

876
877
878

Copresheaf Topological Neural Networks: A Generalized Deep Learning Framework –Supplementary Material–

879 Contents

880	A Notation	2
881	B Sheaves and Copresheaves on Graphs: A Category Theoretical Look	2
882	B.1 Copresheaves	2
883	B.2 Cellular Sheaves	3
884	B.3 Comparison between copresheaves and cellular sheaves	3
885	B.4 Copresheaf Neighborhood Matrix Example	4
886	C Expressive power of CTNNs	5
887	C.1 Universal Approximation of N -Dependent Copresheaves	5
888	D Sheaf Neural Networks Are Copresheaf Message-Passing Neural Networks	6
889	E General Copresheaf-Based Transformer Layer	7
890	F Copresheaf Learning on Euclidean Data	8
891	G Experiments	9
892	G.1 Synthetic Control Tasks	9
893	G.1.1 Structure Recognition datasets	9
894	G.1.2 Classifying Hierarchical Polygons	10
895	G.1.3 Airfoil Self-Noise Regression	10
896	G.2 Pixelwise Regression Tasks: Evaluating CopresheafConv2D Layers	11
897	G.2.1 Learning Token-Relations with Copresheaf Attention	12
898	G.2.2 Segment-wise Token Classification	13
899	G.2.3 Topological Shape Classification	14
900	G.3 TREC Text Classification Task	14
901	G.4 Catalogue of copresheaf maps	14
902	H Extended Related Work on Topological and Sheaf Neural Networks	15

903 A Notation

904 This appendix provides a reference summary of the notation and acronyms used throughout the main
 905 text. Table 4 details key mathematical symbols, while Table 5 lists abbreviations and their expansions.

Notation	Description
S	Underlying vertex set of a combinatorial complex (Def. 1)
\mathcal{X}	Set of nonempty cells $\subseteq \mathcal{P}(S)$ (Def. 1)
$\mathcal{P}(S)$	Power set of S (Def. 1)
$\text{rk} : \mathcal{X} \rightarrow \mathbb{Z}_{\geq 0}$	Rank function on cells, mapping to non-negative integers (Def. 1)
$\mathbb{Z}_{\geq 0}$	Non-negative integers
\mathcal{X}^k	$\{x \in \mathcal{X} : \text{rk}(x) = k\}$, the k -cells (Def. 1)
$\dim \mathcal{X}$	Dimension of the complex, $\max_{x \in \mathcal{X}} \text{rk}(x)$ (Sec. 2.1)
$\mathcal{N} : \mathcal{X} \rightarrow \mathcal{P}(\mathcal{X})$	Neighborhood function, mapping cells to sets of neighbor cells (Def. 2)
\mathcal{N}_{adj}	Adjacency neighborhood function (Def. 2)
\mathcal{N}_{inc}	Incidence neighborhood function (Def. 2)
$\mathcal{X}_{\mathcal{N}}$	Effective support of \mathcal{N} , $\{x \in \mathcal{X} \mid \mathcal{N}(x) \neq \emptyset\}$ (Sec. 3)
$\mathbb{G} \in \{0, 1\}^{m \times n}$	Binary neighborhood matrix (Def. 3)
$\mathbb{G}^{\mathcal{N}}$	Copresheaf neighborhood matrix, entries $\rho_{z_i \rightarrow y_j}$ or 0 (Def. 7)
$\rho_{y \rightarrow x}$	Copresheaf morphism $\mathcal{F}(y) \rightarrow \mathcal{F}(x)$ for edge $y \rightarrow x$ (Def. 4)
$\mathbb{A}_{r,k}$	Copresheaf adjacency matrix between r -cells (Def. 8)
$\mathbb{B}_{r,k}$	Copresheaf incidence matrix between ranks r and k (Def. 8)
$C^k(\mathcal{X}, \mathcal{F})$	k -cochain space, $\bigoplus_{x \in \mathcal{X}^k} \mathcal{F}(x)$ (Sec. 3)
$\text{Hom}(\mathcal{F}(i), \mathcal{F}(j))$	Space of linear maps from $\mathcal{F}(i)$ to $\mathcal{F}(j)$ (Def. 7)
$\mathbf{h}_x^{(\ell)}$	Feature vector at cell x in layer ℓ (Prop. 1)
α	Learnable message function (Prop. 1)
β	Learnable update function (Prop. 1)
\oplus	Permutation-invariant aggregator (Prop. 1, Prop. 3)
$G = (V, E)$	Directed or undirected graph (Sec. 2.2)
$G_{\mathcal{N}} = (\mathcal{X}_{\mathcal{N}}, E_{\mathcal{N}})$	Directed graph induced by \mathcal{N} , edges $y \rightarrow x$ if $y \in \mathcal{N}(x)$ (Def. 6)
$\mathcal{F}(x), \mathcal{F}(e)$	Stalks: vector spaces at vertex x or edge e (Def. 4, Def. 5)
$\mathcal{F}_{x \leq e} : \mathcal{F}(x) \rightarrow \mathcal{F}(e)$	Restriction map in a cellular sheaf for $x \leq e$ (Def. 5)
δ	Coboundary map, measures local disagreement in cellular sheaf (Sec. 2.2)
$\Delta_{\mathcal{F}}$	Sheaf Laplacian, $\Delta_{\mathcal{F}} = \delta^T \delta$ (Sec. 2.2)
\mathcal{Y}, \mathcal{Z}	Collections of cells, used in neighborhood matrices (Def. 3, Def. 7)
W_q, W_k, W_v	Learnable projection matrices for queries, keys, values in copresheaf self-attention (Prop. 4)
q_x, k_x, v_x	Query, key, and value vectors for cell x in copresheaf self-attention (Prop. 4)
a_{xy}	Attention coefficient for cells x and y in copresheaf self-attention (Prop. 4)

Table 4: Summary of key notation used throughout the paper.

