 $\times$	1797.7 $\pm_{8.0}$	470.0 $\pm_{2.4}$	3.83 $\times$
General	2	7.91 $\pm_{.01}$	7.39 $\pm_{.02}$	1.07 $\times$	117.3 $\pm_{.9}$	124.3 $\pm_{.8}$	0.94 $\times$
	4	25.31 $\pm_{.10}$	18.94 $\pm_{.04}$	1.34 $\times$	454.4 $\pm_{1.2}$	348.5 $\pm_{1.2}$	1.30 $\times$
	8	119.14 $\pm_{.58}$	79.16 $\pm_{.14}$	1.51 $\times$	1816.5 $\pm_{4.6}$	463.3 $\pm_{1.0}$	3.92 $\times$
<i>Sheaf Diffusion</i>							
Diagonal	2	4.50 $\pm_{.01}$	3.27 $\pm_{.00}$	1.38 $\times$	104.5 $\pm_{.8}$	96.9 $\pm_{.4}$	1.08 $\times$
	4	8.83 $\pm_{.06}$	6.58 $\pm_{.04}$	1.34 $\times$	207.5 $\pm_{.5}$	191.2 $\pm_{.4}$	1.09 $\times$
	8	15.74 $\pm_{.01}$	12.04 $\pm_{.01}$	1.31 $\times$	415.0 $\pm_{1.1}$	383.0 $\pm_{.8}$	1.08 $\times$
Bundle	2	19.38 $\pm_{.20}$	9.16 $\pm_{.07}$	2.11 $\times$	115.5 $\pm_{.6}$	128.3 $\pm_{.3}$	0.90 $\times$
	4	34.53 $\pm_{.34}$	19.90 $\pm_{.09}$	1.74 $\times$	443.7 $\pm_{2.3}$	356.6 $\pm_{1.9}$	1.24 $\times$
	8	94.22 $\pm_{.53}$	54.57 $\pm_{.37}$	1.73 $\times$	1766.7 $\pm_{9.0}$	470.6 $\pm_{2.2}$	3.75 $\times$
General	2	37.78 $\pm_{.21}$	7.76 $\pm_{.08}$	4.87 $\times$	115.9 $\pm_{.5}$	124.5 $\pm_{.3}$	0.93 $\times$
	4	52.75 $\pm_{.28}$	17.96 $\pm_{.11}$	2.94 $\times$	448.7 $\pm_{1.0}$	348.2 $\pm_{1.2}$	1.29 $\times$
	8	105.07 $\pm_{.76}$	61.56 $\pm_{.49}$	1.71 $\times$	1793.3 $\pm_{4.1}$	463.5 $\pm_{1.3}$	3.87 $\times$
GCN	—	2.68 $\pm_{.02}$	—	—	193.4 $\pm_{.01}$	—	—

matching ISL (Thm. 4) in all families.

Proof sketch. The proof is almost equivalent to Theorem 4; the only difference being the degree term being replaced by an $\mathcal{O}(ndf)$ self loop copy, which is dominated by the two BMM term since $\mathcal{O}(ndf) \leq \mathcal{O}(mdf) \leq \mathcal{O}(md^2f)$ by assumption 2.

Theorem 10. The peak auxiliary memory of Algorithm 2 is:

- **General / Bundle:** $\mathcal{O}(md^2 + mdf)$;
- **Diagonal:** $\mathcal{O}(mdf)$;

Proof sketch. Algorithm 2 allocates a subset of Algorithm 1’s tensors, so the tally of Theorem 5 applies unchanged; both the edge-block storage and the self-loop blocks of the explicit path (see Theorem 7) disappear.

Comparison. *First*, the original Bodnar et al. (2022) implementation performs the reverse-edge lookup sequentially on CPU via a hashmap, incurring severe CPU-GPU synchronization overhead. Our explicit and implicit baselines both replace this with a GPU-vectorized alternative, computed once per graph topology, which is orders of magnitude faster despite the higher $\mathcal{O}(m \log m)$ asymptotic cost. *Second*, the implicit operators eliminate the sparse matrix assembly and the explicit formation of the transport maps entirely. For orthogonal and general sheaves, avoiding transport map formation removes one of the two $\mathcal{O}(md^3)$ computations in the explicit path. *Third*, while in the transductive setting the coalescing step $\mathcal{O}(md^2 \log md^2)$, and index storage $\mathcal{O}(md^2)$ are paid once and amortized, in the inductive setting (where

the operator must be rebuilt for every new graph) these assembly costs dominate. *Fourth*, the implicit operator application incurs one extra BMM compared to the explicit path, but retains substantially less autograd memory since it never materializes the sparse operator, making it preferable even when inductive speedup is not the primary concern (Table 11). Comparing ISL and ISA, both share the same apply cost and degree computation, yielding an advantage in memory of $\mathcal{O}(\min(d, f))$ over their explicit counterparts (Table 34).

A Small Example: Multi-Agent Coordination

We show an example of how ISL performs sheaf diffusion on a graph without materializing any block-sparse Laplacian, while explicit NSD requires $2\times$ more $d \times d$ matrix products than ISL. Let us consider the setup shown in Figure 4, where the three nodes are on a path $v_1 \xrightarrow{e_1} v_2 \xleftarrow{e_2} v_3$; Each node holds an internal measurement in their 2-dimensional stalks $\mathbf{x}_i$, and seeks to reach an agreement with their neighbors through sheaf diffusion. Thus, the application of the Laplacian operator on the 0-cochain $L_{\mathcal{F}}\mathbf{X}$ represents the correction each node must apply to their features so as the network reaches consensus (Hanks et al. 2025). By construction, v_1 and v_2 can agree across their edge e_1 , indeed we can compute $\delta(\mathbf{x})_{e_1} = \mathcal{F}_{v_1 \leq e_1} \mathbf{x}_1 - \mathcal{F}_{v_2 \leq e_1} \mathbf{x}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix} = 0$. Thus, we focus our remaining computations on node v_2 , where there is quantifiable disagreement across e_2 given the current stalk assignments.

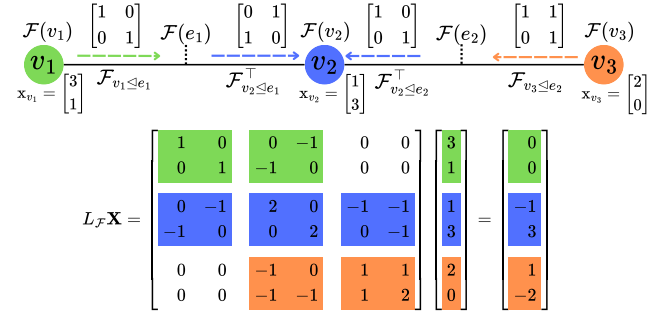


Figure 4: **(Bottom)** Explicit Laplacian diffusion requires materializing the full $nd \times nd$ block-sparse operator; **(Top)** ISO leverages restriction maps to forward individual node features without requiring sparse matrix multiplications.

Explicit path (NSD). Obtaining $(L_{\mathcal{F}}\mathbf{X})_{v_2}$ explicitly requires assembling $L_{\mathcal{F}} = \delta^{\top} \delta$, shown in full in Figure 4. Focusing on the diffusion update received by v_2 , NSD computes the block row by forming the transport maps $\mathcal{F}_{v_2 \leq e_1}^{\top} \mathcal{F}_{v_1 \leq e_1} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ and $\mathcal{F}_{v_2 \leq e_2}^{\top} \mathcal{F}_{v_3 \leq e_2} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$, together with the degree block $D_{\mathcal{F}(v_2)} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$. These dense $d \times d$ blocks ($3d^2$ scalars for a single row) must be formed, sorted and coalesced into the explicit Laplacian operator before any update is computed. Finally, we obtain $(L_{\mathcal{F}}\mathbf{X})_{v_2} = \begin{bmatrix} -1 & 3 \end{bmatrix}^{\top}$ the sheaf diffusion update that brings v_2 in agreement with v_3 over e_2 .

Implicit path (ISL). By leveraging ISL, we bypass the $\mathcal{O}(md^3)$ transport map formation and sparse matrix instantiation costs entirely. Instead, we evaluate $(L_{\mathcal{F}}\mathbf{X})_{v_2} = D_{\mathcal{F}(v_2)} \mathbf{x}_2 - \text{TwoBMM}(\mathcal{F}, \mathbf{X})_{v_2}$ directly through two local operator-vector passes: (1) Neighbor stalks are mapped to their

Table 3: *Node-classification benchmark results (Pei et al. 2020)*. Unless otherwise marked, results are mean $\pm$ std over 10 fixed 48%/32%/20% train/validation/test splits. The top three distinct, directly comparable results for each dataset are colored by **First**, **Second** and **Third**, respectively. Full results are available in Appendix Table 5.

	Texas	Wisconsin	Film	Squirrel	Chameleon	Cornell	Citeseer	Pubmed	Cora
<i>Sheaf Convolution</i>									
Diag-Conv	87.02$\pm$4.95	88.03 $\pm$ 4.06	35.90 $\pm$ 0.52	56.50$\pm$1.47	67.50 $\pm$ 2.04	86.48$\pm$5.53	77.14$\pm$1.58	89.64$\pm$0.59	87.64$\pm$1.49
Bundle-Conv	86.21$\pm$6.21	88.03 $\pm$ 5.29	35.27 $\pm$ 1.11	59.50$\pm$1.61	68.22$\pm$1.42	86.75$\pm$5.46	77.04$\pm$1.66	89.54$\pm$0.41	87.82$\pm$1.42
Gen-Conv	87.83$\pm$5.82	89.01$\pm$4.13	34.97 $\pm$ 0.70	60.27$\pm$2.52	70.42$\pm$1.93	86.48$\pm$5.53	77.11$\pm$1.38	89.42 $\pm$ 0.37	87.50$\pm$1.05
<i>Sheaf Diffusion</i>									
Diag-NSD	85.67 $\pm$ 6.95	88.63 $\pm$ 2.75	37.79 $\pm$ 1.01	54.78 $\pm$ 1.81	68.68$\pm$1.73	86.49$\pm$7.35	77.14$\pm$1.85	89.42 $\pm$ 0.43	87.14 $\pm$ 1.06
Bundle-NSD	85.95 $\pm$ 5.51	89.41$\pm$4.74	37.81$\pm$1.15	56.34 $\pm$ 1.32	68.04 $\pm$ 1.58	84.86 $\pm$ 4.71	76.70 $\pm$ 1.57	89.49$\pm$0.40	86.90 $\pm$ 1.13
Gen-NSD	82.97 $\pm$ 5.13	89.21$\pm$3.84	37.80$\pm$1.22	53.17 $\pm$ 1.31	67.93 $\pm$ 1.58	85.68 $\pm$ 6.51	76.32 $\pm$ 1.65	89.33 $\pm$ 0.35	87.30 $\pm$ 1.15
Conn-NSD	86.16 $\pm$ 2.24	88.73 $\pm$ 4.47	37.91$\pm$1.28	45.19 $\pm$ 1.57	65.21 $\pm$ 2.04	85.95 $\pm$ 7.72	75.61 $\pm$ 1.93	89.28 $\pm$ 0.38	83.74 $\pm$ 2.19

Table 4: Jetson Orin Nano inference at $d=8$, $f=16$. Mean and std over 3 runs. Optimized Explicit vs. ISO; $\times$ is improvement ratio (higher=better). Best in **bold**.

Family	Energy/Graph (mJ)			Memory (MB)		
	Explicit	ISO	$\times$	Explicit	ISO	$\times$
PTC (344 graphs, avg 39.1 nodes)						
<i>Sheaf Convolution</i>						
Diagonal	2.56 $\pm_{.02}$	2.38$\pm_{.02}$	1.07	23 $\pm_1$	24 $\pm_1$	0.97
Bundle	4.59 $\pm_{.05}$	4.15$\pm_{.03}$	1.11	54 $\pm_2$	27$\pm_1$	2.03
General	10.83 $\pm_{.16}$	10.18$\pm_{.18}$	1.06	52 $\pm_1$	26$\pm_1$	2.02
<i>Sheaf Diffusion</i>						
Diagonal	3.28 $\pm_{.06}$	2.63$\pm_{.03}$	1.24	23 $\pm_1$	23$\pm_1$	1.00
Bundle	5.21 $\pm_{.03}$	4.23$\pm_{.05}$	1.23	42 $\pm_1$	26$\pm_1$	1.62
General	11.59 $\pm_{.12}$	9.80$\pm_{.24}$	1.18	50 $\pm_2$	26$\pm_1$	1.95
PROTEINS (1113 graphs, avg 25.5 nodes)						
<i>Sheaf Convolution</i>						
Diagonal	2.99 $\pm_{.09}$	2.81$\pm_{.07}$	1.06	39 $\pm_2$	37$\pm_3$	1.07
Bundle	7.52 $\pm_{.14}$	7.00$\pm_{.17}$	1.07	132 $\pm_5$	46$\pm_2$	2.87
General	19.28 $\pm_{.07}$	18.16$\pm_{.41}$	1.06	137 $\pm_{13}$	48$\pm_4$	2.84
<i>Sheaf Diffusion</i>						
Diagonal	3.75 $\pm_{.12}$	3.06$\pm_{.13}$	1.23	39 $\pm_1$	36$\pm_1$	1.07
Bundle	8.26 $\pm_{.13}$	6.92$\pm_{.16}$	1.19	105 $\pm_{11}$	44$\pm_2$	2.37
General	19.17 $\pm_{.35}$	16.75$\pm_{.57}$	1.14	134 $\pm_9$	44$\pm_1$	3.08

edges $\mathbf{z}_{e_1} = \mathcal{F}_{v_1 \leq e_1} \mathbf{x}_1 = \begin{bmatrix} 3 & 1 \end{bmatrix}^\top$, and $\mathbf{z}_{e_2} = \mathcal{F}_{v_3 \leq e_2} \mathbf{x}_3 = \begin{bmatrix} 2 & 0 \end{bmatrix}^\top$, (2) edge features are transported over to v_2 via its local restriction maps and aggregated $\mathcal{F}_{v_2 \leq e_1}^\top \mathbf{z}_{e_1} + \mathcal{F}_{v_2 \leq e_2}^\top \mathbf{z}_{e_2} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix} + \begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 & 3 \end{bmatrix}^\top$. Finally, by subtracting this aggregated message from $D_{\mathcal{F}(v_2)}(\mathbf{x}_2) = \begin{bmatrix} 2 & 6 \end{bmatrix}^\top$ we obtain $(L_{\mathcal{F}} \mathbf{X})_{v_2} = \begin{bmatrix} -1 & 3 \end{bmatrix}^\top$. By leveraging this gather-scatter formulation, ISL computes the same exact sheaf diffusion update without forming the explicit Laplacian operator.

Experiments

Efficiency Gains. We compared explicit and implicit operators on runtime and memory benchmarks across a wide variety of models and configurations. To evaluate perfor-

mance scaling, we measured across Erdős-Rényi random graphs with fixed sizes $n = 1000$ and edge probabilities $p \in \{0.01, 0.05, 0.1\}$. For each configuration, we varied the stalk dimension $d \in \{2, 4, 8\}$. The recorded metrics are averaged over 10 runs (with 5 warmup runs) across 3 fixed seeds. The evaluation covers distinct modes of execution:

- **Apply:** Assesses the forward-pass and backward-pass of repeatedly assembling the diffusion operator and multiplying it by a feature matrix (Appendix Tables 6, 10 and 11).
- **Inductive:** Evaluates the end-to-end overhead by dynamically updating the underlying graph topology, which forces the models to rebuild or update their internal indices and normalization buffers (Appendix Tables 12 to 24).

In Table 2 we report the experiment on both ISO and the explicit implementation, the results show how ISOs scale better memory-wise when the stalk dimension d is increased.

Energy Savings. We further validate the results on real graph classification datasets from Xu et al. (2019) on a NVIDIA Jetson Orin Nano that we report in Table 4 showing that ISOs save both energy and memory in real dataset scenarios.

Performance Validation. To validate SCN we evaluate them on the Pei et al. (2020) node classification benchmark, as shown in Table 3, SCN variants achieve superior predictive performance relative to the NSD models, up to 3.93% in SQUIRREL, hyperparameter and training details are available in Appendix, full comparison is available in Appendix Table 5.

Conclusions

We addressed the scalability issues of Sheaf Neural Network (SNN) by introducing Implicit Sheaf Operator (ISO), a family of implicit operators that compute operator-vector products directly from restriction maps. We proved that ISOs eliminate $\mathcal{O}(md^2 \log(md^2))$ in time and $\mathcal{O}(\min(d, f))$ in space relative to the explicit pipeline. Experimentally, ISOs achieve up to $4.87\times$ faster latency and $3.92\times$ lower peak memory over optimized explicit baselines, while on a NVIDIA Jetson Orin Nano they reduce per-graph energy by up to $1.24\times$ and on-device memory by up to $3.08\times$. Building on ISA, we introduced Sheaf Convolutional Networks (SCNs), which achieve up to 3.93% higher node-classification accuracy over NSD while retaining the same computational cost.

References

- Barbero, F.; Bodnar, C.; Borde, H. S. d. O.; Bronstein, M.; Veličković, P.; and Liò, P. 2022a. Sheaf Neural Networks with Connection Laplacians. In *Proceedings of Topological, Algebraic, and Geometric Learning Workshops 2022*, 28–36. PMLR.
- Barbero, F.; Bodnar, C.; de Ocariz Borde, H. S.; and Lio, P. 2022b. Sheaf Attention Networks. In *NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations*.
- Bodnar, C.; Di Giovanni, F.; Chamberlain, B. P.; Liò, P.; and Bronstein, M. M. 2022. Neural Sheaf Diffusion: A Topological Perspective on Heterophily and Oversmoothing in GNNs. In *Advances in Neural Information Processing Systems*.
- Bourgerie, R.; Girdzijauskas, Š.; and Fodor, V. 2026. Deep Neural Sheaf Diffusion. In *ICML 2026 Workshop on Graph Foundation Models (GFM@ICML 2026)*.
- Braithwaite, L.; Borgi, A.; Onorato, G.; Tarantelli, K.; Restuccia, F.; Silvestri, F.; and Liò, P. 2026. Heterogeneous Sheaf Neural Networks. arXiv:2409.08036.
- Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2014. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR)*.
- Caralt, F. H.; Bernardez, G.; Duta, I.; Alarcon, E.; and Lio, P. 2024. Joint Diffusion Processes as an Inductive Bias in Sheaf Neural Networks. In *ICML 2024 Workshop on Geometry-grounded Representation Learning and Generative Modeling*.
- Chamberlain, B. P.; Rowbottom, J.; Gorinova, M. I.; Webb, S. D.; Rossi, E.; and Bronstein, M. M. 2021. GRAND: Graph Neural Diffusion. In *NeurIPS Workshop on The Symbiosis of Deep Learning and Differential Equations*.
- Chen, R. T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. 2018. Neural ordinary differential equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS’18*, 6572–6583. Red Hook, NY, USA: Curran Associates Inc.
- Chung, F. R. 1997. *Spectral graph theory*, volume 92. American Mathematical Society.
- Curry, J. M. 2014. *Sheaves, Cosheaves and Applications*. Ph.D. thesis, University of Pennsylvania.
- Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS’16*, 3844–3852. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781510838819.
- Duvenaud, D.; Maclaurin, D.; Aguilera-Iparraguirre, J.; Gómez-Bombarelli, R.; Hirzel, T.; Aspuru-Guzik, A.; and Adams, R. P. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 2, NIPS’15*, 2224–2232. Cambridge, MA, USA: MIT Press.
- Ferriol-Galmés, M.; Paillisse, J.; Suárez-Varela, J.; Rusek, K.; Xiao, S.; Shi, X.; Cheng, X.; Barlet-Ros, P.; and Cabellos-Aparicio, A. 2023. RouteNet-Fermi: Network Modeling With Graph Neural Networks. *IEEE/ACM Trans. Netw.*, 31(6): 3080–3095.
- Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for Quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML’17*, 1263–1272. JMLR.org.
- Gori, M.; Monfardini, G.; and Scarselli, F. 2005. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, 729–734 vol. 2.
- Hajij, M.; Bastian, L.; Osentoski, S.; Kabaria, H.; Davenport, J. L.; Dawood, S.; Cherukuri, B.; Kocheemoolayil, J. G.; Shahmansouri, N.; Lew, A.; Papamarkou, T.; and Birdal, T. 2025. Copresheaf Topological Neural Networks: A Generalized Deep Learning Framework. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS’17*, 1025–1035. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781510860964.
- Hanks, T.; Riess, H.; Cohen, S.; Gross, T.; Hale, M.; and Fairbanks, J. 2025. Distributed Multi-Agent Coordination over Cellular Sheaves. In *2025 IEEE 64th Conference on Decision and Control (CDC)*, 3057–3064.
- Hansen, J.; and Gebhart, T. 2020. Sheaf Neural Networks. In *NeurIPS Workshop on TDA & Beyond*.
- Hansen, J.; and Ghrist, R. 2019a. Learning Sheaf Laplacians from Smooth Signals. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5446–5450.
- Hansen, J.; and Ghrist, R. 2019b. Toward a Spectral Theory of Cellular Sheaves. *Journal of Applied and Computational Topology*, 3(4): 315–358.
- Hansen, J.; and Ghrist, R. 2021. Opinion Dynamics on Discourse Sheaves. *SIAM Journal on Applied Mathematics*, 81(5): 2033–2060.
- Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.
- Kondor, R. I.; and Lafferty, J. D. 2002. Diffusion Kernels on Graphs and Other Discrete Input Spaces. In *Proceedings of the 19th International Conference on Machine Learning (ICML)*, ICML ’02, 315–322. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN 1558608737.
- Lyzinski, V.; Fishkind, D. E.; and Priebe, C. E. 2014. Seeded Graph Matching for Correlated Erdos-Renyi Graphs. *Journal of Machine Learning Research*, 15(108): 3693–3720.
- Mathiasen, A.; Hvilshøj, F.; Jørgensen, J. R.; Nasery, A.; and Mottin, D. 2020a. {Faster Orthogonal Parameterization with Householder Matrices. In *ICML Workshop on Invertible Neural Networks and Normalizing Flows*.

Mathiasen, A.; Hvilshøj, F.; Jørgensen, J. R.; Nasery, A.; and Mottin, D. 2020b. What if neural networks had SVDs? In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20*. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781713829546.

Pei, H.; Wei, B.; Chang, K. C.-C.; Lei, Y.; and Yang, B. 2020. Geom-GCN: Geometric Graph Convolutional Networks. In *International Conference on Learning Representations*.

Rosiak, D. 2022. *Sheaf Theory through Examples*. The MIT Press. ISBN 9780262370424.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1): 61–80.

Seely, J.; Cupiał, B.; and Jones, L. 2026. Learning Multi-Agent Coordination via Sheaf-ADMM. In *Forty-third International Conference on Machine Learning*.

Shepard, A. D. 1985. *A Cellular Description of the Derived Category of a Stratified Space*. Ph.D. thesis, Brown University.

Suk, J.; Giusti, L.; Hemo, T.; Lopez, M.; Barmpas, K.; and Bodnar, C. 2022. Surfing on the Neural Sheaf. In *NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations*.

Thorpe, M.; Nguyen, T. M.; Xia, H.; Strohmer, T.; Bertozzi, A.; Osher, S.; and Wang, B. 2022. GRAND++: Graph Neural Diffusion with A Source Term. In *International Conference on Learning Representations*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is All you Need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1): 4–24.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*.

Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W. L.; and Leskovec, J. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '18*, 974–983. New York, NY, USA: Association for Computing Machinery. ISBN 9781450355520.

Zaghen, O.; Longa, A.; Azzolin, S.; Telyatnikov, L.; Passerini, A.; and Lio, P. 2024. Sheaf Diffusion Goes Nonlinear: Enhancing GNNs with Adaptive Sheaf Laplacians. In *ICML 2024 Workshop on Geometry-grounded Representation Learning and Generative Modeling*.

Implementation Details

Here we describe the final components required to build a SNN for node and graph classification, such as the orthogonal parametrization used for the orthogonal sheaf Laplacian and adjacency operators, the final node embeddings, the loss computation, and the normalization.

Downstream Decoder Architecture & Loss Functions

After the SNN processes the graph, the resulting embeddings are used for downstream prediction. For **node classification**, the final node embeddings are directly fed into an MLP to generate the classification logits. For **graph classification**, we first apply a global readout function (e.g., mean or max pooling) over all node embeddings in the graph to obtain a single graph-level representation, which is then passed to an MLP. During training, we use a softmax activation combined with cross-entropy loss.

Full Node Classification Results

In Table 5, we report the evaluation of our models on the node classification benchmarks introduced by Pei et al. (2020).

Restriction Maps

Here we detail the implementation of the restriction maps depending on the chosen sheaf family for ISL and ISA.

Diagonal sheaves. Maps $\mathcal{F}_{v \leq e} = \text{diag}(\mathbf{f})$, $\mathbf{f} \in \mathbb{R}^d$. All matrix multiplies become Hadamard products; $D_v = \sum_e \mathbf{f}_{v \leq e}^{\odot 2}$ is a vector:

$$(L_{\mathcal{F}}\mathbf{X})_v = D_v \odot \mathbf{x}_v - \sum_{(v,u)} (\mathbf{f}_{v \leq e} \odot \mathbf{f}_{u \leq e}) \odot \mathbf{x}_u. \quad (11)$$

Bundle sheaves (Orthogonal Parameterizations). For bundle sheaves, the restriction maps are constrained to be orthogonal ($\mathcal{F}_{v \leq e} \in O(d)$). We can consider two parametrizations:

- **Householder.** We learn $d(d-1)/2$ parameters, which are converted by the Torch Householder library into a product of Householder reflections covering the full orthogonal group $O(d)$ (Mathiasen et al. 2020a,b).
- **Cayley.** We learn $d(d-1)/2$ parameters that are mapped to a skew-symmetric matrix $A = -A^\top$, then $Q = (\mathbf{I} + A/2)^{-1}(\mathbf{I} - A/2) \in SO(d)$ is computed via batched linear solvers with $\mathcal{O}(d^3)$ complexity.

To ensure numerical stability and prevent precision loss (especially during linear system solving and matrix exponentiation), all parameterizations internally execute the transformations in 32-bit floating-point precision before casting back to the model’s native precision.

General sheaves. Dense $d \times d$ maps. Here, all edge-wise transformations utilize batched matrix multiplications.

Normalization

The symmetrically normalized Sheaf Laplacian is defined as $\Delta_{\mathcal{F}} = D^{-1/2} L_{\mathcal{F}} D^{-1/2}$. In the Degree-Adjacency decomposition, the pre-scaling by $D_{\mathcal{F}}^{-1/2}$ is absorbed into the node

features prior to the gather-scatter steps, while the post-scaling is applied immediately after the final scatter-sum aggregation. An augmented degree $D_{vv} \leftarrow D_{vv} + \mathbf{I}_d$ is employed by default to bound the spectrum of the normalized operator. The normalization cost is based on the *restriction map* family:

- **Diagonal Sheaves:** D_{vv} is diagonal, thus $D_{vv}^{-1/2}$ is computed element-wise in $\mathcal{O}(nd)$.
- **General Sheaves:** D_{vv} is a dense symmetric positive-definite matrix. Its inverse square root is computed via batched eigendecomposition with eigenvalue clamping ($\lambda \geq 10^{-8}$) and an additive $10^{-6}\mathbf{I}_d$ regularizer, requiring $\mathcal{O}(nd^3)$ operations.
- **Bundle Sheaves:** Since restriction maps are orthogonal ($\mathcal{F}_{v \leq e}^\top \mathcal{F}_{v \leq e} = \mathbf{I}_d$), the block simplifies to $D_{vv} = \deg(v)\mathbf{I}_d$. Thus, $D_{vv}^{-1/2}$ is just a per-node scalar, with cost $\mathcal{O}(n)$.

The restriction maps (denoted as $\mathbf{f}$, $\mathcal{F}$, or O) always provide the source endpoint map $\mathcal{F}_{v \leq e}$ for each directed edge.

Memory & Time Benchmarks

This section presents runtime and memory benchmarks comparing the explicit and implicit operator across a wide variety of models and configurations. To evaluate performance scaling, we measured across Erdős-Rényi random graphs with fixed sizes $n = 1000$ and edge probabilities ($p \in \{0.01, 0.05, 0.1\}$). These probabilities correspond to a nominal expected number of directed edges $\mathbb{E}[m] = n(n-1)[1 - (1-p)^2]$, yielding $m = 19880$, $m = 97403$ and $m = 189810$ respectively. For each configuration, we varied the stalk dimension $d \in \{2, 4, 8\}$ under a symmetrically normalized setting. The recorded metrics are averaged over 10 runs (with 5 warmup runs) across 3 fixed seeds to ensure stable measurements. The evaluation covers distinct modes of execution:

- **Apply:** Assesses the forward and backward passes of repeatedly assembling the operator from restriction maps and multiplying it by a feature matrix (Tables 6, 10 and 11).
- **Inductive:** Evaluates the end-to-end overhead of dynamically updating the underlying graph topology, which forces the models to rebuild or update their internal indices and normalization buffers (Tables 12 to 24).

Edge Computing Benchmarks on Jetson Orin

To evaluate the deployment viability of our Implicit formulation, we benchmarked the models on an NVIDIA Jetson Orin Nano edge board.

Setup & Datasets To ensure a stable and fair baseline, we locked the Jetson Orin to its maximum performance profile (MAXN mode with fixed clocks) to prevent thermal throttling and dynamic frequency scaling (DVFS) interventions. We evaluated the full Neural Sheaf Diffusion and Convolution architectures (Diagonal, Bundle and General) on three real-world graph classification datasets (MUTAG, PTC, PROTEINS) with stalk dimensions $d \in \{2, 4, 8\}$. We tested both *Inference* (Tables 27, 28 and 29) and *Training* workloads (Tables 30, 31 and 32).

Table 5: *Node-classification benchmark results (Pei et al. 2020)*. Unless otherwise marked, results are mean $\pm$ std over 10 fixed 48%/32%/20% train/validation/test splits. The top three distinct, directly comparable results for each dataset are colored by **First**, **Second** and **Third**, respectively. The best between sheaf *convolution* and *diffusion* is in **black**.

	Texas	Wisconsin	Film	Squirrel	Chameleon	Cornell	Citeseer	Pubmed	Cora
<i>Homophily</i>	0.11	0.21	0.22	0.22	0.23	0.30	0.74	0.80	0.81
#Nodes	183	251	7,600	5,201	2,277	183	3,327	18,717	2,708
#Edges	295	466	26,752	198,493	31,421	280	4,676	44,327	5,278
#Classes	5	5	5	5	5	5	7	3	6
<i>Sheaf Convolution</i>									
Diag-Conv	87.02$\pm$4.95	88.03 $\pm$ 4.06	35.90 $\pm$ 0.52	56.50$\pm$1.47	67.50 $\pm$ 2.04	86.48 $\pm$ 5.53	77.14$\pm$1.58	89.64$\pm$0.59	87.64$\pm$1.49
Bundle-Conv	86.21$\pm$6.21	88.03 $\pm$ 5.29	35.27 $\pm$ 1.11	59.50$\pm$1.61	68.22$\pm$1.42	86.75$\pm$5.46	77.04$\pm$1.66	89.54$\pm$0.41	87.82$\pm$1.42
Gen-Conv	87.83$\pm$5.82	89.01 $\pm$ 4.13	34.97 $\pm$ 0.70	60.27$\pm$2.52	70.42$\pm$1.93	86.48$\pm$5.53	77.11$\pm$1.38	89.42$\pm$0.37	87.50$\pm$1.05
<i>Sheaf Diffusion</i>									
Diag-NSD	85.67 $\pm$ 6.95	88.63$\pm$2.75	37.79$\pm$1.01	54.78 $\pm$ 1.81	68.68$\pm$1.73	86.49$\pm$7.35	77.14$\pm$1.85	89.42 $\pm$ 0.43	87.14 $\pm$ 1.06
Bundle-NSD	85.95 $\pm$ 5.51	89.41$\pm$4.74	37.81$\pm$1.15	56.34 $\pm$ 1.32	68.04 $\pm$ 1.58	84.86 $\pm$ 4.71	76.70 $\pm$ 1.57	89.49 $\pm$ 0.40	86.90 $\pm$ 1.13
Gen-NSD	82.97 $\pm$ 5.13	89.21$\pm$3.84	37.80$\pm$1.22	53.17 $\pm$ 1.31	67.93 $\pm$ 1.58	85.68 $\pm$ 6.51	76.32 $\pm$ 1.65	89.33 $\pm$ 0.35	87.30 $\pm$ 1.15
Conn-NSD	86.16 $\pm$ 2.24	88.73 $\pm$ 4.47	37.91$\pm$1.28	45.19 $\pm$ 1.57	65.21 $\pm$ 2.04	85.95 $\pm$ 7.72	75.61 $\pm$ 1.93	89.28 $\pm$ 0.38	83.74 $\pm$ 2.19
<i>Nonlinear Sheaf Diffusion</i>									
Diag-NLSD	86.22 $\pm$ 3.91	89.02 $\pm$ 3.19	37.45 $\pm$ 0.94	47.63 $\pm$ 1.51	62.57 $\pm$ 2.31	86.76 $\pm$ 4.60	75.76 $\pm$ 1.62	89.52 $\pm$ 0.32	86.38 $\pm$ 1.20
Bundle-NLSD	86.22 $\pm$ 4.90	89.02 $\pm$ 3.84	37.22 $\pm$ 1.15	51.96 $\pm$ 2.65	65.37 $\pm$ 2.73	87.03$\pm$4.49	76.11 $\pm$ 1.81	89.60 $\pm$ 0.29	86.20 $\pm$ 1.24
<i>Sheaf Propagation</i>									
Diag-NSP	85.68 $\pm$ 5.93	89.02 $\pm$ 3.84	37.12 $\pm$ 1.31	48.78 $\pm$ 2.45	61.80 $\pm$ 2.31	76.22 $\pm$ 4.49	76.82 $\pm$ 1.78	89.38 $\pm$ 0.56	87.02 $\pm$ 1.91
Bundle-NSP	87.03 $\pm$ 5.51	87.06 $\pm$ 4.13	36.56 $\pm$ 1.15	49.54 $\pm$ 1.87	61.01 $\pm$ 4.14	76.22 $\pm$ 4.95	76.77 $\pm$ 1.61	89.23 $\pm$ 0.51	86.22 $\pm$ 1.55
Gen-NSP	84.60 $\pm$ 3.43	87.45 $\pm$ 4.40	37.07 $\pm$ 1.20	50.11 $\pm$ 2.03	62.85 $\pm$ 1.98	76.49 $\pm$ 5.28	76.85 $\pm$ 1.48	89.42 $\pm$ 0.33	87.38 $\pm$ 1.14
<i>Directional Sheaf</i>									
Diag-DSNN	88.65$\pm$4.95	90.20$\pm$4.02	38.34$\pm$1.01	45.37 $\pm$ 2.21	46.84 $\pm$ 4.03	87.84$\pm$5.70	79.80$\pm$1.49	90.23$\pm$0.44	87.36 $\pm$ 1.41
Bundle-DSNN	87.57$\pm$4.04	89.80$\pm$3.82	37.37 $\pm$ 0.98	44.54 $\pm$ 2.26	45.36 $\pm$ 3.29	87.30$\pm$7.26	77.28 $\pm$ 1.63	90.05 $\pm$ 0.55	87.30 $\pm$ 1.62
Gen-DSNN	87.57$\pm$5.43	89.22 $\pm$ 3.31	38.40$\pm$0.75	45.34 $\pm$ 1.69	47.16 $\pm$ 3.54	87.84$\pm$6.86	79.88$\pm$1.21	90.17$\pm$0.44	87.58 $\pm$ 0.72
<i>GNNs & MLP</i>									
GGCN	84.86 $\pm$ 4.55	86.86 $\pm$ 3.29	37.54 $\pm$ 1.56	55.17 $\pm$ 1.58	71.14$\pm$1.84	85.68 $\pm$ 6.63	77.14 $\pm$ 1.45	89.15 $\pm$ 0.37	87.95$\pm$1.05
H2GCN	84.86 $\pm$ 7.23	87.65 $\pm$ 4.98	35.70 $\pm$ 1.00	36.48 $\pm$ 1.86	60.11 $\pm$ 2.15	82.70 $\pm$ 5.28	77.11 $\pm$ 1.57	89.49 $\pm$ 0.38	87.87$\pm$1.20
GPRGNN	78.38 $\pm$ 4.36	82.94 $\pm$ 4.21	34.63 $\pm$ 1.22	31.61 $\pm$ 1.24	46.58 $\pm$ 1.71	80.27 $\pm$ 8.11	77.13 $\pm$ 1.67	87.54 $\pm$ 0.38	87.95$\pm$1.18
FAGCN	82.43 $\pm$ 6.89	82.94 $\pm$ 7.95	34.87 $\pm$ 1.25	42.59 $\pm$ 0.79	55.22 $\pm$ 3.19	79.19 $\pm$ 9.79	—	—	—
MixHop	77.84 $\pm$ 7.73	75.88 $\pm$ 4.90	32.22 $\pm$ 2.34	43.80 $\pm$ 1.48	60.50 $\pm$ 2.53	73.51 $\pm$ 6.34	76.26 $\pm$ 1.33	85.31 $\pm$ 0.61	87.61 $\pm$ 0.85
GCNII	77.57 $\pm$ 3.83	80.39 $\pm$ 3.40	37.44 $\pm$ 1.30	38.47 $\pm$ 1.58	63.86 $\pm$ 3.04	77.86 $\pm$ 3.79	77.33 $\pm$ 1.48	90.15$\pm$0.43	88.37$\pm$1.25
Geom-GCN	66.76 $\pm$ 2.72	64.51 $\pm$ 3.66	31.59 $\pm$ 1.15	38.15 $\pm$ 0.92	60.00 $\pm$ 2.81	60.54 $\pm$ 3.67	78.02$\pm$1.15	89.95 $\pm$ 0.47	85.35 $\pm$ 1.57
PairNorm	60.27 $\pm$ 4.34	48.43 $\pm$ 6.14	27.40 $\pm$ 1.24	50.44 $\pm$ 2.04	62.74 $\pm$ 2.82	58.92 $\pm$ 3.15	73.59 $\pm$ 1.47	87.53 $\pm$ 0.44	85.79 $\pm$ 1.01
GraphSAGE	82.43 $\pm$ 6.14	81.18 $\pm$ 5.56	34.23 $\pm$ 0.99	41.61 $\pm$ 0.74	58.73 $\pm$ 1.68	75.95 $\pm$ 5.01	76.04 $\pm$ 1.30	88.45 $\pm$ 0.50	86.90 $\pm$ 1.04
GCN	55.14 $\pm$ 5.16	51.76 $\pm$ 3.06	27.32 $\pm$ 1.10	53.43 $\pm$ 2.01	64.82 $\pm$ 2.24	60.54 $\pm$ 5.30	76.50 $\pm$ 1.36	88.42 $\pm$ 0.50	86.98 $\pm$ 1.27
GAT	52.16 $\pm$ 6.63	49.41 $\pm$ 4.09	27.44 $\pm$ 0.89	40.72 $\pm$ 1.55	60.26 $\pm$ 2.50	61.89 $\pm$ 5.05	76.55 $\pm$ 1.23	87.30 $\pm$ 1.10	86.33 $\pm$ 0.48
MLP	80.81 $\pm$ 4.75	85.29 $\pm$ 3.31	36.53 $\pm$ 0.70	28.77 $\pm$ 1.56	46.21 $\pm$ 2.99	81.89 $\pm$ 6.40	74.02 $\pm$ 1.90	87.16 $\pm$ 0.37	75.69 $\pm$ 2.00

Table 6: Forward Time (ms) benchmark results for **Apply**. This table is grouped by Sheaf Convolution and Sheaf Diffusion models, expanding graph sizes (m) to avoid aggregating over varying sparsity. Results highlight Explicit materialization (Orig) vs Implicit evaluation (Ours). The best results are in **bold**.

Family	m	d=2 Norm			d=4 Norm			d=8 Norm		
		Orig	Ours	Spd	Orig	Ours	Spd	Orig	Ours	Spd
<i>Sheaf Convolution</i>										
Diagonal	19880	1.83 $\pm$ 0.04	1.80$\pm$0.04	1.01x	1.77$\pm$0.06	1.78 $\pm$ 0.05	1.00x	1.78 $\pm$ 0.05	1.77$\pm$0.03	1.01x
	97403	1.80$\pm$0.03	1.80 $\pm$ 0.04	1.00x	3.15 $\pm$ 0.02	2.91$\pm$0.02	1.08x	7.18 $\pm$ 0.03	6.92$\pm$0.03	1.04x
	189810	3.19 $\pm$ 0.02	2.97$\pm$0.01	1.07x	7.00$\pm$0.04	7.02 $\pm$ 0.03	1.00x	14.32 $\pm$ 0.09	13.32$\pm$0.08	1.07x
Bundle	19880	2.76$\pm$0.06	2.89 $\pm$ 0.05	0.96x	2.89$\pm$0.08	2.91 $\pm$ 0.08	0.99x	7.53 $\pm$ 0.06	4.97$\pm$0.02	1.52x
	97403	3.57 $\pm$ 0.02	3.49$\pm$0.01	1.02x	11.09 $\pm$ 0.04	9.72$\pm$0.02	1.14x	45.32 $\pm$ 0.34	27.61$\pm$0.18	1.64x
	189810	7.67$\pm$0.04	8.01 $\pm$ 0.05	0.96x	23.37 $\pm$ 0.12	19.41$\pm$0.06	1.20x	92.51 $\pm$ 0.78	56.65$\pm$0.23	1.63x
General	19880	3.42 $\pm$ 0.10	3.37$\pm$0.09	1.02x	4.66 $\pm$ 0.08	4.29$\pm$0.07	1.09x	13.83 $\pm$ 0.10	10.90$\pm$0.04	1.27x
	97403	3.99 $\pm$ 0.02	3.67$\pm$0.02	1.09x	12.06 $\pm$ 0.09	10.11$\pm$0.04	1.19x	58.91 $\pm$ 0.32	40.99$\pm$0.18	1.44x
	189810	7.32 $\pm$ 0.03	7.12$\pm$0.05	1.03x	24.59 $\pm$ 0.02	18.67$\pm$0.08	1.32x	117.70 $\pm$ 0.60	78.62$\pm$0.18	1.50x
<i>Sheaf Diffusion</i>										
Diagonal	19880	1.94 $\pm$ 0.05	1.76$\pm$0.13	1.10x	1.93 $\pm$ 0.04	1.70$\pm$0.05	1.14x	1.96 $\pm$ 0.04	1.70$\pm$0.01	1.16x
	97403	1.94 $\pm$ 0.01	1.70$\pm$0.00	1.14x	3.22 $\pm$ 0.02	2.50$\pm$0.02	1.29x	7.33 $\pm$ 0.04	5.95$\pm$0.02	1.23x
	189810	3.16 $\pm$ 0.01	2.73$\pm$0.01	1.16x	7.34 $\pm$ 0.01	5.92$\pm$0.01	1.24x	14.26 $\pm$ 0.04	11.40$\pm$0.04	1.25x
Bundle	19880	2.98 $\pm$ 0.00	2.71$\pm$0.02	1.10x	3.18 $\pm$ 0.02	2.66$\pm$0.03	1.20x	7.94 $\pm$ 0.03	4.68$\pm$0.04	1.70x
	97403	7.80 $\pm$ 0.08	3.30$\pm$0.09	2.37x	15.77 $\pm$ 0.17	9.48$\pm$0.04	1.66x	44.89 $\pm$ 0.15	26.09$\pm$0.25	1.72x
	189810	16.91 $\pm$ 0.06	7.75$\pm$0.04	2.18x	32.06 $\pm$ 0.04	18.40$\pm$0.04	1.74x	89.99 $\pm$ 0.27	53.00$\pm$0.05	1.70x
General	19880	6.48 $\pm$ 0.02	3.44$\pm$0.03	1.89x	7.67 $\pm$ 0.04	4.37$\pm$0.01	1.75x	12.74 $\pm$ 0.05	8.59$\pm$0.07	1.48x
	97403	18.58 $\pm$ 0.20	3.92$\pm$0.11	4.74x	27.40 $\pm$ 0.14	9.95$\pm$0.03	2.75x	52.29 $\pm$ 0.34	32.93$\pm$0.15	1.59x
	189810	36.56 $\pm$ 0.07	7.10$\pm$0.01	5.15x	51.25 $\pm$ 0.22	17.33$\pm$0.02	2.96x	102.35 $\pm$ 0.41	60.85$\pm$0.24	1.68x

Table 7: Forward Time (ms) benchmark results for **Apply**. This table is grouped by Sheaf Convolution and Sheaf Diffusion models, varying features f (with stalk dimension $d = 8$), expanding graph sizes (m) to avoid aggregating over varying sparsity. Results highlight Explicit materialization (Orig) vs Implicit evaluation (Ours). The best results are in **bold**.