906 B Sheaves and Copresheaves on Graphs: A Category Theoretical Look

907 This appendix provides a category-theoretic exposition of sheaves and copresheaves, emphasizing
 908 their definitions within the language of category theory. Additionally, we illustrate the construction
 909 of a copresheaf neighborhood matrix through an explicit combinatorial example, instantiating the
 910 concepts developed in the main text.

911 B.1 Copresheaves

912 Before diving into the technical definition, it’s helpful to think of a *copresheaf* as a way of assigning
 913 data that flows along the structure of a graph—like signals along neurons, or resources in a network.
 914 In categorical terms, this structure formalizes the idea of consistently associating elements of some
 915 category \mathcal{C} .

916 **Definition 9** (Copresheaf on a Directed Graph). Let $G = (V, E)$ be a directed graph, and let \mathcal{C} be a
 917 category. A *copresheaf* on G is a functor $\mathcal{F} : G \rightarrow \mathcal{C}$, where the graph G is regarded as a category
 918 whose objects are the vertices V , and whose morphisms are the directed edges $(x \rightarrow y) \in E$. When
 919 $\mathcal{C} = \mathbf{Vect}_{\mathbb{R}}$, this structure corresponds to a *quiver representation*.

Acronym	Expansion
CC	Combinatorial Complex
CTNN	Copresheaf Topological Neural Network
CMPNN	Copresheaf Message-Passing Neural Network
SNN	Sheaf Neural Network
GNN	Graph Neural Network
CNN	Convolutional Neural Network
CT	Copresheaf Transformer
CGNN	Copresheaf Graph Neural Network
GCN	Graph Convolutional Network
GraphSAGE	Graph Sample and Aggregate
GIN	Graph Isomorphism Network
CopresheafGCN	Copresheaf Graph Convolutional Network
CopresheafSage	Copresheaf Graph Sample and Aggregate
CopresheafGIN	Copresheaf Graph Isomorphism Network
NSD	Neural Sheaf Diffusion
SAN	Sheaf Attention Network
MLP	Multi-Layer Perceptron
GAT	Graph Attention Network
CAM	Copresheaf Adjacency Matrix
CIM	Copresheaf Incidence Matrix
CNM	Copresheaf Neighborhood Matrix

Table 5: List of acronyms used throughout the paper.

B.2 Cellular Sheaves

Definition 10 (Cellular Sheaf on an Undirected Graph). Let $G = (V, E)$ be an undirected graph. Define the *incidence poset* (P, \leq) , where $P = V \cup E$, and the order relation is given by $x \leq e$ whenever vertex $x \in V$ is incident to edge $e \in E$. A *cellular sheaf* on G with values in a category \mathcal{C} is a functor $\mathcal{F} : P \rightarrow \mathcal{C}$, which assigns:

- to each vertex $x \in V$, an object $\mathcal{F}(x) \in \mathcal{C}$;
 - to each edge $e \in E$, an object $\mathcal{F}(e) \in \mathcal{C}$;
 - to each incidence relation $x \leq e$, a morphism $\rho_{x \leq e} : \mathcal{F}(x) \rightarrow \mathcal{F}(e)$, called a *restriction map*,
- such that the functoriality condition is satisfied on composable chains in the poset.

B.3 Comparison between copresheaves and cellular sheaves

Copresheaves provide a versatile and powerful framework for machine learning applications across diverse domains, particularly excelling in scenarios where directional data flow and hierarchical dependencies are paramount. Unlike cellular sheaves, which are defined over undirected graphs and enforce consistency through restriction maps $\rho_{x \leq e} : \mathcal{F}(x) \rightarrow \mathcal{F}(e)$, copresheaves operate on directed graphs, assigning learnable linear maps $\rho_{x \rightarrow y} : \mathcal{F}(x) \rightarrow \mathcal{F}(y)$ along edges. This enables anisotropic information propagation, making them ideal for tasks such as physical simulations—where data flows asymmetrically, as in fluid dynamics or heat transfer—or natural language processing, where sequential word dependencies dominate. More importantly, the vertex-centric design, assigning vector spaces $\mathcal{F}(x)$ solely to vertices, aligns seamlessly with message-passing architectures like Graph Neural Networks (GNNs) and Topological Neural Networks (TNNs), allowing these maps to be parameterized and optimized during training. Empirical evidence from our experiments highlights that copresheaf-based models outperform traditional architectures in capturing complex dynamics, demonstrating their superior ability to model spatially varying patterns and long-range dependencies in general applications. See Table 6 for a summary of the comparison between copresheaves and cellular sheaves.