Family	m	f=32 Norm			f=64 Norm		
		Orig	Ours	Spd	Orig	Ours	Spd
<i>Sheaf Convolution</i>							
Diagonal	19880	2.34 $\pm$ 0.01	2.25$\pm$0.01	1.04x	4.72$\pm$0.03	5.25 $\pm$ 0.04	0.90x
	97403	12.40$\pm$0.05	13.21 $\pm$ 0.04	0.94x	21.77$\pm$0.06	24.85 $\pm$ 0.07	0.88x
	189810	23.84$\pm$0.10	24.96 $\pm$ 0.11	0.96x	42.04$\pm$0.10	47.56 $\pm$ 0.13	0.88x
Bundle	19880	11.66 $\pm$ 0.07	6.68$\pm$0.04	1.74x	19.67 $\pm$ 0.09	10.93$\pm$0.07	1.80x
	97403	62.42 $\pm$ 0.38	34.66$\pm$0.18	1.80x	98.41 $\pm$ 0.30	51.57$\pm$0.17	1.91x
	189810	132.18 $\pm$ 0.77	72.61$\pm$0.19	1.82x	199.86 $\pm$ 0.55	105.72$\pm$0.49	1.89x
General	19880	17.33 $\pm$ 0.11	11.87$\pm$0.07	1.46x	24.63 $\pm$ 0.19	15.29$\pm$0.13	1.61x
	97403	74.96 $\pm$ 0.38	46.46$\pm$0.29	1.61x	106.57 $\pm$ 0.06	57.35$\pm$0.08	1.86x
	189810	151.26 $\pm$ 0.35	88.93$\pm$0.20	1.70x	210.72 $\pm$ 0.89	111.50$\pm$1.16	1.89x
<i>Sheaf Diffusion</i>							
Diagonal	19880	2.39 $\pm$ 0.02	2.07$\pm$0.02	1.16x	4.82 $\pm$ 0.05	4.59$\pm$0.04	1.05x
	97403	12.63 $\pm$ 0.08	11.50$\pm$0.06	1.10x	22.19 $\pm$ 0.08	21.44$\pm$0.07	1.03x
	189810	24.18 $\pm$ 0.17	21.52$\pm$0.19	1.12x	42.74 $\pm$ 0.25	40.95$\pm$0.25	1.04x
Bundle	19880	12.21 $\pm$ 0.09	6.77$\pm$0.09	1.81x	20.39 $\pm$ 0.42	11.14$\pm$0.45	1.83x
	97403	61.92 $\pm$ 0.23	33.45$\pm$0.23	1.85x	98.68 $\pm$ 0.38	50.22$\pm$0.23	1.96x
	189810	126.93 $\pm$ 0.24	69.44$\pm$0.15	1.83x	197.72 $\pm$ 0.77	102.44$\pm$0.26	1.93x
General	19880	16.70 $\pm$ 0.14	10.12$\pm$0.11	1.65x	24.03 $\pm$ 0.15	13.35$\pm$0.16	1.80x
	97403	68.46 $\pm$ 0.47	37.41$\pm$0.17	1.83x	100.62 $\pm$ 0.45	48.68$\pm$0.35	2.07x
	189810	134.95 $\pm$ 0.40	71.94$\pm$0.28	1.88x	195.40 $\pm$ 0.75	93.40$\pm$0.21	2.09x

Table 8: Backward Time (ms) benchmark results for **Apply**. This table is grouped by Sheaf Convolution and Sheaf Diffusion models, varying features f (with stalk dimension $d = 8$), expanding graph sizes (m) to avoid aggregating over varying sparsity. Results highlight Explicit materialization (Orig) vs Implicit evaluation (Ours). The best results are in **bold**.

		f=32 Norm			f=64 Norm		
Family	m	Orig	Ours	Spd	Orig	Ours	Spd
<i>Sheaf Convolution</i>							
Diagonal	19880	3.74$\pm$0.03	4.15 $\pm$ 0.03	0.90x	6.72$\pm$0.03	7.97 $\pm$ 0.04	0.84x
	97403	16.39$\pm$0.08	19.35 $\pm$ 0.08	0.85x	28.29$\pm$0.02	36.38 $\pm$ 0.09	0.78x
	189810	31.69$\pm$0.14	36.80 $\pm$ 0.15	0.86x	54.41$\pm$0.13	69.38 $\pm$ 0.17	0.78x
Bundle	19880	24.61 $\pm$ 0.17	12.14$\pm$0.08	2.03x	38.77 $\pm$ 0.20	16.77$\pm$0.14	2.31x
	97403	128.62 $\pm$ 0.60	62.18$\pm$0.33	2.07x	192.36 $\pm$ 0.53	81.52$\pm$0.29	2.36x
	189810	256.80 $\pm$ 1.46	130.06$\pm$0.82	1.97x	379.42 $\pm$ 1.70	165.86$\pm$1.20	2.29x
General	19880	21.43 $\pm$ 0.15	6.85$\pm$0.08	3.13x	34.46 $\pm$ 0.23	10.13$\pm$0.09	3.40x
	97403	102.61 $\pm$ 0.37	31.60$\pm$0.13	3.25x	160.74 $\pm$ 0.26	43.84$\pm$0.09	3.67x
	189810	203.76 $\pm$ 1.24	64.42$\pm$0.32	3.16x	317.02 $\pm$ 0.77	88.11$\pm$0.38	3.60x
<i>Sheaf Diffusion</i>							
Diagonal	19880	3.41 $\pm$ 0.03	3.25$\pm$0.02	1.05x	6.47 $\pm$ 0.05	6.25$\pm$0.04	1.04x
	97403	15.02 $\pm$ 0.06	14.22$\pm$0.05	1.06x	26.92 $\pm$ 0.07	26.38$\pm$0.07	1.02x
	189810	28.21 $\pm$ 0.14	26.59$\pm$0.16	1.06x	50.90 $\pm$ 0.27	49.91$\pm$0.28	1.02x
Bundle	19880	19.07 $\pm$ 0.09	9.71$\pm$0.05	1.96x	32.35 $\pm$ 0.22	14.19$\pm$0.14	2.28x
	97403	92.04 $\pm$ 0.23	47.71$\pm$0.06	1.93x	154.05 $\pm$ 0.39	66.45$\pm$0.33	2.32x
	189810	187.30 $\pm$ 0.12	99.90$\pm$0.18	1.87x	308.27 $\pm$ 0.84	135.04$\pm$0.38	2.28x
General	19880	19.16 $\pm$ 0.09	7.07$\pm$0.04	2.71x	31.86 $\pm$ 0.22	10.37$\pm$0.09	3.07x
	97403	88.87 $\pm$ 0.37	31.92$\pm$0.04	2.78x	146.47 $\pm$ 0.39	44.62$\pm$0.07	3.28x
	189810	176.46 $\pm$ 0.38	64.62$\pm$0.22	2.73x	287.80 $\pm$ 0.81	89.14$\pm$0.18	3.23x

Table 9: Peak Memory (MB) benchmark results for **Apply**. This table is grouped by Sheaf Convolution and Sheaf Diffusion models, varying features f (with stalk dimension $d = 8$), expanding graph sizes (m) to avoid aggregating over varying sparsity. Results highlight Explicit materialization (Orig) vs Implicit evaluation (Ours). The best results are in **bold**.

Family	m	f=32 Norm			f=64 Norm		
		Orig	Ours	MemR	Orig	Ours	MemR
<i>Sheaf Convolution</i>							
Diagonal	19880	85.11 $\pm$ 0.81	81.32$\pm$0.79	1.05x	163.39 $\pm$ 1.41	159.39$\pm$1.57	1.03x
	97403	406.23 $\pm$ 1.57	388.52$\pm$1.77	1.05x	788.56 $\pm$ 1.55	770.21$\pm$1.50	1.02x
	189810	790.57 $\pm$ 1.46	755.63$\pm$1.35	1.05x	1532.41 $\pm$ 4.74	1497.43$\pm$4.70	1.02x
Bundle	19880	361.20 $\pm$ 1.96	88.81$\pm$0.65	4.07x	689.70 $\pm$ 4.14	167.99$\pm$0.67	4.11x
	97403	1697.60 $\pm$ 5.30	432.39$\pm$1.35	3.93x	3236.90 $\pm$ 10.05	814.68$\pm$2.07	3.97x
	189810	3291.89 $\pm$ 10.38	842.03$\pm$2.66	3.91x	6277.52 $\pm$ 20.17	1585.74$\pm$5.02	3.96x
General	19880	362.97 $\pm$ 2.23	88.31$\pm$0.58	4.11x	691.51 $\pm$ 3.84	167.58$\pm$0.76	4.13x
	97403	1707.38 $\pm$ 5.14	429.04$\pm$1.34	3.98x	3247.48 $\pm$ 10.34	811.01$\pm$2.53	4.00x
	189810	3311.11 $\pm$ 10.01	835.51$\pm$2.64	3.96x	6297.06 $\pm$ 20.40	1578.27$\pm$4.99	3.99x
<i>Sheaf Diffusion</i>							
Diagonal	19880	84.80 $\pm$ 0.81	81.32$\pm$0.79	1.04x	162.88 $\pm$ 1.59	159.39$\pm$1.57	1.02x
	97403	405.02 $\pm$ 1.82	388.53$\pm$1.75	1.04x	786.76 $\pm$ 1.67	770.21$\pm$1.85	1.02x
	189810	787.62 $\pm$ 1.52	755.75$\pm$1.40	1.04x	1529.99 $\pm$ 4.81	1497.14$\pm$4.73	1.02x
Bundle	19880	344.54 $\pm$ 1.15	89.50$\pm$1.06	3.85x	659.07 $\pm$ 3.71	168.51$\pm$1.85	3.91x
	97403	1672.33 $\pm$ 5.21	432.89$\pm$1.89	3.86x	3197.09 $\pm$ 10.79	814.37$\pm$3.41	3.93x
	189810	3253.39 $\pm$ 10.16	842.06$\pm$2.62	3.86x	6225.63 $\pm$ 19.78	1584.76$\pm$5.01	3.93x
General	19880	360.91 $\pm$ 2.11	89.49$\pm$1.29	4.03x	689.41 $\pm$ 3.99	166.91$\pm$0.86	4.13x
	97403	1695.66 $\pm$ 5.45	429.61$\pm$1.95	3.95x	3235.08 $\pm$ 10.22	811.41$\pm$1.73	3.99x
	189810	3288.01 $\pm$ 9.87	836.02$\pm$2.19	3.93x	6273.91 $\pm$ 20.09	1578.83$\pm$4.96	3.97x

Table 10: Backward Time (ms) benchmark results for **Apply**. This table is grouped by Sheaf Convolution and Sheaf Diffusion models, expanding graph sizes (m) to avoid aggregating over varying sparsity. Results highlight Explicit materialization (Orig) vs Implicit evaluation (Ours). The best results are in **bold**.

Family	m	d=2 Norm			d=4 Norm			d=8 Norm		
		Orig	Ours	Spd	Orig	Ours	Spd	Orig	Ours	Spd
<i>Sheaf Convolution</i>										
Diagonal	19880	2.52$\pm$0.06	2.72 $\pm$ 0.06	0.93x	2.50$\pm$0.04	2.74 $\pm$ 0.03	0.91x	2.61$\pm$0.07	2.78 $\pm$ 0.03	0.94x
	97403	2.72$\pm$0.05	2.89 $\pm$ 0.03	0.94x	5.41$\pm$0.01	5.45 $\pm$ 0.02	0.99x	10.60$\pm$0.03	10.75 $\pm$ 0.04	0.99x
	189810	5.32$\pm$0.03	5.52 $\pm$ 0.05	0.96x	10.21$\pm$0.06	10.79 $\pm$ 0.07	0.95x	20.49 $\pm$ 0.12	20.06$\pm$0.08	1.02x
Bundle	19880	4.18$\pm$0.02	4.42 $\pm$ 0.04	0.95x	5.92 $\pm$ 0.02	5.21$\pm$0.00	1.13x	18.27 $\pm$ 0.15	10.74$\pm$0.06	1.70x
	97403	15.21 $\pm$ 0.09	14.65$\pm$0.09	1.04x	29.46 $\pm$ 0.09	25.68$\pm$0.05	1.15x	97.88 $\pm$ 0.57	56.43$\pm$0.40	1.73x
	189810	30.01 $\pm$ 0.08	29.70$\pm$0.10	1.01x	58.87 $\pm$ 0.44	51.46$\pm$0.22	1.14x	194.93 $\pm$ 0.59	113.38$\pm$0.74	1.72x
General	19880	3.93 $\pm$ 0.06	3.49$\pm$0.05	1.13x	3.95 $\pm$ 0.10	3.51$\pm$0.09	1.13x	15.63 $\pm$ 0.09	6.05$\pm$0.02	2.58x
	97403	4.65 $\pm$ 0.00	4.24$\pm$0.02	1.10x	16.22 $\pm$ 0.05	12.98$\pm$0.02	1.25x	74.16 $\pm$ 0.24	26.60$\pm$0.22	2.79x
	189810	8.52 $\pm$ 0.03	8.51$\pm$0.03	1.00x	32.56 $\pm$ 0.06	25.27$\pm$0.12	1.29x	147.66 $\pm$ 0.51	54.25$\pm$0.27	2.72x
<i>Sheaf Diffusion</i>										
Diagonal	19880	2.57 $\pm$ 0.04	2.55$\pm$0.02	1.01x	2.64 $\pm$ 0.10	2.61$\pm$0.03	1.01x	2.64$\pm$0.03	2.70 $\pm$ 0.19	0.98x
	97403	2.66$\pm$0.04	2.68 $\pm$ 0.05	0.99x	4.68 $\pm$ 0.02	4.06$\pm$0.02	1.15x	9.04 $\pm$ 0.03	8.02$\pm$0.04	1.13x
	189810	4.47 $\pm$ 0.02	4.07$\pm$0.02	1.10x	8.90 $\pm$ 0.04	8.05$\pm$0.01	1.11x	16.53 $\pm$ 0.05	14.58$\pm$0.03	1.13x
Bundle	19880	4.34 $\pm$ 0.06	4.23$\pm$0.06	1.03x	5.24 $\pm$ 0.06	4.98$\pm$0.07	1.05x	12.55 $\pm$ 0.03	8.01$\pm$0.03	1.57x
	97403	14.45 $\pm$ 0.09	14.19$\pm$0.08	1.02x	24.46 $\pm$ 0.22	23.71$\pm$0.20	1.03x	61.45 $\pm$ 0.33	41.55$\pm$0.38	1.48x
	189810	28.45$\pm$0.20	28.57 $\pm$ 0.22	1.00x	47.14$\pm$0.10	47.19 $\pm$ 0.17	1.00x	125.22 $\pm$ 0.26	83.16$\pm$0.15	1.51x
General	19880	4.31 $\pm$ 0.12	3.78$\pm$0.05	1.14x	5.67 $\pm$ 0.05	3.69$\pm$0.04	1.53x	12.96 $\pm$ 0.10	5.99$\pm$0.10	2.16x
	97403	14.96 $\pm$ 0.10	4.34$\pm$0.00	3.44x	24.44 $\pm$ 0.11	12.99$\pm$0.03	1.88x	59.26 $\pm$ 0.22	27.05$\pm$0.12	2.19x
	189810	29.28 $\pm$ 0.03	8.47$\pm$0.02	3.46x	45.97 $\pm$ 0.11	25.10$\pm$0.02	1.83x	117.87 $\pm$ 0.76	53.86$\pm$0.32	2.19x

Table 11: Peak Memory (MB) benchmark results for **Apply**. This table is grouped by Sheaf Convolution and Sheaf Diffusion models, expanding graph sizes (m) to avoid aggregating over varying sparsity. Results highlight Explicit materialization (Orig) vs Implicit evaluation (Ours). The best results are in **bold**.

		d=2 Norm			d=4 Norm			d=8 Norm		
Family	m	Orig	Ours	MemR	Orig	Ours	MemR	Orig	Ours	MemR
<i>Sheaf Convolution</i>										
Diagonal	19880	11.07±0.06	10.12±0.06	1.09x	22.14±0.12	20.25±0.11	1.09x	44.90±0.14	41.11±0.12	1.09x
	97403	54.77±0.06	49.58±0.08	1.10x	108.60±0.03	99.63±0.29	1.09x	216.39±0.95	198.49±0.62	1.09x
	189810	106.63±0.19	98.02±0.15	1.09x	209.28±1.20	191.81±1.15	1.09x	419.22±1.80	384.51±1.74	1.09x
Bundle	19880	12.63±0.07	13.52±0.07	0.93x	50.20±0.46	38.85±0.14	1.29x	198.51±0.23	50.99±0.42	3.89x
	97403	60.13±0.08	65.34±0.13	0.92x	233.38±0.47	184.73±0.80	1.26x	928.96±2.15	242.52±1.23	3.83x
	189810	116.49±0.25	128.11±0.16	0.91x	451.77±2.48	357.53±0.95	1.26x	1799.19±5.82	471.44±2.03	3.82x
General	19880	12.63±0.07	13.22±0.07	0.96x	51.25±0.05	38.28±0.56	1.34x	200.75±0.22	49.94±0.41	4.02x
	97403	60.17±0.11	63.44±0.13	0.95x	235.59±0.44	180.70±0.51	1.30x	938.77±2.30	238.32±0.75	3.94x
	189810	116.72±0.69	124.54±0.32	0.94x	455.62±2.86	349.48±0.60	1.30x	1818.19±5.67	464.13±1.47	3.92x
<i>Sheaf Diffusion</i>										
Diagonal	19880	11.00±0.06	10.12±0.06	1.09x	21.99±0.12	20.25±0.11	1.09x	44.60±0.13	41.11±0.12	1.08x
	97403	53.69±0.11	49.58±0.08	1.08x	108.47±0.50	100.41±0.24	1.08x	214.89±0.55	198.06±0.73	1.08x
	189810	104.32±0.45	97.25±0.13	1.07x	208.19±1.31	191.91±1.07	1.08x	416.51±2.33	384.22±1.78	1.08x
Bundle	19880	12.19±0.07	13.39±0.07	0.91x	48.96±0.02	38.53±0.10	1.27x	187.61±0.70	50.49±0.12	3.72x
	97403	59.55±0.21	65.30±0.24	0.91x	229.79±0.06	184.59±0.94	1.24x	908.94±2.87	243.18±0.78	3.74x
	189810	115.24±0.64	128.42±0.31	0.90x	444.67±0.75	357.17±0.65	1.24x	1768.58±5.10	471.39±0.83	3.75x
General	19880	12.47±0.07	13.09±0.07	0.95x	50.35±0.25	38.07±0.40	1.32x	197.96±0.38	50.06±0.68	3.95x
	97403	59.17±0.07	63.34±0.01	0.93x	232.08±0.65	180.49±0.80	1.29x	927.09±2.22	238.85±0.59	3.88x
	189810	116.25±0.24	124.52±0.16	0.93x	449.58±2.45	349.35±1.12	1.29x	1795.11±5.47	464.17±1.41	3.87x

Metrics A background daemon (`tegrastats`) captured the board-level power draw at 100 ms intervals over a continuous minimum 5-second window. The final metrics are averaged across 3 independent seeds:

- **Net Power (W)**: The average dynamic power drawn by the model, obtained by subtracting the static no-load idle power from the gross board power.
- **Peak Memory (MB)**: The maximum CUDA memory allocated during the workload, isolating the dynamic footprint.
- **Throughput (graphs/s)**: The processing speed, computed as the exact number of graphs divided by the synchronized wall-clock time.
- **Energy/Graph (mJ)**: The net energy consumed to process a single graph, effectively normalizing battery drain over batch size.

Hyperparameters

Configuration

For Pei et al. (2020) node classification we used Bodnar et al. (2022) script values to execute a 200 run random sweep. The runs were executed on a NVIDIA L4 via the AWS EC2 g6 instance running UBUNTU 24.04 LTS. The library versions are PyTorch 2.8, PyG-Lib 0.5, Torch Sparse 0.6.18 & Torch Scatter 2.1.2. The reported results in the tables are taken from the best validation set result.

Squirrel

- d: 3
- layers: 5
- hidden_channels: 32
- lr: 0.01
- weight_decay: 0.00011215791366362148
- input_dropout: 0.7
- dropout: 0
- add_lp: True
- second_linear: True
- left_weights: True
- right_weights: True
- use_act: True
- normalized: True
- edge_weights: True

Chameleon

- d: 4
- layers: 5
- hidden_channels: 32
- lr: 0.01
- weight_decay: 0.0002969905682317406
- sheaf_decay: 0.0012638885974822734
- input_dropout: 0.7

Table 12: Forward Time (ms) inductive ablation results for stalk dimension $d = 2$. Performance comparison of the Explicit operator builder with varying graph-update strategies (Explicit original vs Explicit optimized) against the Implicit formulation across graph densities m . The best results are in **bold**.

Stalk Dimension $d = 2$							
Family	m	Explicit original	Explicit optimized		Implicit fully optimized		
		Value	Value	Spd (vs Orig)	Value	Spd (vs Opt)	Spd (vs Orig)
<i>Sheaf Convolution</i>							
Diagonal	19880	904.77 $\pm$ 17.91	2.22 $\pm$ 0.08	406.75x	2.07 $\pm$ 0.08	1.07x	436.23x
	97403	4363.15 $\pm$ 67.54	2.28 $\pm$ 0.04	1915.95x	2.10 $\pm$ 0.02	1.09x	2081.90x
	189810	8668.74 $\pm$ 56.89	3.62 $\pm$ 0.01	2397.63x	3.26 $\pm$ 0.02	1.11x	2659.81x
Bundle	19880	2701.49 $\pm$ 70.02	3.99 $\pm$ 0.06	676.93x	3.86 $\pm$ 0.06	1.03x	699.45x
	97403	13196.65 $\pm$ 162.18	4.90 $\pm$ 0.02	2695.41x	4.54 $\pm$ 0.03	1.08x	2907.89x
	189810	25850.88 $\pm$ 530.53	9.12 $\pm$ 0.03	2834.10x	9.14 $\pm$ 0.03	1.00x	2829.63x
General	19880	899.48 $\pm$ 14.74	4.08 $\pm$ 0.08	220.24x	3.78 $\pm$ 0.14	1.08x	238.06x
	97403	4394.20 $\pm$ 60.95	4.57 $\pm$ 0.04	962.07x	3.96 $\pm$ 0.02	1.15x	1109.33x
	189810	8522.16 $\pm$ 182.23	7.91 $\pm$ 0.01	1077.22x	7.39 $\pm$ 0.02	1.07x	1153.19x
<i>Sheaf Diffusion</i>							
Diagonal	19880	1848.18 $\pm$ 58.54	3.12 $\pm$ 0.03	592.66x	2.17 $\pm$ 0.02	1.44x	853.26x
	97403	9050.49 $\pm$ 269.04	3.31 $\pm$ 0.01	2736.27x	2.27 $\pm$ 0.01	1.46x	3986.80x
	189810	17542.31 $\pm$ 432.85	4.50 $\pm$ 0.01	3900.02x	3.27 $\pm$ 0.00	1.38x	5371.72x
Bundle	19880	3717.90 $\pm$ 130.27	5.27 $\pm$ 0.02	705.30x	3.94 $\pm$ 0.01	1.34x	943.05x
	97403	17803.26 $\pm$ 448.05	10.11 $\pm$ 0.06	1760.92x	4.59 $\pm$ 0.01	2.20x	3876.93x
	189810	34768.95 $\pm$ 247.33	19.38 $\pm$ 0.20	1794.14x	9.16 $\pm$ 0.07	2.11x	3794.16x
General	19880	1885.07 $\pm$ 64.62	7.87 $\pm$ 0.09	239.51x	3.99 $\pm$ 0.07	1.97x	471.86x
	97403	8932.16 $\pm$ 114.44	19.93 $\pm$ 0.15	448.07x	4.53 $\pm$ 0.09	4.40x	1973.58x
	189810	17980.53 $\pm$ 221.29	37.78 $\pm$ 0.21	475.88x	7.76 $\pm$ 0.08	4.87x	2317.55x

Table 13: Forward Time (ms) inductive ablation results for stalk dimension $d = 4$. Performance comparison of the Explicit operator builder with varying graph-update strategies (Explicit original vs Explicit optimized) against the Implicit formulation across graph densities m . The best results are in **bold**.