Furthermore, copresheaves enhance machine learning models with a principled approach to regularization and expressiveness, broadening their suitability across heterogeneous domains. Standard neural network regularizers, such as ℓ_2 decay or dropout, can be readily applied to copresheaf maps,

with optional structural losses like path-consistency ensuring morphism compositionality. This adaptability stands in stark contrast to the rigid cohomological constraints of cellular sheaves, which enforce local-to-global agreement through terms like $\|\mathcal{F}_{x \leq e}(\mathbf{h}_x) - \mathbf{h}_e\|^2$, limiting their flexibility in domains with asymmetric relationships. Copresheaves, by learning edge-wise maps, offer greater expressiveness for tasks involving non-Euclidean or multi-scale data, as evidenced by the superior performance of CopresheafConv layers on grid-based tasks. This makes them particularly effective for applications such as image segmentation, 3D mesh processing, or token-relation learning, where traditional methods like Convolutional Neural Networks (CNNs) struggle with directional or hierarchical structures. Our experiments further corroborate that copresheaf-augmented models consistently improve accuracy and detail recovery across diverse tasks, positioning them as a more suitable and generalizable tool for machine learning applications spanning Euclidean and non-Euclidean domains alike.

Table 6: Comparison between copresheaves and cellular sheaves.

Aspect	Copresheaf	Cellular Sheaf Hansen and Ghrist [2019c]
Graph type	Directed graph $G = (V, E)$	Undirected graph $G = (V, E)$
Assigned to vertices	Vector space $\mathcal{F}(x)$ for each $x \in V$	Vector space $\mathcal{F}(x)$ for each $x \in V$
Assigned to edges	Linear map $\rho_{x \rightarrow y} : \mathcal{F}(x) \rightarrow \mathcal{F}(y)$ for each directed edge $x \rightarrow y \in E$	Vector space $\mathcal{F}(e)$ for each edge $e \in E$
Associated maps	Pushforward: moves data forward along edges	Restriction: pulls data back from vertices to edges
Map direction	$\mathcal{F}(x) \rightarrow \mathcal{F}(y)$ (source to target)	$\mathcal{F}(x) \rightarrow \mathcal{F}(e)$ (vertex to incident edge)
Interpretation	Nodes have local features; edges transform and transmit them	Edges represent shared contexts; vertex features are restricted into them
Goal / Objective	Learn and compose edge-wise feature-space maps	Enforce coherence across by gluing local data
Typical Regularization	Standard NN regularizers (ℓ_2 decay, spectral-norm, dropout, norm); optional structure losses (path-consistency, holonomy)	Agreement between restricted vertex features and the edge-stalk, e.g., $\ \mathcal{F}_{x \leq e}(\mathbf{h}_x) - \mathbf{h}_e\ ^2$
Use in learning	Embedding-level message passing, directional influence, anisotropic information flow	Compatibility across shared structures, enforcing local consistency, cohomological constraints

B.4 Copresheaf Neighborhood Matrix Example

We define a copresheaf neighborhood matrix (CNM) for a combinatorial complex with an incidence neighborhood, guiding the reader through the setup, neighborhood, graph, copresheaf, and matrix.

Setup. Consider a combinatorial complex $\mathcal{X} = (\mathcal{S}, \mathcal{X}, \text{rk})$ with $\mathcal{S} = \{a, b, c\}$, cells $\mathcal{X} = \{\{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}\}$, and ranks $\text{rk}(\{a\}) = \text{rk}(\{b\}) = \text{rk}(\{c\}) = 0$, $\text{rk}(\{a, b\}) = \text{rk}(\{b, c\}) = 1$, etc. Thus, $\mathcal{X}^0 = \{\{a\}, \{b\}, \{c\}\}$, $\mathcal{X}^1 = \{\{a, b\}, \{b, c\}\}$. See Figure 3.

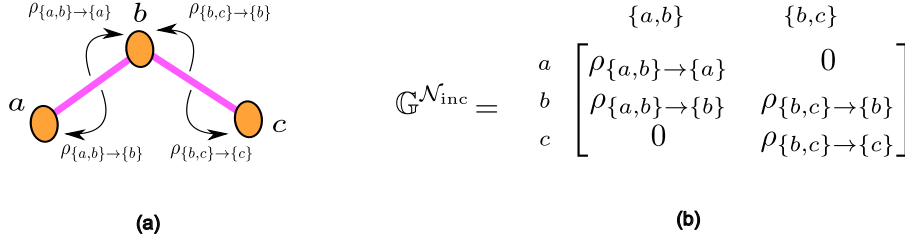


Figure 3: (a) A combinatorial complex $\mathcal{X} = (\mathcal{S}, \mathcal{X}, \text{rk})$ with $\mathcal{S} = \{a, b, c\}$, cells $\mathcal{X} = \{\{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}\}$. The figure also shows the induced directed graph $G_{\mathcal{N}_{\text{inc}}} = (\mathcal{V}_{\mathcal{N}}, E_{\mathcal{N}})$ from the incidence neighborhood structure on the combinatorial complex \mathcal{X} . Each arrow $z \rightarrow y$ represents a directed edge from a 1-cell to a 0-cell where $y \subset z$, and is associated with a linear map $\rho_{z \rightarrow y}$ as part of the copresheaf. (b) The copresheaf neighborhood matrix (CNM) $\mathbb{G}_{\mathcal{N}_{\text{inc}}}$, where rows are indexed by 0-cells $\{a\}, \{b\}, \{c\}$ and columns by 1-cells $\{a, b\}, \{b, c\}$. The matrix entries are linear maps $\rho_{z \rightarrow y}$ when $z \in \mathcal{N}_{\text{inc}}(y)$, and 0 otherwise. This matrix supports directional feature propagation from 1-cells to 0-cells.