Stalk Dimension $d = 4$							
Family	m	Explicit original	Explicit optimized		Implicit fully optimized		
		Value	Value	Spd (vs Orig)	Value	Spd (vs Opt)	Spd (vs Orig)
<i>Sheaf Convolution</i>							
Diagonal	19880	896.96 $\pm$ 19.50	2.25 $\pm$ 0.07	397.94x	2.05 $\pm$ 0.03	1.10x	437.24x
	97403	4435.47 $\pm$ 14.66	3.60 $\pm$ 0.01	1231.68x	3.20 $\pm$ 0.01	1.12x	1385.19x
	189810	8533.18 $\pm$ 198.13	7.49 $\pm$ 0.02	1139.34x	7.33 $\pm$ 0.02	1.02x	1164.13x
Bundle	19880	2672.46 $\pm$ 44.95	4.16 $\pm$ 0.14	642.26x	3.90 $\pm$ 0.17	1.07x	684.45x
	97403	13245.56 $\pm$ 33.64	12.40 $\pm$ 0.02	1068.00x	10.81 $\pm$ 0.04	1.15x	1224.83x
	189810	25600.76 $\pm$ 311.83	24.73 $\pm$ 0.15	1035.34x	20.64 $\pm$ 0.15	1.20x	1240.14x
General	19880	916.33 $\pm$ 6.24	5.29 $\pm$ 0.08	173.20x	4.62 $\pm$ 0.10	1.15x	198.42x
	97403	4440.86 $\pm$ 19.37	12.60 $\pm$ 0.14	352.38x	10.36 $\pm$ 0.05	1.22x	428.78x
	189810	8661.26 $\pm$ 62.13	25.31 $\pm$ 0.10	342.16x	18.94 $\pm$ 0.04	1.34x	457.24x
<i>Sheaf Diffusion</i>							
Diagonal	19880	1842.50 $\pm$ 76.07	3.22 $\pm$ 0.05	572.15x	2.21 $\pm$ 0.02	1.46x	833.08x
	97403	9055.26 $\pm$ 212.78	4.53 $\pm$ 0.03	1998.15x	3.06 $\pm$ 0.02	1.48x	2955.60x
	189810	17857.10 $\pm$ 132.85	8.83 $\pm$ 0.06	2021.77x	6.58 $\pm$ 0.04	1.34x	2712.87x
Bundle	19880	3704.96 $\pm$ 155.49	5.45 $\pm$ 0.04	679.45x	3.90 $\pm$ 0.03	1.40x	948.91x
	97403	18277.27 $\pm$ 16.69	18.08 $\pm$ 0.16	1011.16x	10.81 $\pm$ 0.04	1.67x	1691.46x
	189810	34812.43 $\pm$ 903.42	34.53 $\pm$ 0.34	1008.22x	19.90 $\pm$ 0.09	1.74x	1749.50x
General	19880	1819.94 $\pm$ 57.97	9.01 $\pm$ 0.02	202.01x	4.91 $\pm$ 0.04	1.83x	370.66x
	97403	9068.77 $\pm$ 358.65	28.57 $\pm$ 0.18	317.39x	10.77 $\pm$ 0.41	2.65x	841.92x
	189810	17064.87 $\pm$ 40.95	52.75 $\pm$ 0.28	323.51x	17.96 $\pm$ 0.11	2.94x	950.25x

Table 14: Forward Time (ms) inductive ablation results for stalk dimension $d = 8$. Performance comparison of the Explicit operator builder with varying graph-update strategies (Explicit original vs Explicit optimized) against the Implicit formulation across graph densities m . The best results are in **bold**.

Stalk Dimension $d = 8$							
Family	m	Explicit original	Explicit optimized		Implicit fully optimized		
		Value	Value	Spd (vs Orig)	Value	Spd (vs Opt)	Spd (vs Orig)
<i>Sheaf Convolution</i>							
Diagonal	19880	913.27 $\pm$ 6.55	2.27 $\pm$ 0.07	402.55x	2.09 $\pm$ 0.05	1.09x	437.79x
	97403	4443.28 $\pm$ 14.98	7.69 $\pm$ 0.02	577.64x	7.25 $\pm$ 0.01	1.06x	613.03x
	189810	8674.83 $\pm$ 8.96	14.80 $\pm$ 0.04	586.12x	13.63 $\pm$ 0.03	1.09x	636.39x
Bundle	19880	2675.04 $\pm$ 56.56	8.71 $\pm$ 0.05	307.25x	5.88 $\pm$ 0.02	1.48x	454.58x
	97403	13350.76 $\pm$ 120.23	46.47 $\pm$ 0.25	287.27x	28.63 $\pm$ 0.18	1.62x	466.25x
	189810	25905.62 $\pm$ 360.72	94.96 $\pm$ 0.66	272.81x	58.04 $\pm$ 0.19	1.64x	446.31x
General	19880	918.13 $\pm$ 12.07	14.40 $\pm$ 0.06	63.77x	11.12 $\pm$ 0.06	1.29x	82.54x
	97403	4471.57 $\pm$ 22.51	59.45 $\pm$ 0.35	75.21x	41.38 $\pm$ 0.17	1.44x	108.06x
	189810	8719.75 $\pm$ 131.51	119.14 $\pm$ 0.58	73.19x	79.16 $\pm$ 0.14	1.51x	110.15x
<i>Sheaf Diffusion</i>							
Diagonal	19880	1813.13 $\pm$ 40.73	3.24 $\pm$ 0.07	560.36x	2.23 $\pm$ 0.02	1.45x	811.37x
	97403	9149.54 $\pm$ 86.67	8.70 $\pm$ 0.05	1051.41x	6.54 $\pm$ 0.02	1.33x	1398.40x
	189810	17660.98 $\pm$ 616.22	15.74 $\pm$ 0.01	1122.06x	12.04 $\pm$ 0.01	1.31x	1466.86x
Bundle	19880	3650.08 $\pm$ 70.38	10.06 $\pm$ 0.13	362.65x	5.80 $\pm$ 0.07	1.73x	629.10x
	97403	17676.31 $\pm$ 613.21	47.60 $\pm$ 0.57	371.32x	27.47 $\pm$ 0.34	1.73x	643.49x
	189810	34721.66 $\pm$ 1156.78	94.22 $\pm$ 0.53	368.52x	54.57 $\pm$ 0.37	1.73x	636.30x
General	19880	1801.43 $\pm$ 7.34	14.04 $\pm$ 0.14	128.32x	9.07 $\pm$ 0.08	1.55x	198.57x
	97403	8882.65 $\pm$ 345.64	53.95 $\pm$ 0.13	164.65x	32.34 $\pm$ 0.12	1.67x	274.69x
	189810	17749.20 $\pm$ 424.71	105.07 $\pm$ 0.76	168.93x	61.56 $\pm$ 0.49	1.71x	288.32x

Table 15: Forward Time (ms) inductive ablation results for stalk dimension $d = 8$ and feature dimension $f = 32$. Performance comparison of the Explicit operator builder with varying graph-update strategies (Explicit original vs Explicit optimized) against the Implicit formulation across graph densities m . The best results are in **bold**.

Stalk Dimension $d = 8, f = 32$							
Family	m	Explicit original	Explicit optimized		Implicit fully optimized		
		Value	Value	Spd (vs Orig)	Value	Spd (vs Opt)	Spd (vs Orig)
<i>Sheaf Convolution</i>							
Diagonal	19880	949.05 $\pm$ 15.77	2.79 $\pm$ 0.03	339.69x	2.55$\pm$0.02	1.10x	372.73x
	97403	4604.54 $\pm$ 19.63	12.93$\pm$0.03	356.11x	13.56 $\pm$ 0.04	0.95x	339.61x
	189810	8998.10 $\pm$ 80.64	24.39$\pm$0.07	368.86x	25.30 $\pm$ 0.05	0.96x	355.62x
Bundle	19880	2823.27 $\pm$ 31.34	12.87 $\pm$ 0.11	219.29x	7.61$\pm$0.07	1.69x	371.19x
	97403	13722.73 $\pm$ 95.50	63.93 $\pm$ 0.20	214.64x	35.79$\pm$0.14	1.79x	383.46x
	189810	27017.19 $\pm$ 90.05	134.54 $\pm$ 0.61	200.81x	73.99$\pm$0.27	1.82x	365.16x
General	19880	957.10 $\pm$ 10.12	17.96 $\pm$ 0.11	53.29x	12.20$\pm$0.06	1.47x	78.46x
	97403	4628.21 $\pm$ 17.59	75.79 $\pm$ 0.15	61.06x	46.70$\pm$0.25	1.62x	99.11x
	189810	9110.12 $\pm$ 25.32	152.50 $\pm$ 0.21	59.74x	89.60$\pm$0.32	1.70x	101.67x
<i>Sheaf Diffusion</i>							
Diagonal	19880	1829.54 $\pm$ 69.30	3.75 $\pm$ 0.06	488.37x	2.63$\pm$0.01	1.42x	695.43x
	97403	8862.79 $\pm$ 202.65	14.33 $\pm$ 0.09	618.52x	12.28$\pm$0.05	1.17x	721.44x
	189810	17321.00 $\pm$ 452.95	25.85 $\pm$ 0.24	669.97x	22.32$\pm$0.25	1.16x	776.15x
Bundle	19880	3635.01 $\pm$ 78.45	14.61 $\pm$ 0.10	248.88x	8.08$\pm$0.13	1.81x	449.66x
	97403	17637.76 $\pm$ 421.22	65.13 $\pm$ 0.09	270.81x	35.16$\pm$0.05	1.85x	501.60x
	189810	35360.09 $\pm$ 333.40	131.03 $\pm$ 0.62	269.87x	70.93$\pm$0.33	1.85x	498.50x
General	19880	1848.39 $\pm$ 68.10	18.25 $\pm$ 0.19	101.30x	10.81$\pm$0.20	1.69x	171.03x
	97403	8900.85 $\pm$ 139.61	70.35 $\pm$ 0.33	126.52x	38.26$\pm$0.14	1.84x	232.66x
	189810	17412.25 $\pm$ 535.23	137.62 $\pm$ 0.44	126.52x	72.79$\pm$0.28	1.89x	239.21x

Table 16: Forward Time (ms) inductive ablation results for stalk dimension $d = 8$ and feature dimension $f = 64$. Performance comparison of the Explicit operator builder with varying graph-update strategies (Explicit original vs Explicit optimized) against the Implicit formulation across graph densities m . The best results are in **bold**.

Stalk Dimension $d = 8, f = 64$							
Family	m	Explicit original	Explicit optimized		Implicit fully optimized		
		Value	Value	Spd (vs Orig)	Value	Spd (vs Opt)	Spd (vs Orig)
<i>Sheaf Convolution</i>							
Diagonal	19880	960.41 $\pm$ 35.22	5.20$\pm$0.04	184.67x	5.55 $\pm$ 0.03	0.94x	173.10x
	97403	4647.17 $\pm$ 200.98	22.32$\pm$0.05	208.25x	25.21 $\pm$ 0.04	0.89x	184.31x
	189810	8749.86 $\pm$ 92.36	42.60$\pm$0.06	205.38x	47.93 $\pm$ 0.06	0.89x	182.55x
Bundle	19880	2737.49 $\pm$ 61.58	20.94 $\pm$ 0.18	130.76x	11.90$\pm$0.10	1.76x	230.03x
	97403	13244.14 $\pm$ 81.32	100.23 $\pm$ 0.40	132.13x	52.77$\pm$0.23	1.90x	250.97x
	189810	26307.83 $\pm$ 393.44	202.87 $\pm$ 0.68	129.68x	107.30$\pm$0.65	1.89x	245.18x
General	19880	980.03 $\pm$ 47.36	25.29 $\pm$ 0.17	38.76x	15.69$\pm$0.15	1.61x	62.48x
	97403	4610.82 $\pm$ 181.27	106.99 $\pm$ 0.54	43.09x	57.62$\pm$0.24	1.86x	80.02x
	189810	9175.09 $\pm$ 320.49	212.59 $\pm$ 2.18	43.16x	112.38$\pm$1.50	1.89x	81.64x
<i>Sheaf Diffusion</i>							
Diagonal	19880	1850.07 $\pm$ 22.37	6.20 $\pm$ 0.06	298.52x	5.16$\pm$0.01	1.20x	358.79x
	97403	8915.33 $\pm$ 12.57	23.84 $\pm$ 0.28	374.04x	22.19$\pm$0.26	1.07x	401.71x
	189810	17429.92 $\pm$ 414.73	44.58 $\pm$ 0.35	390.97x	41.83$\pm$0.33	1.07x	416.69x
Bundle	19880	3625.53 $\pm$ 79.31	22.97 $\pm$ 0.92	157.83x	12.52$\pm$0.81	1.83x	289.48x
	97403	17709.97 $\pm$ 152.68	102.00 $\pm$ 0.32	173.63x	51.76$\pm$0.21	1.97x	342.15x
	189810	34518.72 $\pm$ 467.54	203.09 $\pm$ 1.19	169.97x	104.43$\pm$0.54	1.94x	330.54x
General	19880	1852.39 $\pm$ 34.30	25.54 $\pm$ 0.44	72.52x	13.93$\pm$0.29	1.83x	132.97x
	97403	8970.53 $\pm$ 154.37	102.51 $\pm$ 0.12	87.51x	49.38$\pm$0.20	2.08x	181.65x
	189810	17411.19 $\pm$ 226.86	198.75 $\pm$ 1.27	87.60x	94.29$\pm$0.37	2.11x	184.65x

Table 17: Backward Time (ms) inductive ablation results for stalk dimension $d = 2$. Performance comparison of the Explicit operator builder with varying graph-update strategies (Explicit original vs Explicit optimized) against the Implicit formulation across graph densities m . The best results are in **bold**.

Stalk Dimension $d = 2$							
Family	m	Explicit original	Explicit optimized		Implicit fully optimized		
		Value	Value	Spd (vs Orig)	Value	Spd (vs Opt)	Spd (vs Orig)
<i>Sheaf Convolution</i>							
Diagonal	19880	2.86 $\pm$ 0.07	2.49$\pm$0.04	1.15x	2.78 $\pm$ 0.08	0.90x	1.03x
	97403	3.02 $\pm$ 0.11	2.70$\pm$0.03	1.12x	2.95 $\pm$ 0.03	0.91x	1.02x
	189810	5.33 $\pm$ 0.03	5.28$\pm$0.03	1.01x	5.48 $\pm$ 0.03	0.96x	0.97x
Bundle	19880	4.49 $\pm$ 0.04	4.18$\pm$0.04	1.07x	4.45 $\pm$ 0.07	0.94x	1.01x
	97403	14.54 $\pm$ 0.06	14.93 $\pm$ 0.11	0.97x	14.41$\pm$0.10	1.04x	1.01x
	189810	27.91$\pm$0.07	29.88 $\pm$ 0.08	0.93x	29.58 $\pm$ 0.10	1.01x	0.94x
General	19880	4.34 $\pm$ 0.07	3.96 $\pm$ 0.07	1.10x	3.52$\pm$0.08	1.12x	1.23x
	97403	4.81 $\pm$ 0.04	4.64 $\pm$ 0.01	1.04x	4.25$\pm$0.01	1.09x	1.13x
	189810	8.47$\pm$0.02	8.48 $\pm$ 0.01	1.00x	8.48 $\pm$ 0.03	1.00x	1.00x
<i>Sheaf Diffusion</i>							
Diagonal	19880	2.90 $\pm$ 0.03	2.54 $\pm$ 0.01	1.14x	2.54$\pm$0.01	1.00x	1.14x
	97403	2.92 $\pm$ 0.07	2.74 $\pm$ 0.19	1.06x	2.67$\pm$0.05	1.03x	1.09x
	189810	4.57 $\pm$ 0.03	4.38 $\pm$ 0.02	1.04x	4.00$\pm$0.03	1.10x	1.14x
Bundle	19880	4.64 $\pm$ 0.08	4.31 $\pm$ 0.05	1.08x	4.25$\pm$0.09	1.01x	1.09x
	97403	14.09 $\pm$ 0.06	14.17 $\pm$ 0.04	0.99x	13.92$\pm$0.07	1.02x	1.01x
	189810	26.96$\pm$0.05	28.09 $\pm$ 0.39	0.96x	28.36 $\pm$ 0.41	0.99x	0.95x
General	19880	4.48 $\pm$ 0.09	4.35 $\pm$ 0.09	1.03x	3.78$\pm$0.06	1.15x	1.19x
	97403	14.94 $\pm$ 0.07	14.87 $\pm$ 0.05	1.00x	4.33$\pm$0.02	3.44x	3.45x
	189810	28.27 $\pm$ 0.07	29.13 $\pm$ 0.18	0.97x	8.47$\pm$0.05	3.44x	3.34x

Table 18: Backward Time (ms) inductive ablation results for stalk dimension $d = 4$. Performance comparison of the Explicit operator builder with varying graph-update strategies (Explicit original vs Explicit optimized) against the Implicit formulation across graph densities m . The best results are in **bold**.

Stalk Dimension $d = 4$							
Family	m	Explicit original	Explicit optimized		Implicit fully optimized		
		Value	Value	Spd (vs Orig)	Value	Spd (vs Opt)	Spd (vs Orig)
<i>Sheaf Convolution</i>							
Diagonal	19880	2.92±0.04	2.62±0.16	1.12x	2.76±0.05	0.95x	1.06x
	97403	5.41±0.02	5.37±0.03	1.01x	5.45±0.04	0.99x	0.99x
	189810	10.06±0.04	10.21±0.04	0.99x	10.77±0.03	0.95x	0.93x
Bundle	19880	6.02±0.06	5.90±0.07	1.02x	5.26±0.01	1.12x	1.15x
	97403	27.31±0.17	29.18±0.09	0.94x	25.41±0.08	1.15x	1.08x
	189810	56.19±0.10	58.66±0.38	0.96x	51.30±0.45	1.14x	1.10x
General	19880	4.28±0.04	4.04±0.04	1.06x	3.55±0.05	1.14x	1.21x
	97403	15.80±0.06	16.12±0.06	0.98x	12.95±0.07	1.24x	1.22x
	189810	31.52±0.02	32.56±0.09	0.97x	25.24±0.08	1.29x	1.25x
<i>Sheaf Diffusion</i>							
Diagonal	19880	2.96±0.08	2.60±0.03	1.14x	2.63±0.07	0.99x	1.12x
	97403	4.79±0.05	4.62±0.03	1.04x	4.01±0.05	1.15x	1.19x
	189810	8.79±0.03	8.81±0.02	1.00x	8.06±0.02	1.09x	1.09x
Bundle	19880	5.40±0.05	5.24±0.07	1.03x	4.99±0.07	1.05x	1.08x
	97403	22.76±0.10	24.03±0.17	0.95x	23.51±0.17	1.02x	0.97x
	189810	44.55±0.22	46.55±0.32	0.96x	47.23±0.63	0.99x	0.94x
General	19880	5.82±0.03	5.68±0.02	1.03x	3.68±0.04	1.54x	1.58x
	97403	23.24±0.11	24.13±0.16	0.96x	12.99±0.07	1.86x	1.79x
	189810	45.12±0.20	45.97±0.37	0.98x	25.10±0.10	1.83x	1.80x

Table 19: Backward Time (ms) inductive ablation results for stalk dimension $d = 8$. Performance comparison of the Explicit operator builder with varying graph-update strategies (Explicit original vs Explicit optimized) against the Implicit formulation across graph densities m . The best results are in **bold**.

Stalk Dimension $d = 8$							
Family	m	Explicit original Value	Explicit optimized		Implicit fully optimized		
			Value	Spd (vs Orig)	Value	Spd (vs Opt)	Spd (vs Orig)
<i>Sheaf Convolution</i>							
Diagonal	19880	2.98 $\pm$ 0.06	2.62$\pm$0.08	1.14x	2.82 $\pm$ 0.08	0.93x	1.06x
	97403	10.34$\pm$0.02	10.58 $\pm$ 0.03	0.98x	10.77 $\pm$ 0.04	0.98x	0.96x
	189810	19.64$\pm$0.07	20.33 $\pm$ 0.01	0.97x	20.01 $\pm$ 0.01	1.02x	0.98x
Bundle	19880	17.29 $\pm$ 0.17	18.07 $\pm$ 0.15	0.96x	10.60$\pm$0.08	1.71x	1.63x
	97403	95.75 $\pm$ 0.50	97.85 $\pm$ 0.19	0.98x	55.99$\pm$0.31	1.75x	1.71x
	189810	194.50 $\pm$ 1.08	195.30 $\pm$ 0.79	1.00x	113.87$\pm$0.64	1.72x	1.71x
General	19880	15.21 $\pm$ 0.09	15.63 $\pm$ 0.06	0.97x	6.05$\pm$0.05	2.58x	2.51x
	97403	74.20 $\pm$ 0.26	73.98 $\pm$ 0.25	1.00x	26.58$\pm$0.12	2.78x	2.79x
	189810	147.08 $\pm$ 0.88	148.22 $\pm$ 0.75	0.99x	54.38$\pm$0.20	2.73x	2.70x
<i>Sheaf Diffusion</i>							
Diagonal	19880	2.93 $\pm$ 0.09	2.65$\pm$0.07	1.11x	2.68 $\pm$ 0.07	0.99x	1.10x
	97403	8.90 $\pm$ 0.03	8.96 $\pm$ 0.05	0.99x	8.04$\pm$0.05	1.11x	1.11x
	189810	15.87 $\pm$ 0.05	16.38 $\pm$ 0.05	0.97x	14.55$\pm$0.01	1.13x	1.09x
Bundle	19880	12.09 $\pm$ 0.10	12.23 $\pm$ 0.10	0.99x	7.79$\pm$0.05	1.57x	1.55x
	97403	59.07 $\pm$ 0.65	61.13 $\pm$ 0.80	0.97x	41.31$\pm$0.47	1.48x	1.43x
	189810	124.27 $\pm$ 0.48	125.50 $\pm$ 1.02	0.99x	83.29$\pm$0.40	1.51x	1.49x
General	19880	12.88 $\pm$ 0.03	12.87 $\pm$ 0.09	1.00x	5.94$\pm$0.02	2.16x	2.17x
	97403	58.65 $\pm$ 0.61	59.13 $\pm$ 0.24	0.99x	26.34$\pm$0.14	2.24x	2.23x
	189810	117.54 $\pm$ 0.85	118.15 $\pm$ 1.12	0.99x	53.93$\pm$0.50	2.19x	2.18x

Table 20: Backward Time (ms) inductive ablation results for stalk dimension $d = 8$ and feature dimension $f = 32$. Performance comparison of the Explicit operator builder with varying graph-update strategies (Explicit original vs Explicit optimized) against the Implicit formulation across graph densities m . The best results are in **bold**.

Stalk Dimension $d = 8, f = 32$							
Family	m	Explicit original	Explicit optimized		Implicit fully optimized		
		Value	Value	Spd (vs Orig)	Value	Spd (vs Opt)	Spd (vs Orig)
<i>Sheaf Convolution</i>							
Diagonal	19880	3.84 $\pm$ 0.05	3.73$\pm$0.01	1.03x	4.16 $\pm$ 0.03	0.90x	0.92x
	97403	15.91$\pm$0.04	16.39 $\pm$ 0.06	0.97x	19.39 $\pm$ 0.07	0.85x	0.82x
	189810	30.40$\pm$0.11	31.66 $\pm$ 0.07	0.96x	36.82 $\pm$ 0.09	0.86x	0.83x
Bundle	19880	23.56 $\pm$ 0.27	24.39 $\pm$ 0.19	0.97x	11.97$\pm$0.10	2.04x	1.97x
	97403	126.82 $\pm$ 0.95	128.22 $\pm$ 0.33	0.99x	62.10$\pm$0.41	2.06x	2.04x
	189810	255.22 $\pm$ 1.06	257.52 $\pm$ 0.84	0.99x	130.88$\pm$0.48	1.97x	1.95x
General	19880	21.00 $\pm$ 0.12	21.46 $\pm$ 0.08	0.98x	6.87$\pm$0.06	3.12x	3.06x
	97403	102.46 $\pm$ 0.48	102.27 $\pm$ 0.28	1.00x	31.54$\pm$0.07	3.24x	3.25x
	189810	203.25 $\pm$ 0.26	204.39 $\pm$ 0.64	0.99x	64.54$\pm$0.11	3.17x	3.15x
<i>Sheaf Diffusion</i>							
Diagonal	19880	3.58 $\pm$ 0.03	3.36 $\pm$ 0.04	1.07x	3.21$\pm$0.05	1.05x	1.12x
	97403	14.60 $\pm$ 0.08	14.91 $\pm$ 0.02	0.98x	14.17$\pm$0.04	1.05x	1.03x
	189810	27.02 $\pm$ 0.12	28.03 $\pm$ 0.09	0.96x	26.56$\pm$0.11	1.06x	1.02x
Bundle	19880	18.25 $\pm$ 0.19	18.75 $\pm$ 0.20	0.97x	9.48$\pm$0.11	1.98x	1.93x
	97403	91.28 $\pm$ 0.50	91.99 $\pm$ 0.32	0.99x	47.73$\pm$0.49	1.93x	1.91x
	189810	186.63 $\pm$ 0.70	188.20 $\pm$ 0.82	0.99x	100.07$\pm$0.56	1.88x	1.87x
General	19880	18.87 $\pm$ 0.10	19.14 $\pm$ 0.09	0.99x	7.09$\pm$0.04	2.70x	2.66x
	97403	88.91 $\pm$ 0.72	88.93 $\pm$ 0.18	1.00x	31.89$\pm$0.03	2.79x	2.79x
	189810	176.34 $\pm$ 0.67	176.67 $\pm$ 1.28	1.00x	65.14$\pm$0.37	2.71x	2.71x

Table 21: Backward Time (ms) inductive ablation results for stalk dimension $d = 8$ and feature dimension $f = 64$. Performance comparison of the Explicit operator builder with varying graph-update strategies (Explicit original vs Explicit optimized) against the Implicit formulation across graph densities m . The best results are in **bold**.

Stalk Dimension $d = 8, f = 64$							
Family	m	Explicit original	Explicit optimized		Implicit fully optimized		
		Value	Value	Spd (vs Orig)	Value	Spd (vs Opt)	Spd (vs Orig)
<i>Sheaf Convolution</i>							
Diagonal	19880	6.74 $\pm$ 0.05	6.69$\pm$0.03	1.01x	7.96 $\pm$ 0.05	0.84x	0.85x
	97403	27.65$\pm$0.11	28.30 $\pm$ 0.11	0.98x	36.46 $\pm$ 0.08	0.78x	0.76x
	189810	53.63$\pm$0.21	54.49 $\pm$ 0.21	0.98x	69.34 $\pm$ 0.04	0.79x	0.77x
Bundle	19880	37.06 $\pm$ 0.37	38.41 $\pm$ 0.39	0.96x	16.57$\pm$0.22	2.32x	2.24x
	97403	190.70 $\pm$ 0.57	192.34 $\pm$ 0.59	0.99x	81.43$\pm$0.59	2.36x	2.34x
	189810	376.93 $\pm$ 1.30	380.52 $\pm$ 2.11	0.99x	166.45$\pm$0.95	2.29x	2.26x
General	19880	33.56 $\pm$ 0.17	34.57 $\pm$ 0.21	0.97x	10.17$\pm$0.07	3.40x	3.30x
	97403	160.95 $\pm$ 1.21	160.58 $\pm$ 0.95	1.00x	43.66$\pm$0.25	3.68x	3.69x
	189810	317.59 $\pm$ 2.40	318.22 $\pm$ 2.25	1.00x	88.58$\pm$0.70	3.59x	3.59x
<i>Sheaf Diffusion</i>							
Diagonal	19880	6.52 $\pm$ 0.10	6.41 $\pm$ 0.01	1.02x	6.20$\pm$0.02	1.03x	1.05x
	97403	26.39 $\pm$ 0.23	26.83 $\pm$ 0.15	0.98x	26.38$\pm$0.14	1.02x	1.00x
	189810	50.20 $\pm$ 0.19	50.83 $\pm$ 0.11	0.99x	49.91$\pm$0.22	1.02x	1.01x
Bundle	19880	31.17 $\pm$ 0.33	32.00 $\pm$ 0.34	0.97x	13.96$\pm$0.12	2.29x	2.23x
	97403	153.00 $\pm$ 0.91	154.10 $\pm$ 0.40	0.99x	66.21$\pm$0.24	2.33x	2.31x
	189810	306.60 $\pm$ 0.91	309.15 $\pm$ 1.53	0.99x	135.62$\pm$0.81	2.28x	2.26x
General	19880	31.26 $\pm$ 0.24	31.82 $\pm$ 0.14	0.98x	10.33$\pm$0.04	3.08x	3.02x
	97403	146.33 $\pm$ 0.90	146.49 $\pm$ 0.60	1.00x	44.42$\pm$0.29	3.30x	3.29x
	189810	287.85 $\pm$ 0.69	289.07 $\pm$ 0.35	1.00x	89.31$\pm$0.21	3.24x	3.22x

Table 22: Peak Memory (MB) inductive ablation results for stalk dimension $d = 2$. Performance comparison of the Explicit operator builder with varying graph-update strategies (Explicit original vs Explicit optimized) against the Implicit formulation across graph densities m . The best results are in **bold**.

Stalk Dimension $d = 2$							
Family	m	Explicit original	Explicit optimized		Implicit fully optimized		
		Value	Value	MemR (vs Orig)	Value	MemR (vs Opt)	MemR (vs Orig)
<i>Sheaf Convolution</i>							
Diagonal	19880	11.15 $\pm$ 0.08	11.10 $\pm$ 0.08	1.01x	10.15$\pm$0.08	1.09x	1.10x
	97403	54.55 $\pm$ 1.12	54.37 $\pm$ 0.42	1.00x	50.11$\pm$0.43	1.08x	1.09x
	189810	106.53 $\pm$ 0.86	105.75 $\pm$ 0.28	1.01x	97.77$\pm$0.09	1.08x	1.09x
Bundle	19880	12.59$\pm$0.12	12.60 $\pm$ 0.11	1.00x	13.52 $\pm$ 0.08	0.93x	0.93x
	97403	60.55 $\pm$ 0.20	60.40$\pm$0.14	1.00x	65.65 $\pm$ 0.22	0.92x	0.92x
	189810	117.73 $\pm$ 1.03	117.10$\pm$0.16	1.01x	128.54 $\pm$ 0.72	0.91x	0.92x
General	19880	12.74 $\pm$ 0.07	12.66$\pm$0.05	1.01x	13.25 $\pm$ 0.06	0.96x	0.96x
	97403	60.27 $\pm$ 0.52	60.24$\pm$0.22	1.00x	63.58 $\pm$ 0.27	0.95x	0.95x
	189810	117.42 $\pm$ 0.53	117.29$\pm$0.86	1.00x	124.32 $\pm$ 0.79	0.94x	0.94x
<i>Sheaf Diffusion</i>							
Diagonal	19880	11.08 $\pm$ 0.08	11.02 $\pm$ 0.08	1.01x	10.15$\pm$0.08	1.09x	1.09x
	97403	54.21 $\pm$ 0.47	54.19 $\pm$ 0.01	1.00x	49.62$\pm$0.00	1.09x	1.09x
	189810	106.18 $\pm$ 1.57	104.51 $\pm$ 0.75	1.02x	96.90$\pm$0.40	1.08x	1.10x
Bundle	19880	12.15$\pm$0.12	12.16 $\pm$ 0.11	1.00x	13.39 $\pm$ 0.08	0.91x	0.91x
	97403	60.09 $\pm$ 0.51	59.14$\pm$0.14	1.02x	65.28 $\pm$ 0.16	0.91x	0.92x
	189810	115.87 $\pm$ 0.80	115.49$\pm$0.58	1.00x	128.25 $\pm$ 0.33	0.90x	0.90x
General	19880	12.58 $\pm$ 0.07	12.50$\pm$0.05	1.01x	13.12 $\pm$ 0.06	0.95x	0.96x
	97403	59.80$\pm$0.48	59.98 $\pm$ 0.55	1.00x	63.85 $\pm$ 0.36	0.94x	0.94x
	189810	116.58 $\pm$ 0.44	115.89$\pm$0.46	1.01x	124.52 $\pm$ 0.25	0.93x	0.94x

Table 23: Peak Memory (MB) inductive ablation results for stalk dimension $d = 4$. Performance comparison of the Explicit operator builder with varying graph-update strategies (Explicit original vs Explicit optimized) against the Implicit formulation across graph densities m . The best results are in **bold**.

Stalk Dimension $d = 4$							
		Explicit original	Explicit optimized		Implicit fully optimized		
Family	m	Value	Value	MemR (vs Orig)	Value	MemR (vs Opt)	MemR (vs Orig)
<i>Sheaf Convolution</i>							
Diagonal	19880	22.31 $\pm$ 0.16	22.19 $\pm$ 0.18	1.01x	20.29$\pm$0.16	1.09x	1.10x
	97403	108.89 $\pm$ 0.72	108.59 $\pm$ 0.21	1.00x	99.60$\pm$0.32	1.09x	1.09x
	189810	209.39 $\pm$ 1.22	208.84 $\pm$ 0.45	1.00x	191.29$\pm$0.40	1.09x	1.09x
Bundle	19880	49.94 $\pm$ 0.66	49.61 $\pm$ 0.61	1.01x	39.13$\pm$0.44	1.27x	1.28x
	97403	232.74 $\pm$ 0.67	232.50 $\pm$ 0.61	1.00x	184.42$\pm$0.80	1.26x	1.26x
	189810	450.73 $\pm$ 1.54	451.66 $\pm$ 2.57	1.00x	356.75$\pm$1.75	1.27x	1.26x
General	19880	51.09 $\pm$ 0.34	51.24 $\pm$ 0.26	1.00x	38.26$\pm$0.38	1.34x	1.34x
	97403	235.15 $\pm$ 0.89	234.85 $\pm$ 0.58	1.00x	179.69$\pm$0.82	1.31x	1.31x
	189810	455.62 $\pm$ 0.38	454.39 $\pm$ 1.15	1.00x	348.45$\pm$1.17	1.30x	1.31x
<i>Sheaf Diffusion</i>							
Diagonal	19880	22.15 $\pm$ 0.16	22.04 $\pm$ 0.17	1.01x	20.29$\pm$0.16	1.09x	1.09x
	97403	107.51 $\pm$ 0.04	107.98 $\pm$ 0.45	1.00x	98.95$\pm$0.37	1.09x	1.09x
	189810	208.23 $\pm$ 0.99	207.51 $\pm$ 0.46	1.00x	191.24$\pm$0.35	1.09x	1.09x
Bundle	19880	48.58 $\pm$ 0.52	48.52 $\pm$ 0.49	1.00x	38.93$\pm$0.36	1.25x	1.25x
	97403	227.90 $\pm$ 1.41	228.81 $\pm$ 1.48	1.00x	184.12$\pm$1.14	1.24x	1.24x
	189810	442.70 $\pm$ 1.61	443.65 $\pm$ 2.34	1.00x	356.59$\pm$1.86	1.24x	1.24x
General	19880	51.65 $\pm$ 0.21	50.43 $\pm$ 0.20	1.02x	38.55$\pm$0.13	1.31x	1.34x
	97403	231.89 $\pm$ 0.31	231.68 $\pm$ 0.27	1.00x	179.68$\pm$0.88	1.29x	1.29x
	189810	449.83 $\pm$ 0.38	448.67 $\pm$ 1.04	1.00x	348.24$\pm$1.16	1.29x	1.29x

Table 24: Peak Memory (MB) inductive ablation results for stalk dimension $d = 8$. Performance comparison of the Explicit operator builder with varying graph-update strategies (Explicit original vs Explicit optimized) against the Implicit formulation across graph densities m . The best results are in **bold**.

Stalk Dimension $d = 8$							
		Explicit original	Explicit optimized		Implicit fully optimized		
Family	m	Value	Value	MemR (vs Orig)	Value	MemR (vs Opt)	MemR (vs Orig)
<i>Sheaf Convolution</i>							
Diagonal	19880	45.71 $\pm$ 0.68	44.95 $\pm$ 0.20	1.02x	41.16$\pm$0.17	1.09x	1.11x
	97403	216.34 $\pm$ 0.23	216.22 $\pm$ 1.23	1.00x	198.53$\pm$0.69	1.09x	1.09x
	189810	419.12 $\pm$ 0.93	418.01 $\pm$ 1.22	1.00x	382.88$\pm$0.83	1.09x	1.09x
Bundle	19880	196.33 $\pm$ 1.86	197.34 $\pm$ 2.00	0.99x	50.82$\pm$0.97	3.88x	3.86x
	97403	924.51 $\pm$ 4.32	926.58 $\pm$ 3.16	1.00x	241.81$\pm$0.79	3.83x	3.82x
	189810	1794.61 $\pm$ 4.75	1797.68 $\pm$ 8.02	1.00x	469.96$\pm$2.44	3.83x	3.82x
General	19880	201.04 $\pm$ 1.20	200.66 $\pm$ 0.22	1.00x	49.70$\pm$0.54	4.04x	4.05x
	97403	938.74 $\pm$ 3.02	934.27 $\pm$ 3.17	1.00x	237.41$\pm$1.07	3.94x	3.95x
	189810	1820.02 $\pm$ 2.86	1816.54 $\pm$ 4.56	1.00x	463.25$\pm$0.95	3.92x	3.93x
<i>Sheaf Diffusion</i>							
Diagonal	19880	44.78 $\pm$ 0.17	44.65 $\pm$ 0.19	1.00x	41.16$\pm$0.17	1.08x	1.09x
	97403	215.04 $\pm$ 0.52	214.49 $\pm$ 1.05	1.00x	198.23$\pm$0.83	1.08x	1.08x
	189810	415.81 $\pm$ 1.61	414.96 $\pm$ 1.09	1.00x	382.97$\pm$0.76	1.08x	1.09x
Bundle	19880	187.71 $\pm$ 0.76	187.41 $\pm$ 0.52	1.00x	50.10$\pm$0.08	3.74x	3.75x
	97403	903.34 $\pm$ 5.80	906.58 $\pm$ 4.13	1.00x	241.96$\pm$1.47	3.75x	3.73x
	189810	1763.21 $\pm$ 4.54	1766.72 $\pm$ 9.00	1.00x	470.60$\pm$2.24	3.75x	3.75x
General	19880	198.56 $\pm$ 1.08	198.19 $\pm$ 0.23	1.00x	50.03$\pm$0.40	3.96x	3.97x
	97403	927.41 $\pm$ 2.89	922.81 $\pm$ 2.88	1.00x	237.77$\pm$0.77	3.88x	3.90x
	189810	1796.66 $\pm$ 1.90	1793.27 $\pm$ 4.09	1.00x	463.45$\pm$1.25	3.87x	3.88x

Table 25: Peak Memory (MB) inductive ablation results for stalk dimension $d = 8$ and feature dimension $f = 32$. Performance comparison of the Explicit operator builder with varying graph-update strategies (Explicit original vs Explicit optimized) against the Implicit formulation across graph densities m . The best results are in **bold**.

Stalk Dimension $d = 8, f = 32$							
		Explicit original	Explicit optimized		Implicit fully optimized		
Family	m	Value	Value	MemR (vs Orig)	Value	MemR (vs Opt)	MemR (vs Orig)
<i>Sheaf Convolution</i>							
Diagonal	19880	85.31 $\pm$ 0.67	85.14 $\pm$ 0.81	1.00x	81.35$\pm$0.78	1.05x	1.05x
	97403	408.51 $\pm$ 0.94	406.91 $\pm$ 1.70	1.00x	388.73$\pm$1.57	1.05x	1.05x
	189810	790.88 $\pm$ 1.52	789.69 $\pm$ 2.23	1.00x	754.68$\pm$2.24	1.05x	1.05x
Bundle	19880	360.84 $\pm$ 2.74	361.09 $\pm$ 3.98	1.00x	88.97$\pm$1.19	4.06x	4.06x
	97403	1689.84 $\pm$ 9.18	1693.63 $\pm$ 6.30	1.00x	431.55$\pm$1.80	3.92x	3.92x
	189810	3282.63 $\pm$ 7.53	3289.33 $\pm$ 15.60	1.00x	841.09$\pm$3.99	3.91x	3.90x
General	19880	366.84 $\pm$ 2.45	364.25 $\pm$ 1.82	1.01x	88.73$\pm$0.13	4.11x	4.13x
	97403	1710.55 $\pm$ 8.19	1702.12 $\pm$ 5.25	1.00x	427.64$\pm$1.43	3.98x	4.00x
	189810	3315.04 $\pm$ 3.09	3306.51 $\pm$ 8.57	1.00x	834.36$\pm$2.37	3.96x	3.97x
<i>Sheaf Diffusion</i>							
Diagonal	19880	85.00 $\pm$ 0.67	84.84 $\pm$ 0.81	1.00x	81.35$\pm$0.78	1.04x	1.04x
	97403	406.98 $\pm$ 0.99	405.25 $\pm$ 1.60	1.00x	388.73$\pm$1.56	1.04x	1.05x
	189810	787.97 $\pm$ 2.00	786.61 $\pm$ 2.27	1.00x	754.53$\pm$2.15	1.04x	1.04x
Bundle	19880	343.15 $\pm$ 4.25	342.90 $\pm$ 3.24	1.00x	89.27$\pm$1.23	3.84x	3.84x
	97403	1663.67 $\pm$ 9.57	1667.63 $\pm$ 6.47	1.00x	431.67$\pm$2.03	3.86x	3.85x
	189810	3244.47 $\pm$ 8.09	3249.71 $\pm$ 15.23	1.00x	841.54$\pm$3.89	3.86x	3.86x
General	19880	364.50 $\pm$ 2.32	362.23 $\pm$ 1.45	1.01x	89.11$\pm$0.65	4.07x	4.09x
	97403	1698.96 $\pm$ 8.37	1690.36 $\pm$ 5.71	1.01x	428.00$\pm$2.04	3.95x	3.97x
	189810	3291.95 $\pm$ 2.98	3283.55 $\pm$ 8.39	1.00x	834.39$\pm$2.06	3.94x	3.95x

Table 26: Peak Memory (MB) inductive ablation results for stalk dimension $d = 8$ and feature dimension $f = 64$. Performance comparison of the Explicit operator builder with varying graph-update strategies (Explicit original vs Explicit optimized) against the Implicit formulation across graph densities m . The best results are in **bold**.