Incidence Neighborhood. The incidence neighborhood $\mathcal{N}_{\text{inc}} : \mathcal{X} \rightarrow \mathcal{P}(\mathcal{X})$ is:

$$\mathcal{N}_{\text{inc}}(x) = \{y \in \mathcal{X} \mid x \subset y\}.$$

For 0-cells: $\mathcal{N}_{\text{inc}}(\{a\}) = \{\{a, b\}\}$, $\mathcal{N}_{\text{inc}}(\{b\}) = \{\{a, b\}, \{b, c\}\}$, $\mathcal{N}_{\text{inc}}(\{c\}) = \{\{b, c\}\}$. For 1-cells: $\mathcal{N}_{\text{inc}}(\{a, b\}) = \mathcal{N}_{\text{inc}}(\{b, c\}) = \emptyset$. The effective support is $\mathcal{X}_{\mathcal{N}} = \{\{a\}, \{b\}, \{c\}\}$.

969 **Induced Graph.** We induce a directed graph $G_{\mathcal{N}} = (V_{\mathcal{N}}, E_{\mathcal{N}})$, with:

$$V_{\mathcal{N}} = \mathcal{X}_{\mathcal{N}} \cup \bigcup_{x \in \mathcal{X}} \mathcal{N}_{\text{inc}}(x) = \{\{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}\} = \mathcal{X},$$

970 $E_{\mathcal{N}} = \{y \rightarrow x \mid x \in \mathcal{X}_{\mathcal{N}}, y \in \mathcal{N}_{\text{inc}}(x)\} = \{\{a, b\} \rightarrow \{a\}, \{a, b\} \rightarrow \{b\}, \{b, c\} \rightarrow \{b\}, \{b, c\} \rightarrow \{c\}\}.$

971 **Copresheaf.** The \mathcal{N}_{inc} -dependent copresheaf assigns $\mathcal{F}(x) = \mathbb{R}^2$ to each $x \in V_{\mathcal{N}}$, and linear maps
972 $\rho_{y \rightarrow x} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ for $y \rightarrow x \in E_{\mathcal{N}}$:

$$\rho_{\{a, b\} \rightarrow \{a\}} = \begin{bmatrix} 1 & 0 \\ 0 & 0.5 \end{bmatrix}, \quad \rho_{\{a, b\} \rightarrow \{b\}} = \rho_{\{b, c\} \rightarrow \{b\}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \rho_{\{b, c\} \rightarrow \{c\}} = \begin{bmatrix} 1 & 0 \\ 0 & 0.75 \end{bmatrix}.$$

973 **Neighborhood Matrix.** The CNM $\mathbb{G}^{\mathcal{N}}$ for $\mathcal{Y} = \mathcal{X}^0$, $\mathcal{Z} = \mathcal{X}^1$ is:

$$[\mathbb{G}^{\mathcal{N}}]_{i,j} = \begin{cases} \rho_{z_j \rightarrow y_i} & \text{if } z_j \in \mathcal{N}_{\text{inc}}(y_i), \\ 0 & \text{otherwise.} \end{cases}$$

974 For $\mathcal{Y} = \{\{a\}, \{b\}, \{c\}\}$, $\mathcal{Z} = \{\{a, b\}, \{b, c\}\}$:

$$\mathbb{G}^{\mathcal{N}} = \begin{bmatrix} \rho_{\{a, b\} \rightarrow \{a\}} & 0 \\ \rho_{\{a, b\} \rightarrow \{b\}} & \rho_{\{b, c\} \rightarrow \{b\}} \\ 0 & \rho_{\{b, c\} \rightarrow \{c\}} \end{bmatrix}.$$

975 This matrix facilitates message passing from 1-cells to 0-cells, e.g., bond-to-atom feature propagation.

976 More generally, Figure 4 shows an illustrative example of the general setup of copresheaf higher-order
977 message passing on a CC with multiple neighborhood functions.

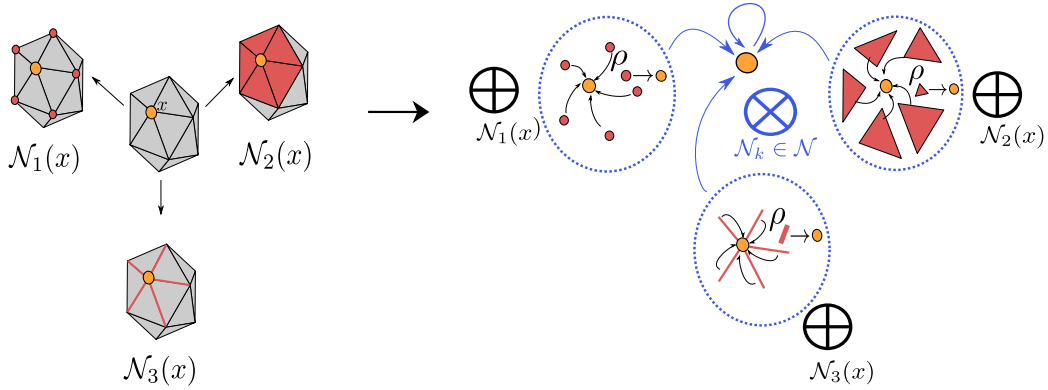


Figure 4: Illustration of copresheaf higher-order message passing. Left-hand Side: Shows a central cell x (as a circle) with arrows pointing to boxes labeled $\mathcal{N}_1(x), \mathcal{N}_2(x), \dots, \mathcal{N}_k(x)$, representing the collection of neighborhood functions $\mathfrak{N} = \{\mathcal{N}_k\}_{k=1}^n$. Right-hand Side: Depicts the message-passing process for the same cell x . Each neighborhood $\mathcal{N}_k(x)$ produces an aggregated message using $\bigoplus_{y \in \mathcal{N}_k(x)} \alpha_{\mathcal{N}_k}(\rho_{y \rightarrow x}(h_y^{(\ell)}))$. These messages are then combined using the inter-neighborhood function β , shown as a box, with an arrow updating x .

978 C Expressive power of CTNNs

979 C.1 Universal Approximation of \mathcal{N} -Dependent Copresheaves

980 Here, we demonstrate that multilayer perceptrons (MLPs) can approximate arbitrary copresheaf
981 morphisms induced by neighborhood functions. This result ensures that the proposed sheaf-based
982 model is sufficiently expressive to capture complex data interactions.

983 **Proposition 5** (Universal Approximation of \mathcal{N} -Dependent Copresheaves). Let \mathcal{X} be a finite combina-
984 torial complex and \mathcal{N} a neighborhood function on \mathcal{X} . Suppose $G_{\mathcal{N}} \rightarrow \mathbf{Vect}_{\mathbb{R}}$ is an \mathcal{N} -dependent
985 copresheaf with stalks $\mathcal{F}^{\mathcal{N}}(x) = \mathbb{R}^d$ and morphisms $\rho_{y \rightarrow x}^{\mathcal{N}}$. Let a feature map

$$h : \mathcal{X} \rightarrow \mathbb{R}^d, \quad x \mapsto \mathbf{h}_x$$

986 be given so that the $2d$ -vectors $(\mathbf{h}_y, \mathbf{h}_x)$ are pairwise distinct for every directed edge $y \rightarrow x$. Define

$$A = \{(\mathbf{h}_y, \mathbf{h}_x) \mid y \rightarrow x\} \subset \mathbb{R}^{2d}, \quad g: A \rightarrow \mathbb{R}^{d \times d}, \quad g(\mathbf{h}_y, \mathbf{h}_x) = \rho_{y \rightarrow x}^{\mathcal{N}}.$$

987 Then for any $\varepsilon > 0$ there exists a multilayer perceptron $\Phi: \mathbb{R}^{2d} \rightarrow \mathbb{R}^{d \times d}$ with sufficiently many
 988 hidden units such that $\|\Phi(\mathbf{h}_y, \mathbf{h}_x) - \rho_{y \rightarrow x}^{\mathcal{N}}\| < \varepsilon$ for all $y \rightarrow x$.

989 *Proof.* Since A is finite and its elements are distinct, the assignment $g: A \rightarrow \mathbb{R}^{d \times d}$ is well-defined.
 990 Enumerate $A = \{a_i\}_{i \in I}$, choose disjoint open neighborhoods $U_i \ni a_i$, and pick smooth ‘‘bump’’
 991 functions

$$\varphi_i: \mathbb{R}^{2d} \rightarrow [0, 1], \quad \varphi_i(a_i) = 1, \quad \text{supp}(\varphi_i) \subset U_i.$$

992 Then the sum

$$f(a) = \sum_{i \in I} g(a_i) \varphi_i(a)$$

993 is a smooth map $f: \mathbb{R}^{2d} \rightarrow \mathbb{R}^{d \times d}$ satisfying $f|_A = g$.

994 Since A is finite, choose a compact set $K \subseteq \mathbb{R}^{2d}$ containing A , ensuring the applicability of the
 995 Universal Approximation Theorem. By that theorem, for any $\varepsilon > 0$ there is an MLP Φ such that

$$\sup_{a \in K} \|\Phi(a) - f(a)\| < \varepsilon.$$

996 In particular, for each $a_i \in A$ we have

$$\|\Phi(a_i) - g(a_i)\| = \|\Phi(\mathbf{h}_y, \mathbf{h}_x) - \rho_{y \rightarrow x}^{\mathcal{N}}\| < \varepsilon,$$

997 for every $y \rightarrow x$. This completes the proof. \square

998 D Sheaf Neural Networks Are Copresheaf Message-Passing Neural Networks

999 In this appendix, we prove Theorem 1 and Proposition 2, which demonstrate that existing Sheaf
 1000 Neural Networks (SNNs), including sheaf diffusion networks, are special cases of the Copresheaf
 1001 Message-Passing Neural Network (CMPNN) framework (Definition 1). Moreover, we summarize
 1002 how other sheaf-based neural architectures align with our unifying message-passing framework (see
 1003 Table 7).

1004 *Proof of Theorem 1.* We construct the bidirected graph G' and define the copresheaf \mathcal{G} on it, then
 1005 demonstrate the equivalence of the message-passing operations.

1006 First, construct the bidirected graph $G' = (V, E')$ from $G = (V, E)$ by replacing each undirected
 1007 edge $\{u, v\} \in E$ with two directed edges $(u, v), (v, u) \in E'$. This ensures that G' retains the
 1008 connectivity of G while introducing explicit directionality.