Stalk Dimension $d = 8, f = 64$							
		Explicit original	Explicit optimized		Implicit fully optimized		
Family	m	Value	Value	MemR (vs Orig)	Value	MemR (vs Opt)	MemR (vs Orig)
<i>Sheaf Convolution</i>							
Diagonal	19880	163.80 $\pm$ 1.73	163.36 $\pm$ 1.76	1.00x	159.76$\pm$1.65	1.02x	1.03x
	97403	790.95 $\pm$ 2.64	788.99 $\pm$ 1.91	1.00x	770.78$\pm$2.06	1.02x	1.03x
	189810	1533.19 $\pm$ 5.34	1531.16 $\pm$ 3.35	1.00x	1496.92$\pm$3.32	1.02x	1.02x
Bundle	19880	687.34 $\pm$ 6.61	688.59 $\pm$ 6.67	1.00x	167.95$\pm$1.51	4.10x	4.09x
	97403	3221.55 $\pm$ 17.87	3229.90 $\pm$ 11.70	1.00x	812.30$\pm$3.46	3.98x	3.97x
	189810	6261.05 $\pm$ 14.95	6270.85 $\pm$ 29.53	1.00x	1583.81$\pm$7.34	3.96x	3.95x
General	19880	697.78 $\pm$ 3.57	694.40 $\pm$ 2.64	1.00x	167.31$\pm$0.73	4.15x	4.17x
	97403	3252.40 $\pm$ 12.89	3237.16 $\pm$ 10.13	1.00x	807.80$\pm$2.59	4.01x	4.03x
	189810	6302.12 $\pm$ 6.08	6288.10 $\pm$ 17.03	1.00x	1576.70$\pm$5.08	3.99x	4.00x
<i>Sheaf Diffusion</i>							
Diagonal	19880	163.49 $\pm$ 1.73	163.05 $\pm$ 1.76	1.00x	159.56$\pm$1.74	1.02x	1.02x
	97403	789.72 $\pm$ 2.98	787.13 $\pm$ 2.21	1.00x	770.37$\pm$1.91	1.02x	1.03x
	189810	1531.06 $\pm$ 6.81	1528.93 $\pm$ 3.11	1.00x	1496.82$\pm$3.24	1.02x	1.02x
Bundle	19880	657.32 $\pm$ 5.85	657.76 $\pm$ 7.33	1.00x	167.91$\pm$1.61	3.92x	3.91x
	97403	3181.06 $\pm$ 17.43	3190.22 $\pm$ 10.36	1.00x	812.02$\pm$2.85	3.93x	3.92x
	189810	6208.41 $\pm$ 15.17	6218.95 $\pm$ 29.21	1.00x	1583.75$\pm$7.49	3.93x	3.92x
General	19880	695.51 $\pm$ 3.13	692.22 $\pm$ 2.74	1.00x	167.50$\pm$0.58	4.13x	4.15x
	97403	3240.94 $\pm$ 13.32	3225.15 $\pm$ 10.14	1.00x	808.92$\pm$1.55	3.99x	4.01x
	189810	6279.86 $\pm$ 7.24	6265.30 $\pm$ 16.86	1.00x	1577.01$\pm$3.46	3.97x	3.98x

Table 27: Jetson Orin benchmarks for **Inference** on dataset **Mutag**. We report the net (dynamic) power draw, energy per graph, throughput and peak memory footprint across stalk dimensions $d \in \{2, 4, 8\}$. The best results are in **bold**.

Model	d	Net Power (W)			Energy/Graph (mJ)			Throughput (graphs/s)			Peak Mem (MB)		
		Orig	Ours	Ratio	Orig	Ours	Ratio	Orig	Ours	Spd.	Orig	Ours	Mem.
Sheaf Convolution													
Diagonal	2	0.90±0.00	0.92±0.01	0.98x	2.33±0.00	2.20±0.02	1.06x	2170.91±3.63	2307.35±20.12	1.06x	17.62±0.01	17.56±0.00	1.00x
	4	0.99±0.01	1.00±0.00	0.99x	2.37±0.01	2.24±0.02	1.06x	2171.38±5.77	2298.25±16.42	1.06x	18.36±0.02	18.38±0.03	1.00x
	8	1.18±0.01	1.19±0.02	0.99x	2.47±0.02	2.32±0.01	1.06x	2158.19±16.86	2301.29±11.79	1.07x	20.41±0.07	20.43±0.01	1.00x
Bundle	2	0.89±0.00	0.90±0.01	0.99x	3.45±0.01	3.35±0.04	1.03x	1462.52±5.20	1510.26±17.48	1.03x	17.98±0.03	17.74±0.02	1.01x
	4	1.09±0.00	1.04±0.00	1.05x	3.59±0.01	3.41±0.03	1.05x	1460.21±4.23	1524.36±11.75	1.04x	22.32±0.18	19.95±0.03	1.12x
	8	1.82±0.01	1.51±0.00	1.20x	4.23±0.02	3.87±0.02	1.09x	1411.13±7.57	1464.90±9.25	1.04x	39.69±0.46	22.24±0.05	1.78x
General	2	1.08±0.01	1.11±0.00	0.97x	4.10±0.08	3.80±0.04	1.08x	1275.08±23.72	1383.70±14.53	1.09x	17.93±0.01	17.67±0.01	1.01x
	4	1.29±0.00	1.27±0.00	1.02x	5.63±0.09	5.37±0.11	1.05x	967.01±15.24	1009.34±20.99	1.04x	22.27±0.06	19.94±0.03	1.12x
	8	1.78±0.01	1.62±0.01	1.09x	8.78±0.24	8.19±0.09	1.07x	675.56±18.95	705.49±8.32	1.04x	39.40±0.38	22.09±0.08	1.78x
Sheaf Diffusion													
Diagonal	2	0.84±0.00	0.97±0.01	0.87x	2.97±0.01	2.56±0.06	1.16x	1682.36±3.90	1999.24±46.43	1.19x	17.61±0.00	17.58±0.01	1.00x
	4	0.92±0.01	0.97±0.01	0.95x	3.04±0.02	2.49±0.00	1.22x	1670.95±14.72	2054.20±4.24	1.23x	18.33±0.05	18.23±0.04	1.01x
	8	1.08±0.01	1.15±0.01	0.93x	3.14±0.03	2.47±0.02	1.27x	1666.89±18.64	2149.58±11.55	1.29x	20.26±0.01	20.12±0.01	1.01x
Bundle	2	0.95±0.00	0.89±0.01	1.07x	4.39±0.01	3.53±0.06	1.24x	1164.09±1.92	1431.27±23.62	1.23x	17.69±0.01	17.70±0.04	1.00x
	4	1.07±0.01	1.02±0.00	1.05x	4.52±0.03	3.61±0.03	1.25x	1156.18±9.41	1436.17±12.55	1.24x	21.42±0.02	19.81±0.02	1.08x
	8	1.55±0.02	1.45±0.00	1.07x	5.00±0.10	4.00±0.00	1.25x	1141.56±27.18	1401.73±1.66	1.23x	33.09±0.03	21.86±0.03	1.51x
General	2	1.19±0.00	1.08±0.01	1.10x	5.67±0.02	4.21±0.01	1.35x	942.80±4.23	1244.07±0.20	1.32x	17.88±0.05	17.62±0.00	1.01x
	4	1.33±0.02	1.24±0.01	1.07x	7.24±0.01	5.77±0.18	1.25x	757.21±3.04	935.49±28.53	1.24x	21.89±0.05	19.82±0.09	1.10x
	8	1.64±0.01	1.56±0.01	1.05x	9.48±0.21	7.93±0.15	1.19x	611.07±13.51	720.37±13.11	1.18x	38.42±0.05	21.94±0.07	1.75x

Table 28: Jetson Orin benchmarks for **Inference** on dataset **Ptc**. We report the net (dynamic) power draw, energy per graph, throughput and peak memory footprint across stalk dimensions $d \in \{2, 4, 8\}$. The best results are in **bold**.

Model	d	Net Power (W)			Energy/Graph (mJ)			Throughput (graphs/s)			Peak Mem (MB)		
		Orig	Ours	Ratio	Orig	Ours	Ratio	Orig	Ours	Spd.	Orig	Ours	Mem.
Sheaf Convolution													
Diagonal	2	0.95±0.01	0.97±0.00	0.98x	2.32±0.01	2.18±0.01	1.06x	2199.95±5.21	2345.71±7.47	1.07x	18.06±0.04	18.21±0.05	0.99x
	4	1.04±0.03	1.07±0.01	0.97x	2.36±0.01	2.23±0.01	1.06x	2199.66±13.91	2343.70±10.02	1.07x	19.80±0.16	19.82±0.07	1.00x
	8	1.33±0.04	1.33±0.01	1.00x	2.56±0.02	2.38±0.02	1.07x	2144.31±33.43	2305.07±15.20	1.07x	22.80±0.06	23.55±0.09	0.97x
Bundle	2	0.94±0.03	0.96±0.00	0.98x	3.37±0.02	3.24±0.01	1.04x	1511.41±3.45	1579.58±3.86	1.05x	19.05±0.05	18.63±0.09	1.02x
	4	1.22±0.02	1.15±0.00	1.06x	3.56±0.02	3.39±0.01	1.05x	1508.21±5.84	1565.76±5.46	1.04x	26.56±0.11	22.37±0.32	1.19x
	8	2.14±0.05	1.73±0.01	1.24x	4.59±0.05	4.15±0.03	1.11x	1369.70±25.10	1417.68±11.01	1.04x	54.04±1.59	26.59±0.41	2.03x
General	2	1.19±0.02	1.23±0.01	0.97x	4.30±0.03	4.07±0.03	1.06x	1241.70±11.13	1323.61±8.53	1.07x	18.93±0.03	18.61±0.13	1.02x
	4	1.40±0.03	1.38±0.00	1.01x	6.55±0.01	6.16±0.06	1.06x	847.98±4.15	897.90±8.49	1.06x	26.50±0.34	22.02±0.30	1.20x
	8	1.87±0.04	1.70±0.00	1.09x	10.83±0.16	10.18±0.18	1.06x	556.25±7.97	575.38±9.58	1.03x	52.40±1.36	25.99±0.10	2.02x
Sheaf Diffusion													
Diagonal	2	0.91±0.00	1.02±0.01	0.89x	2.94±0.00	2.49±0.03	1.18x	1721.08±1.68	2080.51±20.59	1.21x	17.98±0.07	18.01±0.08	1.00x
	4	0.99±0.02	1.05±0.01	0.94x	3.09±0.02	2.51±0.00	1.23x	1666.34±14.98	2073.25±5.37	1.24x	19.67±0.15	19.69±0.08	1.00x
	8	1.22±0.01	1.28±0.01	0.95x	3.28±0.06	2.63±0.03	1.24x	1640.13±25.50	2063.88±24.50	1.26x	22.68±0.23	22.64±0.25	1.00x
Bundle	2	1.03±0.00	0.96±0.00	1.08x	4.31±0.02	3.41±0.03	1.26x	1204.40±4.38	1497.78±13.46	1.24x	18.67±0.19	18.57±0.07	1.01x
	4	1.20±0.03	1.13±0.00	1.06x	4.53±0.04	3.54±0.03	1.28x	1182.55±13.95	1492.87±14.07	1.26x	23.56±0.54	22.11±0.05	1.07x
	8	1.80±0.03	1.68±0.01	1.07x	5.21±0.03	4.23±0.05	1.23x	1142.18±10.69	1380.06±17.28	1.21x	41.80±0.41	25.82±0.23	1.62x
General	2	1.29±0.01	1.19±0.01	1.08x	6.24±0.02	4.46±0.02	1.40x	873.29±3.38	1198.43±8.60	1.37x	18.91±0.08	18.44±0.11	1.03x
	4	1.42±0.02	1.36±0.00	1.04x	8.38±0.11	6.62±0.09	1.27x	665.21±10.59	832.79±10.91	1.25x	25.64±0.07	22.17±0.47	1.16x
	8	1.76±0.02	1.65±0.00	1.06x	11.59±0.12	9.80±0.24	1.18x	509.82±6.56	592.97±14.91	1.16x	49.94±2.31	25.61±0.18	1.95x

Table 29: Jetson Orin benchmarks for **Inference** on dataset **Proteins**. We report the net (dynamic) power draw, energy per graph, throughput and peak memory footprint across stalk dimensions $d \in \{2, 4, 8\}$. The best results are in **bold**.

Model	d	Net Power (W)			Energy/Graph (mJ)			Throughput (graphs/s)			Peak Mem (MB)		
		Orig	Ours	Ratio	Orig	Ours	Ratio	Orig	Ours	Spd.	Orig	Ours	Mem.
Sheaf Convolution													
Diagonal	2	1.11±0.01	1.15±0.00	0.97x	2.50±0.08	2.38±0.07	1.05x	2134.33±46.75	2250.32±40.98	1.05x	22.34±0.56	22.15±0.43	1.01x
	4	1.41±0.01	1.43±0.01	0.99x	2.66±0.08	2.50±0.07	1.06x	2119.31±43.77	2256.96±40.63	1.06x	27.10±0.88	26.76±1.01	1.01x
	8	2.01±0.02	2.04±0.02	0.99x	2.99±0.09	2.81±0.07	1.06x	2083.78±49.43	2227.77±45.73	1.07x	38.99±1.89	36.58±2.61	1.07x
Bundle	2	1.18±0.01	1.19±0.01	0.99x	3.70±0.13	3.58±0.11	1.03x	1458.97±36.45	1508.88±33.29	1.03x	25.25±1.03	23.54±0.92	1.07x
	4	1.88±0.01	1.81±0.01	1.04x	4.30±0.10	4.01±0.12	1.07x	1417.85±23.83	1501.71±36.21	1.06x	48.22±4.48	37.60±1.28	1.28x
	8	3.44±0.02	2.65±0.01	1.30x	7.52±0.14	7.00±0.17	1.07x	1017.80±15.69	981.59±18.22	0.96x	131.59±5.31	45.86±1.88	2.87x
General	2	1.43±0.01	1.47±0.01	0.97x	5.27±0.18	4.99±0.11	1.06x	1071.70±29.44	1139.31±12.62	1.06x	24.51±0.95	23.94±0.65	1.02x
	4	1.72±0.01	1.70±0.01	1.01x	9.15±0.36	8.69±0.50	1.05x	648.82±20.57	681.50±34.45	1.05x	48.57±3.44	35.68±0.52	1.36x
	8	2.24±0.03	1.99±0.01	1.13x	19.28±0.07	18.16±0.41	1.06x	334.66±3.18	341.75±4.59	1.02x	137.39±13.27	48.35±3.54	2.84x
Sheaf Diffusion													
Diagonal	2	1.04±0.01	1.19±0.01	0.88x	3.25±0.12	2.66±0.11	1.22x	1619.33±43.44	2035.83±67.71	1.26x	21.93±0.94	21.22±0.04	1.03x
	4	1.27±0.01	1.35±0.01	0.94x	3.41±0.12	2.75±0.10	1.24x	1609.04±41.76	2023.16±56.55	1.26x	27.23±0.31	26.60±0.14	1.02x
	8	1.73±0.02	1.88±0.03	0.92x	3.75±0.12	3.06±0.13	1.23x	1588.19±45.54	1994.80±80.23	1.26x	38.72±1.00	36.26±1.45	1.07x
Bundle	2	1.41±0.01	1.17±0.01	1.21x	4.92±0.15	3.75±0.09	1.31x	1145.12±25.28	1437.61±19.92	1.26x	23.28±0.39	23.91±0.55	0.97x
	4	1.85±0.01	1.73±0.01	1.07x	5.54±0.16	4.15±0.12	1.33x	1094.97±23.98	1431.32±29.38	1.31x	41.64±0.99	36.87±1.26	1.13x
	8	2.90±0.01	2.48±0.01	1.17x	8.26±0.13	6.92±0.16	1.19x	861.09±8.57	967.49±15.74	1.12x	104.99±11.30	44.31±2.29	2.37x
General	2	1.57±0.01	1.43±0.01	1.10x	9.17±0.26	5.47±0.08	1.68x	631.49±11.83	1031.86±6.03	1.63x	24.81±0.69	24.02±0.19	1.03x
	4	1.73±0.01	1.68±0.01	1.03x	12.92±0.32	9.04±0.11	1.43x	460.75±6.61	652.09±2.29	1.42x	42.99±1.49	35.37±0.26	1.22x
	8	2.16±0.02	1.94±0.01	1.11x	19.17±0.35	16.75±0.57	1.14x	332.28±2.54	367.62±9.53	1.11x	134.32±8.62	43.65±0.81	3.08x

Table 30: Jetson Orin benchmarks for **Train** on dataset **Mutag**. We report the net (dynamic) power draw, energy per graph, throughput and peak memory footprint across stalk dimensions $d \in \{2, 4, 8\}$. The best results are in **bold**.

Model	d	Net Power (W)			Energy/Graph (mJ)			Throughput (graphs/s)			Peak Mem (MB)		
		Orig	Ours	Ratio	Orig	Ours	Ratio	Orig	Ours	Spd.	Orig	Ours	Mem.
Sheaf Convolution													
Diagonal	2	0.94±0.00	0.93±0.02	1.01x	5.05±0.02	4.89±0.13	1.03x	1009.07±4.72	1040.38±27.46	1.03x	19.92±0.02	20.14±0.03	0.99x
	4	1.06±0.00	1.03±0.01	1.03x	5.19±0.05	5.04±0.07	1.03x	1005.07±9.95	1028.21±14.66	1.02x	22.64±0.12	23.02±0.08	0.98x
	8	1.31±0.00	1.27±0.00	1.03x	5.48±0.05	5.43±0.06	1.01x	997.95±8.74	999.29±11.01	1.00x	29.22±0.39	29.53±0.72	0.99x
Bundle	2	1.04±0.00	1.03±0.00	1.01x	7.26±0.02	7.24±0.14	1.00x	714.94±2.42	716.63±13.74	1.00x	22.05±0.04	21.22±0.12	1.04x
	4	1.34±0.01	1.22±0.02	1.10x	7.76±0.04	7.67±0.09	1.01x	708.22±3.50	701.20±9.42	0.99x	34.58±0.11	28.04±0.66	1.23x
	8	2.34±0.01	1.81±0.01	1.29x	10.37±0.05	8.76±0.13	1.18x	625.76±3.58	680.75±11.33	1.09x	81.35±1.47	38.16±0.07	2.13x
General	2	1.08±0.00	1.09±0.01	0.99x	7.95±0.02	7.28±0.15	1.09x	658.11±1.17	720.72±13.52	1.10x	21.09±0.01	20.18±0.02	1.04x
	4	1.33±0.00	1.31±0.01	1.01x	9.86±0.09	9.26±0.14	1.06x	556.36±5.27	590.48±9.69	1.06x	32.96±0.32	25.54±0.60	1.29x
	8	2.16±0.00	1.81±0.02	1.20x	14.42±0.05	12.43±0.10	1.16x	437.90±1.64	479.61±4.91	1.10x	77.12±1.26	31.61±0.09	2.44x
Sheaf Diffusion													
Diagonal	2	0.90±0.01	0.94±0.03	0.95x	6.21±0.06	5.10±0.21	1.22x	813.43±7.44	1000.77±36.27	1.23x	19.92±0.07	19.92±0.03	1.00x
	4	0.99±0.01	1.01±0.00	0.98x	6.36±0.02	5.34±0.34	1.19x	807.82±3.74	968.72±64.17	1.20x	22.39±0.09	22.51±0.10	0.99x
	8	1.17±0.00	1.20±0.01	0.97x	6.65±0.01	5.93±0.11	1.12x	800.63±1.17	903.85±18.66	1.13x	28.75±0.37	28.02±0.65	1.03x
Bundle	2	1.04±0.00	1.03±0.01	1.01x	8.79±0.06	7.13±0.18	1.23x	590.55±3.84	726.73±18.60	1.23x	21.29±0.04	21.25±0.05	1.00x
	4	1.20±0.00	1.21±0.00	0.99x	9.09±0.09	7.65±0.30	1.19x	588.97±5.23	701.66±28.59	1.19x	31.44±0.31	27.03±0.04	1.16x
	8	1.79±0.01	1.65±0.01	1.09x	10.20±0.04	8.73±0.12	1.17x	582.54±2.74	664.70±10.50	1.14x	65.16±1.07	37.03±0.85	1.76x
General	2	1.27±0.00	1.07±0.02	1.18x	10.30±0.03	7.88±0.21	1.31x	526.35±1.22	663.87±19.02	1.26x	21.05±0.02	20.40±0.01	1.03x
	4	1.44±0.00	1.30±0.01	1.11x	12.27±0.00	9.87±0.26	1.24x	456.43±0.25	552.56±15.72	1.21x	32.25±0.26	26.65±0.10	1.21x
	8	1.97±0.01	1.78±0.00	1.11x	15.36±0.01	12.81±0.09	1.20x	398.79±0.43	463.11±3.67	1.16x	75.57±2.22	32.59±0.08	2.32x

Table 31: Jetson Orin benchmarks for **Train** on dataset **Ptc**. We report the net (dynamic) power draw, energy per graph, throughput and peak memory footprint across stalk dimensions $d \in \{2, 4, 8\}$. The best results are in **bold**.

Model	d	Net Power (W)			Energy/Graph (mJ)			Throughput (graphs/s)			Peak Mem (MB)		
		Orig	Ours	Ratio	Orig	Ours	Ratio	Orig	Ours	Spd.	Orig	Ours	Mem.
Sheaf Convolution													
Diagonal	2	1.00±0.01	0.98±0.02	1.02x	5.04±0.03	4.95±0.03	1.02x	1021.94±2.77	1036.94±3.36	1.01x	21.65±0.13	21.86±0.11	0.99x
	4	1.14±0.01	1.12±0.01	1.02x	5.22±0.06	5.15±0.04	1.01x	1015.41±10.98	1024.92±5.04	1.01x	26.49±0.30	26.19±0.49	1.01x
	8	1.51±0.01	1.44±0.01	1.05x	5.58±0.06	5.54±0.06	1.01x	1014.66±9.68	1010.47±10.88	1.00x	36.52±0.44	37.22±0.88	0.98x
Bundle	2	1.14±0.01	1.11±0.02	1.03x	7.22±0.07	7.31±0.05	0.99x	733.40±6.99	719.96±4.04	0.98x	25.74±0.31	23.42±0.27	1.10x
	4	1.56±0.02	1.40±0.02	1.12x	7.82±0.06	7.66±0.01	1.02x	729.94±7.16	724.45±1.79	0.99x	46.17±0.68	34.88±0.25	1.32x
	8	2.71±0.01	2.15±0.01	1.26x	11.87±0.10	9.31±0.01	1.28x	578.35±3.95	677.02±0.44	1.17x	115.29±4.81	49.50±0.60	2.33x
General	2	1.19±0.01	1.20±0.02	0.99x	8.21±0.08	7.70±0.05	1.07x	650.72±5.74	695.66±3.16	1.07x	24.06±0.16	22.10±0.20	1.09x
	4	1.49±0.00	1.47±0.01	1.01x	10.77±0.02	10.28±0.10	1.05x	524.27±0.31	547.52±4.08	1.04x	41.84±0.44	30.45±0.64	1.37x
	8	2.32±0.02	1.96±0.01	1.18x	17.81±0.16	15.22±0.21	1.17x	363.49±3.89	401.68±5.33	1.11x	112.09±2.71	41.31±0.90	2.71x
Sheaf Diffusion													
Diagonal	2	0.95±0.00	0.98±0.02	0.98x	5.99±0.05	5.43±0.07	1.10x	852.67±6.57	944.71±14.92	1.11x	21.43±0.34	21.54±0.09	0.99x
	4	1.07±0.01	1.10±0.02	0.98x	6.15±0.07	5.51±0.05	1.12x	849.56±10.43	952.47±10.57	1.12x	25.91±0.13	25.44±0.10	1.02x
	8	1.34±0.01	1.39±0.02	0.96x	6.52±0.10	5.87±0.05	1.11x	842.79±12.67	943.65±7.19	1.12x	35.51±0.66	36.62±0.66	0.97x
Bundle	2	1.14±0.00	1.10±0.02	1.04x	8.54±0.12	7.51±0.07	1.14x	620.21±9.10	699.96±5.55	1.13x	24.29±0.29	23.74±0.44	1.02x
	4	1.37±0.00	1.38±0.02	1.00x	8.98±0.09	7.84±0.11	1.14x	615.62±5.92	705.47±8.40	1.15x	38.42±1.03	34.33±1.12	1.12x
	8	2.17±0.01	1.99±0.03	1.09x	10.41±0.12	8.97±0.11	1.16x	607.59±7.07	685.41±8.77	1.13x	92.66±2.08	48.41±0.55	1.91x
General	2	1.44±0.00	1.17±0.02	1.23x	10.82±0.12	8.36±0.08	1.29x	517.25±5.75	637.49±4.96	1.23x	24.04±0.42	22.77±0.13	1.06x
	4	1.64±0.01	1.47±0.01	1.11x	13.32±0.02	10.66±0.17	1.25x	434.57±0.49	527.43±9.49	1.21x	41.53±0.76	31.05±0.71	1.34x
	8	2.17±0.00	1.95±0.01	1.11x	18.43±0.26	15.40±0.13	1.20x	342.96±4.78	396.68±2.65	1.16x	105.32±1.53	43.44±2.17	2.42x

Table 32: Jetson Orin benchmarks for **Train** on dataset **Proteins**. We report the net (dynamic) power draw, energy per graph, throughput and peak memory footprint across stalk dimensions $d \in \{2, 4, 8\}$. The best results are in **bold**.