1009 Next, define the copresheaf $\mathcal{G}: G' \rightarrow \mathbf{Vect}_{\mathbb{R}}$ as follows:

- 1010 • For each vertex $x \in V$, assign $\mathcal{G}(x) = \mathcal{F}(x)$, where $\mathcal{F}(x)$ is the vector space associated to
 1011 x by the cellular sheaf \mathcal{F} .
- 1012 • For each directed edge $(v, u) \in E'$, corresponding to the undirected edge $e = \{u, v\} \in$
 1013 E , define the linear morphism $\rho_{v \rightarrow u}: \mathcal{G}(v) \rightarrow \mathcal{G}(u)$ by $\rho_{v \rightarrow u} = \mathcal{F}_{u \triangleleft e}^{\top} \circ \mathcal{F}_{v \triangleleft e}$, where
 1014 $\mathcal{F}_{v \triangleleft e}: \mathcal{F}(v) \rightarrow \mathcal{F}(e)$ and $\mathcal{F}_{u \triangleleft e}: \mathcal{F}(u) \rightarrow \mathcal{F}(e)$ are the restriction maps of \mathcal{F} , and
 1015 $\mathcal{F}_{u \triangleleft e}^{\top}: \mathcal{F}(e) \rightarrow \mathcal{F}(u)$ is the adjoint with respect to an inner product on $\mathcal{F}(e)$.

1016 Now, consider the SNN message-passing mechanism along the edge $e = \{u, v\}$. For a feature vector
 1017 $\mathbf{h}_v \in \mathcal{F}(v)$ at vertex v , the message transmitted to vertex u is given by $\mathcal{F}_{u \triangleleft e}^{\top} \circ \mathcal{F}_{v \triangleleft e}(\mathbf{h}_v)$.

1018 In the copresheaf \mathcal{G} on G' , the morphism associated with the directed edge (v, u) is $\rho_{v \rightarrow u} =$
 1019 $\mathcal{F}_{u \triangleleft e}^{\top} \circ \mathcal{F}_{v \triangleleft e}$. Applying this morphism, the message-passing operation in the Copresheaf Message-
 1020 Passing Neural Network (CMPNN) yields $\rho_{v \rightarrow u}(\mathbf{h}_v) = \mathcal{F}_{u \triangleleft e}^{\top} \circ \mathcal{F}_{v \triangleleft e}(\mathbf{h}_v)$, which is identical to the
 1021 SNN message.

Method (Paper)	Message Passing Equation	Notable Features	Restriction Map $\rho_{y \rightarrow x}$
Sheaf Neural Network (SNN) Hansen & Gebhart (2020)	$\mathbf{h}_x^{(l+1)} = \sigma \left(\mathbf{h}_x^{(l)} + \sum_{y \in \mathcal{N}(x)} \rho_{y \rightarrow x} \mathbf{h}_y^{(l)} \right)$	Linear restriction maps $\rho_{y \rightarrow x}$ assigned per edge; enables high-dimensional, direction-aware message passing via sheaf structure.	$\rho_{y \rightarrow x} = \mathcal{F}_{x \rightarrow y}^\top \mathcal{F}_{y \rightarrow x}$, fixed linear map, $e = \{x, y\}$
Neural Sheaf Diffusion (NSD) Bodnar et al. (2022)	$\mathbf{h}_x^{(l+1)} = \mathbf{h}_x^{(l)} - \sigma \left(\sum_{x \in e} \rho_{x \rightarrow x} \mathbf{h}_x^{(l)} - \sum_{y \in \mathcal{N}(x)} \rho_{y \rightarrow x} \mathbf{h}_y^{(l)} \right)$	Diffusion over learned sheaf Laplacian; restriction maps $\rho_{y \rightarrow x}$ are learnable, reflecting edge-mediated interactions.	$\rho_{y \rightarrow x} = \mathcal{F}_{x \rightarrow y}^\top \mathcal{F}_{y \rightarrow x}$, learned linear map, $e = \{x, y\}$
Sheaf Attention Network (SAN) Barbero et al. (2022)	$\mathbf{h}_x^{(l+1)} = \sigma \left(\sum_{y \in \mathcal{N}(x)} \alpha_{xy}(\mathbf{h}_x, \mathbf{h}_y) \rho_{y \rightarrow x} \mathbf{h}_y^{(l)} \right)$	Attentional sheaf: attention weights α_{xy} modulate the restricted neighbor feature; mitigates oversmoothing in GAT-style setups.	$\rho_{y \rightarrow x}$: learned linear map, parameterized to capture feature space relationships
Connection Laplacian SNN Barbero et al. (2022)	$\mathbf{h}_x^{(l+1)} = \sigma \left(\mathbf{h}_x^{(l)} + \sum_{y \in \mathcal{N}(x)} O_{xy} \mathbf{h}_y^{(l)} \right)$	Edge maps O_{xy} are orthonormal, derived from feature space alignment; reduces learnable parameters and reflects local geometric priors.	O_{xy} : orthonormal matrix, learned to align feature spaces across edges
Heterogeneous Sheaf Neural Network (HetSheaf) Braithwaite et al. (2024)	$\mathbf{h}_x^{(l+1)} = \sigma \left(\mathbf{h}_x^{(l)} + \sum_{y \in \mathcal{N}(x)} \rho_{y \rightarrow x}(\mathbf{h}_x, \mathbf{h}_y) \mathbf{h}_y^{(l)} \right)$	Type-aware sheaf morphisms: $\rho_{y \rightarrow x}$ depend on node and edge types, enabling structured heterogeneity across the graph.	$\rho_{y \rightarrow x}$: type-aware learned linear map, parameterized by node and edge types
Adaptive Sheaf Diffusion Zaghen et al. (2024)	$\mathbf{h}_x^{(l+1)} = \mathbf{h}_x^{(l)} + \sigma \left(\sum_{y \in \mathcal{N}(x)} \rho_{y \rightarrow x}(\mathbf{h}_x, \mathbf{h}_y) (\mathbf{h}_y^{(l)} - \mathbf{h}_x^{(l)}) \right)$	Nonlinear Laplacian-like dynamics with adaptive, feature-aware restriction maps $\rho_{y \rightarrow x}$; enhances expressiveness and locality.	$\rho_{y \rightarrow x}$: feature-aware learned linear map, parameterized by node features