Model	d	Net Power (W)			Energy/Graph (mJ)			Throughput (graphs/s)			Peak Mem (MB)		
		Orig	Ours	Ratio	Orig	Ours	Ratio	Orig	Ours	Spd.	Orig	Ours	Mem.
Sheaf Convolution													
Diagonal	2	1.19±0.02	1.19±0.02	1.00x	5.53±0.25	5.41±0.15	1.02x	978.08±38.11	993.45±20.61	1.02x	33.07±2.35	31.99±1.92	1.03x
	4	1.54±0.02	1.54±0.01	1.00x	5.95±0.22	5.78±0.19	1.03x	968.01±29.25	991.18±25.70	1.02x	49.70±1.59	48.07±0.20	1.03x
	8	2.31±0.04	2.27±0.03	1.01x	6.81±0.25	6.61±0.18	1.03x	959.29±33.87	977.48±23.43	1.02x	77.86±6.03	76.98±0.03	1.01x
Bundle	2	1.63±0.02	1.62±0.01	1.01x	8.67±0.27	8.55±0.25	1.01x	674.72±17.34	679.07±14.14	1.01x	47.64±2.23	39.99±0.51	1.19x
	4	2.42±0.01	2.35±0.02	1.03x	10.98±0.26	10.01±0.23	1.10x	604.90±9.24	652.85±11.98	1.08x	110.38±3.09	80.71±5.41	1.37x
	8	3.54±0.00	2.92±0.00	1.21x	23.92±0.34	18.15±0.33	1.32x	324.28±2.50	391.56±4.11	1.21x	346.01±14.00	122.70±4.01	2.82x
General	2	1.46±0.01	1.52±0.00	0.96x	9.88±0.36	9.04±0.20	1.09x	575.14±16.98	631.78±8.96	1.10x	42.24±2.24	34.68±1.09	1.22x
	4	2.02±0.00	2.02±0.01	1.00x	14.48±0.26	13.49±0.33	1.07x	430.66±4.22	459.88±8.36	1.07x	97.04±5.21	62.90±4.49	1.54x
	8	2.71±0.01	2.29±0.01	1.18x	34.19±0.58	26.68±0.57	1.28x	202.69±1.61	242.90±3.24	1.20x	343.13±18.34	96.21±5.99	3.57x
Sheaf Diffusion													
Diagonal	2	1.10±0.01	1.16±0.03	0.95x	6.31±0.27	5.52±0.23	1.14x	843.35±29.30	968.50±32.17	1.15x	30.83±0.26	32.47±2.58	0.95x
	4	1.39±0.02	1.48±0.01	0.94x	6.67±0.26	5.87±0.21	1.14x	841.50±28.20	964.69±27.10	1.15x	47.70±3.21	43.03±1.32	1.11x
	8	1.96±0.03	2.11±0.05	0.93x	7.43±0.28	6.64±0.26	1.12x	830.68±29.61	948.92±36.17	1.14x	75.24±1.93	71.32±1.37	1.05x
Bundle	2	1.65±0.03	1.61±0.02	1.02x	9.91±0.36	8.48±0.22	1.17x	591.97±19.59	683.87±12.66	1.16x	41.78±0.68	41.94±1.97	1.00x
	4	2.22±0.02	2.28±0.02	0.97x	11.31±0.36	9.82±0.23	1.15x	569.06±14.99	658.62±12.33	1.16x	95.95±3.17	75.81±1.87	1.27x
	8	3.42±0.02	2.88±0.01	1.19x	17.87±0.34	15.57±0.30	1.15x	427.03±6.37	453.46±5.86	1.06x	272.55±25.82	121.13±3.68	2.25x
General	2	1.77±0.01	1.49±0.02	1.19x	15.85±0.41	9.80±0.27	1.62x	377.93±6.99	579.32±10.01	1.53x	41.33±2.05	34.79±1.94	1.19x
	4	2.01±0.01	2.00±0.01	1.01x	20.76±0.51	14.11±0.34	1.47x	300.10±4.73	438.35±7.65	1.46x	100.22±6.33	65.86±4.24	1.52x
	8	2.63±0.00	2.29±0.01	1.15x	32.14±0.67	25.89±0.48	1.24x	212.99±2.81	250.06±2.27	1.17x	294.77±7.80	97.67±7.79	3.02x

Table 33: Hyperparameter search space for Node Classification Adjacency sweeps.

Hyperparameter	Cora, Citeseer, Pubmed	Cornell, Texas, Wisconsin	Film
Common Parameters			
Sheaf dimension (d)	{2, 3, 4}		
Add low-pass filter	{0, 1}		
Add high-pass filter	{0, 1}		
Weight decay	Log-Uniform(-9.2, -6.8)		
Sheaf decay	Log-Uniform(-11.0, -6.8)		
Input dropout	{0.0, 0.1, . . . , 0.9}		
Dropout	{0.0, 0.1, . . . , 0.9}		
Epochs	1000		
Folds	10		
Dataset-Specific Parameters			
Layers	{2, 3, 4, 5}	{2, 3, 4, 5, 6}	{2, 3, 4, 5, 6}
Hidden channels	{32, 64, 128}	{8, 16, 32}	{16, 32, 64}
Learning rate	{0.001, 0.005, 0.01}	0.02	{0.005, 0.01, 0.02}
Early stopping epochs	200	200	100
Second linear layer	{False, True}	{False, True}	{False, True}

- dropout: 0
- add_lp: True
- add_hp: False
- second_linear: True
- left_weights: True
- right_weights: True
- use_act: True
- normalized: True
- edge_weights: True

Pubmed

Diagonal

- d: 3
- dropout: 0.4
- layers: 2
- add_hp: 1
- add_lp: 0
- edge_weights: true
- input_dropout: 0.4
- left_weights: true
- lr: 0.005
- normalized: true
- hidden_channels: 32
- right_weights: true
- weight_decay: 0.00018836
- sheaf_decay: 0.00021287
- second_linear: false
- use_act: true

Bundle

- d: 3
- dropout: 0.5
- layers: 3
- add_hp: 0
- add_lp: 1
- edge_weights: true
- input_dropout: 0.3
- left_weights: true
- lr: 0.01
- normalized: true
- hidden_channels: 32
- right_weights: true
- weight_decay: 0.00026179
- sheaf_decay: 0.00057764
- second_linear: false
- use_act: true

General

- d: 2
- dropout: 0.8
- layers: 2
- add_hp: 0
- add_lp: 0
- edge_weights: true
- input_dropout: 0.3
- left_weights: true
- lr: 0.01
- normalized: true
- hidden_channels: 128
- right_weights: true
- weight_decay: 0.00039514
- sheaf_decay: 0.00050855
- second_linear: false
- use_act: true

Wisconsin

Diagonal

- d: 4
- dropout: 0.7
- layers: 4
- add_hp: 1
- add_lp: 0
- edge_weights: true
- input_dropout: 0
- left_weights: true
- lr: 0.02
- normalized: true
- hidden_channels: 8
- right_weights: true
- weight_decay: 0.0005688
- sheaf_decay: 0.000467
- second_linear: false
- use_act: true

Bundle

- d: 2
- dropout: 0.1
- layers: 4
- add_hp: 1
- add_lp: 1
- edge_weights: true
- input_dropout: 0.4
- left_weights: true
- lr: 0.02
- normalized: true
- hidden_channels: 8

- right_weights: true
- weight_decay: 0.00077825
- sheaf_decay: 0.00002084
- second_linear: false
- use_act: true

General

- d: 3
- dropout: 0.5
- layers: 4
- add_hp: 1
- add_lp: 0
- edge_weights: true
- input_dropout: 0
- left_weights: true
- lr: 0.02
- normalized: true
- hidden_channels: 8
- right_weights: true
- weight_decay: 0.00037602
- sheaf_decay: 0.00017423
- second_linear: false
- use_act: true

Texas

Diagonal

- d: 2
- dropout: 0.5
- layers: 3
- add_hp: 1
- add_lp: 0
- edge_weights: true
- input_dropout: 0.5
- left_weights: true
- lr: 0.02
- normalized: true
- hidden_channels: 16
- right_weights: true
- weight_decay: 0.00031919
- sheaf_decay: 0.00014633
- second_linear: false
- use_act: true

Bundle

- d: 2
- dropout: 0.5
- layers: 4
- add_hp: 1
- add_lp: 1
- edge_weights: true
- input_dropout: 0.2
- left_weights: true
- lr: 0.02
- normalized: true
- hidden_channels: 32
- right_weights: true
- weight_decay: 0.00029942
- sheaf_decay: 0.00010752
- second_linear: false
- use_act: true

General

- d: 3
- dropout: 0.8
- layers: 2
- add_hp: 1
- add_lp: 0
- edge_weights: true
- input_dropout: 0.1
- left_weights: true
- lr: 0.02
- normalized: true
- hidden_channels: 32
- right_weights: true
- weight_decay: 0.00045952
- sheaf_decay: 0.000043569
- second_linear: false
- use_act: true

Cora

Diagonal

- d: 2
- dropout: 0.8
- layers: 4
- add_hp: 1
- add_lp: 0
- edge_weights: true
- input_dropout: 0.8
- left_weights: true
- lr: 0.005
- normalized: true
- hidden_channels: 64

- right_weights: true
- weight_decay: 0.00020941
- sheaf_decay: 0.000077872
- second_linear: false
- use_act: true

Bundle

- d: 2
- dropout: 0.8
- layers: 2
- add_hp: 1
- add_lp: 0
- edge_weights: true
- input_dropout: 0.8
- left_weights: true
- lr: 0.01
- normalized: true
- hidden_channels: 64
- right_weights: true
- weight_decay: 0.00012531
- sheaf_decay: 0.000080333
- second_linear: false
- use_act: true

General

- d: 3
- dropout: 0.8
- layers: 5
- add_hp: 0
- add_lp: 1
- edge_weights: true
- input_dropout: 0.8
- left_weights: true
- lr: 0.005
- normalized: true
- hidden_channels: 32
- right_weights: true
- weight_decay: 0.00044352
- sheaf_decay: 0.00075317
- second_linear: false
- use_act: true

Cornell

Diagonal

- d: 4
- dropout: 0.9
- layers: 2
- add_hp: 1
- add_lp: 1
- edge_weights: true

- input_dropout: 0
- left_weights: true
- lr: 0.02
- normalized: true
- hidden_channels: 32
- right_weights: true
- weight_decay: 0.00037316
- sheaf_decay: 0.00002094
- second_linear: false
- use_act: true

Bundle

- d: 2
- dropout: 0.8
- layers: 5
- add_hp: 1
- add_lp: 1
- edge_weights: true
- input_dropout: 0
- left_weights: true
- lr: 0.02
- normalized: true
- hidden_channels: 16
- right_weights: true
- weight_decay: 0.00024439
- sheaf_decay: 0.00035285
- second_linear: false
- use_act: true

General

- d: 4
- dropout: 0.9
- layers: 2
- add_hp: 1
- add_lp: 1
- edge_weights: true
- input_dropout: 0.1
- left_weights: true
- lr: 0.02
- normalized: true
- hidden_channels: 16
- right_weights: true
- weight_decay: 0.00019184
- sheaf_decay: 0.000096788
- second_linear: false
- use_act: true

Citeseer

Diagonal

- d: 2
- dropout: 0.7
- layers: 4
- add_hp: 0
- add_lp: 1
- edge_weights: true
- input_dropout: 0.9
- left_weights: true
- lr: 0.01
- normalized: true
- hidden_channels: 128
- right_weights: true
- weight_decay: 0.00015503
- sheaf_decay: 0.000042988
- second_linear: false
- use_act: true

Bundle

- d: 2
- dropout: 0.8
- layers: 3
- add_hp: 0
- add_lp: 0
- edge_weights: true
- input_dropout: 0.8
- left_weights: true
- lr: 0.01
- normalized: true
- hidden_channels: 64
- right_weights: true
- weight_decay: 0.00039572
- sheaf_decay: 0.000040762
- second_linear: false
- use_act: true

General

- d: 3
- dropout: 0.9
- layers: 3
- add_hp: 0
- add_lp: 1
- edge_weights: true
- input_dropout: 0.7
- left_weights: true
- lr: 0.005
- normalized: true
- hidden_channels: 128
- right_weights: true

- weight_decay: 0.00045383
- sheaf_decay: 0.000022339
- second_linear: false
- use_act: true

Film

Diagonal

- d: 3
- dropout: 0.9
- layers: 3
- add_hp: 1
- add_lp: 1
- edge_weights: true
- input_dropout: 0
- left_weights: true
- lr: 0.02
- normalized: true
- hidden_channels: 16
- right_weights: true
- weight_decay: 0.0010625
- sheaf_decay: 0.000021583
- second_linear: true
- use_act: true

Bundle

- d: 4
- dropout: 0.9
- layers: 5
- add_hp: 1
- add_lp: 0
- edge_weights: true
- input_dropout: 0.1
- left_weights: true
- lr: 0.02
- normalized: true
- hidden_channels: 64
- right_weights: true
- weight_decay: 0.00039378
- sheaf_decay: 0.000034147
- second_linear: true
- use_act: true

General

- d: 4
- dropout: 0.9
- layers: 4
- add_hp: 1
- add_lp: 1
- edge_weights: true
- input_dropout: 0.5

- left_weights: true
- lr: 0.01
- normalized: true
- hidden_channels: 32
- right_weights: true
- weight_decay: 0.00010567
- sheaf_decay: 0.00048703
- second_linear: false
- use_act: true

Full Proofs

Table 34 summarizes the time and space complexity gains that are formally proven in this section.

Explicit Sheaf Laplacian

Theorem 1. *Constructing and applying the explicit normalized sheaf Laplacian $\Delta_{\mathcal{F}}\mathbf{X}$ costs:*

- **General:** $\mathcal{O}(md^3 + md^2 \log(md^2) + md^2 f)$;
- **Bundle:** $\mathcal{O}(md^3 + md^2 \log(md^2) + md^2 f)$;
- **Diagonal:** $\mathcal{O}(md \log(md) + md f)$.

Proof. Constructing and applying $\Delta_{\mathcal{F}}\mathbf{X}$ proceeds in four stages: (a) assembling the coboundary δ and forming the Laplacian as its inner product $L_{\mathcal{F}} = \delta^{\top} \delta$, (b) the degree pipeline, i.e. separately computing every $D_{\mathcal{F}(v)}$ and every $D_{\mathcal{F}(v)}^{-1/2}$, which is required for the symmetric normalization $\Delta_{\mathcal{F}} = D_{\mathcal{F}}^{-1/2} L_{\mathcal{F}} D_{\mathcal{F}}^{-1/2}$; (c) coalescing the resulting sparse tensor into a single COO operator (assumption 1); and (d) performing one sparse matrix multiplication against the $nd \times f$ feature matrix.

General sheaves. By Table 1, forming the coboundary inner product in stage (a) costs $\mathcal{O}(d^3)$ per directed incidence to build each $d \times d$ block, hence $\mathcal{O}(md^3)$ over all $2m$ incidences; the two normalizing multiplications by $D_{\mathcal{F}}^{-1/2}$ only add a constant factor per block and do not change the order. Stage (b): the degree blocks cost $\mathcal{O}(md^3)$ and the inverse square roots cost $\mathcal{O}(nd^3)$; the latter is a per-node cost, so by assumption 2 we have $nd^3 \leq 2md^3 = \mathcal{O}(md^3)$, thus this cost is absorbed, and stage (b) contributes $\mathcal{O}(md^3)$. Stage (c): the Laplacian has $n + 2m$ blocks of size $d \times d$, hence $(n + 2m)d^2 = \mathcal{O}(md^2)$ nonzero scalar entries after applying assumption 2; coalescing sorts these entries at cost $\mathcal{O}(md^2 \log(md^2))$. Stage (d): by assumption 1 the Sparse Matrix-Matrix Multiplication (SpMM) costs $\mathcal{O}(\text{nnz} \cdot f) = \mathcal{O}(md^2 f)$. Summing the four contributions and merging the identical md^3 terms from stages (a) and (b) gives $\mathcal{O}(md^3 + md^2 \log(md^2) + md^2 f)$.

Bundle (orthogonal) sheaves. The time complexity of the coboundary inner product taken at Stage (a) is unchanged ($\mathcal{O}(md^3)$) given orthogonal restriction maps. For Stage (b), since $D_{\mathcal{F}(v)} = \deg(v)I_d$, DEGREE reduces to edge counting at $\mathcal{O}(m)$ and DEGREEINV SQRT is a single scalar per node at $\mathcal{O}(n)$; by assumption 2, $\mathcal{O}(n) = \mathcal{O}(m)$, so stage (b) contributes $\mathcal{O}(m)$, which is dominated by the $\mathcal{O}(md^3)$ of stage (a). Stages (c) and (d) have the same nonzero structure as the general case, contributing $\mathcal{O}(md^2 \log(md^2))$ and $\mathcal{O}(md^2 f)$ respectively. Thus, we

have a cost of $\mathcal{O}(md^3 + md^2 \log(md^2) + md^2 f)$, matching the general case.

Diagonal sheaves. Every restriction map is a d -vector, so all block operations become elementwise. Stage (a): forming the Laplacian blocks via the coboundary inner product is an Hadamard product at $\mathcal{O}(md)$. Stage (b): DEGREE costs $\mathcal{O}(md)$ and DEGREEINV SQRT costs $\mathcal{O}(nd)$; by assumption 2, $nd \leq 2md = \mathcal{O}(md)$, so stage (b) contributes $\mathcal{O}(md)$. Stage (c): the operator now has $(n + 2m)d = \mathcal{O}(md)$ nonzeros, so coalescing costs $\mathcal{O}(md \log(md))$. Stage (d): the SpMM costs $\mathcal{O}(md \cdot f) = \mathcal{O}(md f)$. The $\mathcal{O}(md)$ contributions of stages (a) and (b) are each dominated by $\mathcal{O}(md \log(md))$, since $\log(md) \geq 1$, and are absorbed, leaving a cost of $\mathcal{O}(md \log(md) + md f)$. $\square$

Theorem 2. *The peak auxiliary memory of the explicit sheaf Laplacian is:*

- **General / Bundle:** $\mathcal{O}(md^2 f)$;
- **Diagonal:** $\mathcal{O}(md f)$.

Proof. Peak auxiliary memory is the largest amount of working memory allocated at any instant beyond the input, parameters and output. With explicit Laplacian operators, the dominant allocation is given by the gradient storage for the backward pass, which has to memorize the Laplacian operator, as well as a per-nonzero, per-channel activation. The Laplacian has $n + 2m$ blocks of size $d \times d$ for general and bundle sheaves, which by assumption 2 is $\mathcal{O}(md^2)$ words; for diagonal sheaves each block contributes only d entries, giving $\mathcal{O}(md)$ words. The intermediate features are thus $\mathcal{O}(md^2 f)$ ($\mathcal{O}(md f)$ in the diagonal case), dominating every other forward pass buffer. Thus peak memory is $\mathcal{O}(md^2 f)$ for general/bundle sheaves and $\mathcal{O}(md f)$ for diagonal sheaves. $\square$

Implicit Sheaf Laplacian

Theorem 3. *For any cellular sheaf $\mathcal{F}$ on $\mathcal{G}$, with uniform stalk dimension d , square $d \times d$ restriction maps and 0-cochain $\mathbf{X} \in \mathbb{R}^{nd \times f}$, Algorithm 1 computes $\Delta_{\mathcal{F}}\mathbf{X}$ exactly via Eq. 4 and 6 without materializing the $nd \times nd$ sparse matrix.*

Proof. We show that Algorithm 1 reproduces, node by node, the block action of the normalized Laplacian, using only per-node and per-edge tensors.

Step 1: the unnormalized block action. Fix a node v . By the definition of the sheaf Laplacian, block-row v of $L_{\mathcal{F}}$ has exactly one diagonal block, $D_{\mathcal{F}(v)}$, and one off-diagonal block $-\mathcal{F}_{v \triangleleft e}^{\top} \mathcal{F}_{u \triangleleft e}$ for each edge e incident to v with neighbor u ; every other block in that row is zero. Therefore

$$(L_{\mathcal{F}}\mathbf{X})_v = D_{\mathcal{F}(v)}\mathbf{x}_v - \sum_{u: v \triangleleft e} (\mathcal{F}_{v \triangleleft e}^{\top} \mathcal{F}_{u \triangleleft e})\mathbf{x}_u, \quad (12)$$

$$= D_{\mathcal{F}(v)}\mathbf{x}_v - \sum_{u: v \triangleleft e} \mathcal{F}_{v \triangleleft e}^{\top} (\mathcal{F}_{u \triangleleft e} \mathbf{x}_u). \quad (13)$$

Where the last equality re-parenthesizes each summand by associativity of matrix multiplication, letting $\mathcal{F}_{u \triangleleft e}$ act on $\mathbf{x}_u$ before $\mathcal{F}_{v \triangleleft e}^{\top}$ acts on the result. The trailing sum is $\text{TwoBMM}(\mathcal{F}, \mathbf{X})_v$, giving Equation (4).

Table 34: Complexity comparison. Only rows where at least one implementation differs are shown. Bodnar et al. (2022) only implemented the Sheaf Laplacian. n : nodes, m : edges, d : stalk dim, f : features/channels.

		Bodnar orig.	Explicit (ours)	Implicit (ours)
Time	Rev.-edge lookup (both)	$\mathcal{O}(m)$ CPU dict	$\mathcal{O}(m \log m)$ GPU	$\mathcal{O}(m \log m)$ GPU
	Sparse index (ISL)	$\mathcal{O}(md^2)$	$\mathcal{O}(md^2)$	—
	Sparse index (ISA)	—	$\mathcal{O}(md^2 + nd^2)$	—
	Sparse coalescing (ISL)	$\mathcal{O}(md^2 \log md^2)$	$\mathcal{O}(md^2 \log md^2)$	—
	Sparse coalescing (ISA)	—	$\mathcal{O}(md^2 \log md^2)$	—
	Transport map form. (diag)	$\mathcal{O}(md)$	$\mathcal{O}(md)$	—
	Transport map form. (bundle)	$\mathcal{O}(md^3)$	$\mathcal{O}(md^3)$	—
	Transport map form. (general)	$\mathcal{O}(md^3)$	$\mathcal{O}(md^3)$	—
	Operator application (diag)	$\mathcal{O}(mdf)$	$\mathcal{O}(mdf)$	$\mathcal{O}(mdf)$
	Operator application (bundle)	$\mathcal{O}(md^2 f)$	$\mathcal{O}(md^2 f)$	$\mathcal{O}(md^2 f)$
	Operator application (general)	$\mathcal{O}(md^2 f)$	$\mathcal{O}(md^2 f)$	$\mathcal{O}(md^2 f)$
Memory	Sparse matrix (ISL)	$\mathcal{O}(md^2)$	$\mathcal{O}(md^2)$	$\mathcal{O}(m)$
	Sparse matrix (ISA)	—	$\mathcal{O}(md^2 + nd^2)$	$\mathcal{O}(m)$
	Backward Pass (gen/bundle)	$\mathcal{O}(md^2 f)$	$\mathcal{O}(md^2 f)$	$\mathcal{O}(md^2 + mdf)$
	Backward Pass (diag)	$\mathcal{O}(mdf)$	$\mathcal{O}(mdf)$	$\mathcal{O}(mdf)$

Step 2: the algorithm realizes Step 1. Lines 1–3 of Algorithm 1 compute $D_{\mathcal{F}(v)}$ and its inverse square root. Line 4 applies the pre-scale $\mathbf{x}_v \mapsto \mathbf{y}_v := D_{\mathcal{F}(v)}^{-1/2} \mathbf{x}_v$ to every node. Line 6 forms $D_{\mathcal{F}(v)} \mathbf{y}_v$, the diagonal contribution. Lines 7–9 use the reverse-edge index $\mathbf{r}$ to pair, for each directed edge, the head map $\mathcal{F}_{v \leq e}$ with the tail map $\mathcal{F}_{u \leq e}$, and compute $\text{step2}_e = \mathcal{F}_{v \leq e}^\top (\mathcal{F}_{u \leq e} \mathbf{y}_u)$ via two batched matrix multiplications: the first (line 8) forms $\mathcal{F}_{u \leq e} \mathbf{y}_u$, the second (line 9) left-multiplies by $\mathcal{F}_{v \leq e}^\top$. The scatter-add of line 10 sums these per-edge results into the destination node v , producing $\sum_{u: v \leq e} \mathcal{F}_{v \leq e}^\top (\mathcal{F}_{u \leq e} \mathbf{y}_u) = \text{TwoBMM}(\mathcal{F}, \mathbf{Y})_v$. Line 11 subtracts, giving $D_{\mathcal{F}(v)} \mathbf{y}_v - \text{TwoBMM}(\mathcal{F}, \mathbf{Y})_v = (L_{\mathcal{F}} \mathbf{Y})_v$ by Step 1 applied to $\mathbf{Y}$.

Step 3: normalization. Because $D_{\mathcal{F}}^{-1/2}$ is block-diagonal, the symmetric composition $\Delta_{\mathcal{F}} = D_{\mathcal{F}}^{-1/2} L_{\mathcal{F}} D_{\mathcal{F}}^{-1/2}$ factors, node by node, into a pre-scale of the input and a post-scale of the output. With $\mathbf{y}_v = D_{\mathcal{F}(v)}^{-1/2} \mathbf{x}_v$ from line 4, the post-scale of line 13 yields $D_{\mathcal{F}(v)}^{-1/2} (L_{\mathcal{F}} \mathbf{Y})_v = (D_{\mathcal{F}}^{-1/2} L_{\mathcal{F}} D_{\mathcal{F}}^{-1/2} \mathbf{X})_v = (\Delta_{\mathcal{F}} \mathbf{X})_v$, which is Equation (6).

Every object allocated by Algorithm 1 is indexed either by node (the degree blocks, the inverse roots, the pre/post-scaled features) or by directed edge (the gathered maps and the two-BMM intermediates). At no point is an $nd \times nd$ Laplacian formed. $\square$

Theorem 4. Algorithm 1 computes $\Delta_{\mathcal{F}} \mathbf{X}$ in:

- **General:** $\mathcal{O}(m \log m + md^3 + md^2 f)$;
- **Bundle:** $\mathcal{O}(m \log m + md^2 f)$;
- **Diagonal:** $\mathcal{O}(m \log m + mdf)$.

This eliminates the $\mathcal{O}(md^2 \log(md^2))$ coalescing and $\mathcal{O}(md^3)$ transport-map costs of the explicit path (Thm. 1).

Proof. Algorithm 1 has three cost sources: building the reverse-edge index, running the degree pipeline (lines 1–5,

the degree term in line 6, and the post-scale in line 13), and performing the two-BMM gather–scatter (lines 7–10). We cost each and sum, family by family, using assumption 2 to absorb per-node costs.

Reverse-edge index. The index $\mathbf{r}$ is obtained by sorting the $2m$ directed pairs once per topology, at cost $\mathcal{O}(m \log m)$; this term is common to all three families and is cached across forward passes for a fixed graph.

General sheaves. The degree pipeline costs $\mathcal{O}(md^3)$ for the degree blocks plus $\mathcal{O}(nd^3)$ for the inverse roots (Table 1); by assumption 2, $nd^3 \leq 2md^3 = \mathcal{O}(md^3)$, so the inverse-root term merges into the degree term, contributing $\mathcal{O}(md^3)$. The pre-scale, post-scale, and degree-term multiply are per-node BMMs costing $\mathcal{O}(nd^2 f)$, which by assumption 2 is $\mathcal{O}(md^2 f)$. The two-BMM gather–scatter costs $\mathcal{O}(md^2 f)$ (Table 1). Summing,

$$\mathcal{O}(m \log m) + \mathcal{O}(md^3) + \mathcal{O}(md^2 f) + \mathcal{O}(md^2 f) \quad (14)$$

$$= \mathcal{O}(m \log m + md^3 + md^2 f). \quad (15)$$

The only md^3 term arises from the degree computation, not from the formation of transport maps, thus the implicit path never forms $\mathcal{F}_{v \leq e}^\top \mathcal{F}_{u \leq e}$.

Bundle sheaves. The degree pipeline costs $\mathcal{O}(m)$ (degrees) plus $\mathcal{O}(n)$ (scalar inverse roots); by assumption 2 this is $\mathcal{O}(m)$. The per-node BMMs cost $\mathcal{O}(ndf)$, which by assumption 2 is $\mathcal{O}(mdf) \leq \mathcal{O}(md^2 f)$. The two-BMM costs $\mathcal{O}(md^2 f)$. Since $\mathcal{O}(m)$ is dominated by $\mathcal{O}(md^2 f)$, the bound collapses to $\mathcal{O}(m \log m + md^2 f)$: no md^3 term survives, because for bundle sheaves the degree computation reduces to edge counting and transport-map formation never occurs.

Diagonal sheaves. The degree pipeline costs $\mathcal{O}(md)$ (degrees) plus $\mathcal{O}(nd)$ (elementwise inverse roots); by assumption 2, $nd \leq 2md = \mathcal{O}(md)$. The per-node BMMs cost $\mathcal{O}(ndf) = \mathcal{O}(mdf)$ by assumption 2. The two-BMM costs $\mathcal{O}(mdf)$. Since $\mathcal{O}(md)$ is dominated by $\mathcal{O}(mdf)$, the bound is $\mathcal{O}(m \log m + mdf)$.

In all three families the algorithm performs no coalescing and forms no transport maps, so both the $\mathcal{O}(md^2 \log(md^2))$ term and the $\mathcal{O}(md^3)$ transport-map term of Theorem 1 are absent, which is the claimed elimination. $\square$

Theorem 5. *The peak auxiliary memory of Algorithm 1 is:*

- **General / Bundle:** $\mathcal{O}(md^2 + mdf)$;
- **Diagonal:** $\mathcal{O}(mdf)$;

Proof. Since Algorithm 1 never assembles a sparse operator, the $\mathcal{O}(md^2)$ storage cost of the explicit path (Theorem 2) does not arise here. We instead bound the peak memory by listing every buffer the algorithm keeps alive and taking the largest.

(i) **Index structures.** The arrays *row*, *col* and the reverse-edge index *r* each have length $2m$, contributing $\mathcal{O}(m)$ words in total.

(ii) **Edge-wise activations.** Computing gradients through the two batched multiplies requires retaining their inputs: the gathered restriction maps $\mathcal{F}[\mathbf{r}]$, and the intermediate tensors *step1* and *step2*. The gathered maps occupy $\mathcal{O}(md^2)$ words for general/bundle sheaves, or $\mathcal{O}(md)$ for diagonal sheaves (whose restriction maps are *d*-vectors); the two intermediates occupy $\mathcal{O}(mdf)$ each.

(iii) **Node-wise buffers.** The degree blocks and their inverse square roots occupy $\mathcal{O}(nd^2)$ words for general/bundle sheaves, or $\mathcal{O}(nd)$ for diagonal sheaves; the pre- and post-scaled node features occupy $\mathcal{O}(ndf)$. By Assumption 2 ($n \leq 2m$), these are respectively $\mathcal{O}(md^2)$, $\mathcal{O}(md)$ and $\mathcal{O}(mdf)$, dominated by (ii).

Taking the maximum over (i)–(iii): for general and bundle sheaves the edge-wise activations dominate at $\mathcal{O}(mdf + md^2)$, giving an overall peak memory of $\mathcal{O}(mdf + md^2)$; for diagonal sheaves the gathered maps shrink to $\mathcal{O}(md) \leq \mathcal{O}(mdf)$, giving an upper bound of $\mathcal{O}(mdf)$. $\square$

Explicit Augmented Sheaf Adjacency

Theorem 6. *Constructing and applying the explicit normalized augmented sheaf adjacency $\hat{\mathbf{A}}_{\mathcal{F}} \mathbf{X}$ costs:*

- **General:** $\mathcal{O}(md^3 + md^2 \log(md^2) + md^2 f)$;
- **Bundle:** $\mathcal{O}(md^3 + md^2 \log(md^2) + md^2 f)$;
- **Diagonal:** $\mathcal{O}(md \log(md) + mdf)$.

Proof. We reduce to the four-stage accounting of Theorem 1 and show that the augmented adjacency changes that accounting only by lower-order or absorbed terms.

Recall $\mathbf{A}_{\mathcal{F}}^{(k)} = \mathbf{A}_{\mathcal{F}}^{(k)} + I_{nd}$, the augmented sheaf degree $\tilde{D}_{\mathcal{F}}^{(k)}(v) = D_{\mathcal{F}}^{(k)}(v) + I_d$ and $\hat{\mathbf{A}}_{\mathcal{F}}^{(k)} = \tilde{D}_{\mathcal{F}^{(k)}}^{-1/2} \mathbf{A}_{\mathcal{F}}^{(k)} \tilde{D}_{\mathcal{F}^{(k)}}^{-1/2}$. Compared with the Laplacian, the off-diagonal blocks are $+\mathcal{F}_{v \leq e}^{\top} \mathcal{F}_{u \leq e}$ instead of $-\mathcal{F}_{v \leq e}^{\top} \mathcal{F}_{u \leq e}$, a sign carrying no additional impact to computational complexity. The diagonal blocks differ: where the Laplacian carries the degree blocks $D_{\mathcal{F}(v)}$, the augmented adjacency carries the n normalized self-loop blocks $\tilde{D}_{\mathcal{F}(v)}^{-1/2} I_d \tilde{D}_{\mathcal{F}(v)}^{-1/2} = \tilde{D}_{\mathcal{F}(v)}^{-1}$.

We go through four stages of Theorem 1. Stage (a), forming the off-diagonal transport maps, is identical to Theorem 1. Stage (b), the degree pipeline, is unchanged, since normalization still requires forming every $\tilde{D}_{\mathcal{F}(v)}^{-1/2}$; for general sheaves

the self-loop blocks $\tilde{D}_{\mathcal{F}(v)}^{-1}$ are by-products of the same eigen-decomposition already counted in the $\mathcal{O}(nd^3)$ inverse-root cost, and materializing them explicitly adds only $\mathcal{O}(nd^2)$, which by assumption 2 is $\mathcal{O}(md^2)$; for bundle and diagonal sheaves the analogous cost is $\mathcal{O}(nd) = \mathcal{O}(md)$. Stages (c) and (d) now run over $\mathcal{O}(md^2 + nd^2)$ (general/bundle) resp. $\mathcal{O}(md + nd)$ (diagonal) nonzeros; by assumption 2 these collapse to $\mathcal{O}(md^2)$ resp. $\mathcal{O}(md)$, exactly as in Theorem 1.

Since every additional term introduced by augmentation is either free (the sign) or a per-node cost absorbed by assumption 2, the four-stage sum is identical to Theorem 1 in each family. $\square$

Theorem 7. *The peak auxiliary memory of the explicit augmented sheaf adjacency is:*

- **General / Bundle:** $\mathcal{O}(md^2 f)$;
- **Diagonal:** $\mathcal{O}(mdf)$;

Proof. The proof follows the same argument as Theorem 2: the dominant allocation is the assembled sparse operator, whose size equals its nonzero count (assumption 1), and we identify that count and then apply assumption 2 to it.

The augmented adjacency $\mathbf{A}_{\mathcal{F}} = \mathbf{A}_{\mathcal{F}} + I_{nd}$ has the same block structure as the Laplacian, namely $n + 2m$ blocks of size $d \times d$: the $2m$ off-diagonal blocks $\mathcal{F}_{v \leq e}^{\top} \mathcal{F}_{u \leq e}$, shared with $L_{\mathcal{F}}$, and n diagonal blocks, which here are the normalized self-loop blocks $\tilde{D}_{\mathcal{F}(v)}^{-1}$ in place of the Laplacian's degree blocks $\tilde{D}_{\mathcal{F}(v)}$. Crucially, the self-loop blocks occupy exactly the same shape as the degree blocks: dense $d \times d$ for general and bundle sheaves, and a d -vector for diagonal sheaves. Consequently the two operators have an identical nonzero count, $(n + 2m)d^2$ scalar entries for general/bundle sheaves and $(n + 2m)d$ for diagonal sheaves. By the same argument as Theorem 2, we consider the backward pass buffers as the peak memory allocation. We must therefore consider an additional factor of f . Thus the peak auxiliary memory is $\mathcal{O}(md^2 f)$ for general and bundle sheaves and $\mathcal{O}(mdf)$ for diagonal sheaves, identical to Theorem 2. $\square$

Implicit Sheaf Adjacency

Theorem 8. *For any cellular sheaf $\mathcal{F}$ on $\mathcal{G}$, with uniform stalk dimension d , square $d \times d$ restriction maps and 0-cochain $\mathbf{X} \in \mathbb{R}^{nd \times f}$, Algorithm 2 computes $\hat{\mathbf{A}}_{\mathcal{F}} \mathbf{X}$ exactly via Eq. 9 and 10 without materializing an $nd \times nd$ matrix.*

Proof. The argument mirrors Theorem 3, with two localized changes, so we highlight exactly where Algorithm 2 departs from Algorithm 1.

Step 1: the unnormalized block action. Fix a node v . Block-row v of $\mathbf{A}_{\mathcal{F}} = \mathbf{A}_{\mathcal{F}} + I_{nd}$ has one diagonal block equal to the identity I_d (the self-loop), and one off-diagonal block $+\mathcal{F}_{v \leq e}^{\top} \mathcal{F}_{u \leq e}$ for each incident edge with neighbor u . Hence

$$(\mathbf{A}_{\mathcal{F}} \mathbf{X})_v = I_d \mathbf{x}_v + \sum_{u: v \leq e} \mathcal{F}_{v \leq e}^{\top} (\mathcal{F}_{u \leq e} \mathbf{x}_u) \quad (16)$$

$$= \mathbf{x}_v + \text{TwoBMM}(\mathcal{F}, \mathbf{X})_v, \quad (17)$$

using the same associativity re-parenthesization as in Theorem 3. This is Eq. 9.

Step 2: differences from Algorithm 1. Algorithm 2 differs from Algorithm 1 in exactly two lines. First, line 7 initializes the output with the self-loop copy $\text{out} \leftarrow \mathbf{Y}$, in place of the degree term $\tilde{D}_{\mathcal{F}(v)} \mathbf{y}_v$ used by the Laplacian. Second, line 12 *adds* the scattered aggregation rather than subtracting it. Everything else, in particular the two-BMM in lines 8–11, is identical to Algorithm 1; therefore, by Step 2 of the proof of Theorem 3, after line 11 we again have $\text{adj_term}_v = \text{TwoBMM}(\mathcal{F}, \mathbf{Y})_v$.

Step 3: assembling the normalized output. Combining the two departures, line 12 produces $\mathbf{y}_v + \text{TwoBMM}(\mathcal{F}, \mathbf{Y})_v = (\mathbf{A}_{\mathcal{F}} \mathbf{Y})_v$ by Step 1 applied to $\mathbf{Y}$, where $\mathbf{y}_v = \tilde{D}_{\mathcal{F}(v)}^{-1/2} \mathbf{x}_v$ is the pre-scale from line 5. The post-scale of line 14 then yields $\tilde{D}_{\mathcal{F}(v)}^{-1/2} (\mathbf{A}_{\mathcal{F}} \mathbf{Y})_v = (\tilde{D}_{\mathcal{F}}^{-1/2} \mathbf{A}_{\mathcal{F}} \tilde{D}_{\mathcal{F}}^{-1/2} \mathbf{X})_v = (\hat{\mathbf{A}}_{\mathcal{F}} \mathbf{X})_v$, which is Eq. 10.

Step 4: the self-loop is never normalized as a block. The identity contribution is realized as a feature-space copy of $\mathbf{Y}$, not as a stored block, so the normalized self-loop blocks $\tilde{D}_{\mathcal{F}(v)}^{-1}$ that the explicit path materializes (Theorem 7) are never formed here. As in ISL, every tensor is indexed per node or per directed edge, so no $nd \times nd$ matrix is created. $\square$

Theorem 9. *Algorithm 2 computes $\hat{\mathbf{A}}_{\mathcal{F}} \mathbf{X}$ in:*

- **General:** $\mathcal{O}(m \log m + md^3 + md^2 f)$;
- **Bundle:** $\mathcal{O}(m \log m + md^2 f)$;
- **Diagonal:** $\mathcal{O}(m \log m + md f)$;

matching ISL (Thm. 4) in all families.

Proof. We transfer the accounting of Theorem 4 line by line, identifying the single substitution and checking that it does not change any bound.

The reverse-edge index ($\mathcal{O}(m \log m)$), the degree pipeline and the two-BMM gather–scatter are computed exactly as in Algorithm 1, with identical costs by family. The only difference is in the diagonal contribution: Algorithm 1 forms the degree term $\tilde{D}_{\mathcal{F}(v)} \mathbf{y}_v$, whereas Algorithm 2 forms the self-loop copy $\text{out} \leftarrow \mathbf{Y}$. The degree-term multiply is a per-node BMM costing $\mathcal{O}(nd^2 f)$ (general) or $\mathcal{O}(nd f)$ (bundle/diagonal); the self-loop copy costs $\mathcal{O}(nd f)$. Both are per-node costs, and by assumption 2 we have $nd f \leq \mathcal{O}(md f) \leq \mathcal{O}(md^2 f)$, so each is dominated by the two-BMM term $\mathcal{O}(md^2 f)$ (resp. $\mathcal{O}(md f)$ for diagonal). The substitution therefore leaves every family bound unchanged.

One point deserves emphasis: even though ISA replaces the degree term by a copy, the degree pipeline still runs in the normalized setting, since line 1 must still compute every $\tilde{D}_{\mathcal{F}(v)}^{-1/2}$ for the pre- and post-scales. Hence the md^3 term in the general case (from the degree blocks) is present in ISA exactly as in ISL. Summing gives the three bounds, identical to Theorem 4. $\square$

Theorem 10. *The peak auxiliary memory of Algorithm 2 is:*

- **General / Bundle:** $\mathcal{O}(md^2 + md f)$;
- **Diagonal:** $\mathcal{O}(md f)$;

Proof. We show that Algorithm 2 ISA allocates a subset of the live autograd buffers allocated by Algorithm 1 ISL,

establishing that its peak memory cannot exceed the ISL bound, and then verify that it matches it.

Comparing the two algorithms line by line, Algorithm 2 uses identical index arrays (row, col, r of total size $\mathcal{O}(m)$), the same gathered restriction maps $\mathcal{F}[\mathbf{r}]$ ($\mathcal{O}(md^2)$ for general/bundle sheaves, $\mathcal{O}(md)$ for diagonal), and the same two-BMM feature intermediates step1 and step2 ($\mathcal{O}(md f)$ each). The only buffer that changes is the diagonal contribution: the degree-term buffer of size $\mathcal{O}(nd^2 f)$ (general) or $\mathcal{O}(nd f)$ (bundle) in ISL is replaced by a self-loop identity copy of size $\mathcal{O}(nd f)$. Under assumption 2 ($n \leq 2m$), $\mathcal{O}(nd f) \leq \mathcal{O}(md f)$, which is strictly dominated by the edge-wise intermediates. Hence, every live buffer of Algorithm 2 is bounded by the corresponding buffer of Algorithm 1, yielding an overall peak training memory of $\mathcal{O}(md^2 + md f)$ for general/bundle sheaves and $\mathcal{O}(md f)$ for diagonal sheaves, matching Theorem 5.

Finally, the comparison with the explicit path (Theorem 7) follows directly from autograd activation accounting. As established in Theorem 7, the explicit path requires $\mathcal{O}(md^2 f)$ peak training memory due to the backward pass buffers. By avoiding explicit matrix assembly, Algorithm 2 avoids constructing both the $2m$ off-diagonal blocks and the n self-loop blocks, completely eliminating the $\mathcal{O}(md^2 f)$ per-nonzero activation tensor. Instead, ISA requires only $\mathcal{O}(md f)$ feature activations and $\mathcal{O}(md^2)$ gathered maps during training, achieving a peak memory reduction from $\mathcal{O}(md^2 f)$ to $\mathcal{O}(md^2 + md f)$ for general/bundle sheaves (for a reduction of $\mathcal{O}(\min(d, f))$), while both paths require $\mathcal{O}(md f)$ for diagonal sheaves. $\square$