Table 7: Unified message passing formulations of various sheaf neural networks using HOMP notation. The restriction maps $\rho_{y \rightarrow x}$ may be linear, data-driven, or attentional depending on the model.

To ensure that \mathcal{G} is a well-defined copresheaf, observe that it assigns vector spaces to vertices and linear maps to directed edges in a functorial manner. Specifically, for each directed edge $(v, u) \in E'$, the map $\rho_{v \rightarrow u}$ is a composition of linear maps and thus linear. The identity and composition properties are satisfied implicitly through the consistency of the sheaf restriction maps.

Therefore, the SNN message passing, which operates via intermediate edge spaces in \mathcal{F} , is equivalently represented as direct vertex-to-vertex message passing in the copresheaf \mathcal{G} on G' . This completes the proof. \square

Proof of Proposition 2 Compute:

$$\begin{aligned}
(I_n \otimes W_1)\mathbf{H} &= [W_1 \mathbf{h}_v]_{v \in V}, \\
(\Delta_{\mathcal{F}} \otimes I)(I_n \otimes W_1)\mathbf{H}W_2 &= \left[\sum_{u \in V} L_{F,v,u} W_1 \mathbf{h}_u W_2 \right]_{v \in V}, \\
\mathbf{h}_v^+ &= \mathbf{h}_v - \sum_{u \in \mathcal{N}(v) \cup \{v\}} W_2 L_{F,v,u} W_1 \mathbf{h}_u,
\end{aligned}$$

since $L_{F,v,u} = 0$ for $u \notin \mathcal{N}(v) \cup \{v\}$.

Interpret G as a directed graph with edges $u \rightarrow v$ for $u \in \mathcal{N}(v)$ and $v \rightarrow v$. Define: - Message function: $\alpha(\mathbf{h}_v, \rho_{u \rightarrow v} \mathbf{h}_u) = W_2 L_{F,v,u} W_1 \mathbf{h}_u$, - Morphisms: $\rho_{u \rightarrow v}$ implicitly encoded via $L_{F,v,u}$, - Aggregator: $\oplus = \sum$, - Update: $\beta(\mathbf{h}_v, m) = \mathbf{h}_v - m$.

Thus:

$$\mathbf{h}_v^+ = \beta \left(\mathbf{h}_v, \sum_{u \rightarrow v} \alpha(\mathbf{h}_v, \rho_{u \rightarrow v} \mathbf{h}_u) \right),$$

matching Definition 1. \square

Table 7 provides a summary of sheaf neural networks realized in terms of Definition 1.

E General Copresheaf-Based Transformer Layer

The main idea to introduce a copresheaf structure to the transformer is the following. For every ordered pair $y \rightarrow x$ (within an attention head) we define a parametrized =copresheaf map

$$\rho_{y \rightarrow x} : \mathbb{R}^d \longrightarrow \mathbb{R}^d, \quad v_y \longmapsto \rho_{y \rightarrow x} v_y,$$

1042 transports the value vector from the stalk at y to the stalk at x . Given attention weights $\alpha_{xy} =$
 1043 $\text{softmax}_{y \in \mathcal{N}(x)}((q_x^\top k_y)/\sqrt{d})$, the head message is $m_x = \sum_{y \in \mathcal{N}(x)} \alpha_{xy} \rho_{y \rightarrow x} v_y$.

1044 In Alg. [I](#), we provide the pseudocode for our generic copresheaf-based transformer layer. This
 1045 algorithm outlines the layer-wise update rule combining self-attention within cells of equal rank and
 1046 cross-attention between different ranks, using learned copresheaf morphisms to transfer features
 1047 between stalks. It generalizes standard transformer mechanisms by introducing neighborhood-
 1048 dependent transformations.

General Copresheaf-Based Transformer Layer

```

1: procedure SHEAFTRANSFORMERLAYER( $\mathcal{X}, \mathcal{N}, \{h_x^{(\ell)} \in \mathcal{F}(x)\}_{x \in \mathcal{X}}$ )
2:   for  $x \in \mathcal{X}^k$  do                                      $\triangleright$  Self-attention on  $k$ -cells
3:      $q_x \leftarrow W_q h_x^{(\ell)}, k_x \leftarrow W_k h_x^{(\ell)}, v_x \leftarrow W_v h_x^{(\ell)}$ 
4:      $m_x \leftarrow 0$ 
5:     for  $y \in \mathcal{N}_k(x)$  do
6:        $\alpha_{xy} \leftarrow \text{softmax}_{\mathcal{N}_k(x)}(\langle q_x, k_y \rangle / \sqrt{p})$ 
7:        $\tilde{v}_{xy} \leftarrow \rho_{y \rightarrow x}(v_y)$ 
8:        $m_x \leftarrow m_x + \alpha_{xy} \tilde{v}_{xy}$ 
9:     end for
10:     $h_x^{(\ell+1)} \leftarrow \beta(h_x^{(\ell)}, m_x)$ 
11:  end for
12:  for  $x \in \mathcal{X}^{k_t}$  do                                      $\triangleright$  Cross-attention from  $k_s$  to  $k_t$ 
13:     $q_x \leftarrow W_q^{s \rightarrow t} h_x^{(\ell)}$ 
14:     $m_x \leftarrow 0$ 
15:    for  $y \in \mathcal{N}_{s \rightarrow t}(x)$  do
16:       $k_y \leftarrow W_k^{s \rightarrow t} h_y^{(\ell)}, v_y \leftarrow W_v^{s \rightarrow t} h_y^{(\ell)}$ 
17:       $\alpha_{xy} \leftarrow \text{softmax}_{\mathcal{N}_{s \rightarrow t}(x)}(\langle q_x, k_y \rangle / \sqrt{p})$ 
18:       $\tilde{v}_{xy} \leftarrow \rho_{y \rightarrow x}(v_y)$ 
19:       $m_x \leftarrow m_x + \alpha_{xy} \tilde{v}_{xy}$ 
20:    end for
21:     $h_x^{(\ell+1)} \leftarrow \beta(h_x^{(\ell)}, m_x)$ 
22:  end for
23:  return  $\{h_x^{(\ell+1)}\}_{x \in \mathcal{X}}$ 
24: end procedure

```

Algorithm 1: Copresheaf Transformer layer integrating standard attention with learned copresheaf morphisms.

1049

1050 F Copresheaf Learning on Euclidean Data

1051 The CopresheafConv layer leverages copresheaf structures to process data on a D -dimensional grid
 1052 $\mathcal{X} \subset \mathbb{Z}^D$, offering distinct advantages over traditional convolutional neural networks (CNNs). By
 1053 defining a copresheaf on a combinatorial complex (CC) constructed from the grid, where cells
 1054 represent grid points (0-cells) and their pairwise connections (1-cells), the layer employs learnable
 1055 morphisms $\rho_{y \rightarrow x} : \mathcal{F}(y) \rightarrow \mathcal{F}(x)$ that dynamically adapt to directional relationships between
 1056 points. Unlike static convolutional filters, these morphisms capture anisotropic, directionally de-
 1057 pendent interactions, preserving topological nuances of the grid’s geometry. In contrast, regular
 1058 convolutional kernels enforce translation invariance, limiting their ability to model spatially vary-
 1059 ing or directional patterns. The copresheaf is defined over an adjacency neighborhood function
 1060 $\mathcal{N}_{\text{adj}}(x) = \{y \in \mathcal{X} \mid \{x, y\} \in \mathcal{X}^1\}$, restricting computation to local, grid-adjacent neighbors, thus
 1061 ensuring efficiency comparable to CNNs. The morphisms, potentially nonlinear, are conditioned on
 1062 input features $\mathbf{h}_x^{(\ell)}, \mathbf{h}_y^{(\ell)}$ and grid positions, enabling the layer to model complex, multi-scale depen-
 1063 dencies. This makes CopresheafConv ideal for tasks like image segmentation, 3D mesh processing,
 1064 or geometric deep learning, where local and hierarchical relationships are critical. Empirical results
 1065 demonstrate superior performance in capturing physical dynamics, underscoring its ability to handle
 1066 spatially varying patterns. Algorithm [F](#) shows the pseudocode for the CopresheafConv used in our
 1067 experiments.

CopresheafConv on a D -Dimensional Grid

```

1: procedure COPRESHEAFCONV( $\mathcal{X} \subset \mathbf{Z}^D$ ,  $\{h_x^{(\ell)} \in \mathcal{F}(x) = \mathbb{R}^{C_{\text{in}}}\}_{x \in \mathcal{X}}$ )
2:   for  $x \in \mathcal{X}$  do
3:      $m_x \leftarrow 0 \in \mathbb{R}^{C_{\text{out}}}$ 
4:     for  $y \in \mathcal{N}(x)$  do
5:        $\rho_{y \rightarrow x} \leftarrow \text{CopresheafMorphism}(y, x)$   $\triangleright$  map conditioned on  $h_x^{(\ell)}, h_y^{(\ell)}$  (and thus on
         $x, y$ )
6:        $m_x \leftarrow m_x + \rho_{y \rightarrow x}(h_y^{(\ell)})$ 
7:     end for
8:      $h_x^{(\ell+1)} \leftarrow m_x$ 
9:   end for
10:  return  $\{h_x^{(\ell+1)}\}_{x \in \mathcal{X}}$ 
11: end procedure
12: procedure COPRESHEAFMORPHISM( $y, x$ )  $\triangleright$  Return the learned copresheaf morphism  $\rho_{y \rightarrow x}$ ,
    potentially nonlinear,  $\triangleright$  conditioned on both source and target features  $(h_x^{(\ell)}, h_y^{(\ell)})$ .
13:  return  $\rho_{y \rightarrow x}$ 
14: end procedure

```

1068

G Experiments

1069

1070 We now provide further evaluations on additional real and synthetic datasets. We start with a synthetic
1071 timeseries dataset where.... We then introduce a structure recognition benchmark of oriented ellipses,
1072 hierarchical triangles and hierarchical polygons. We then put our transformer models to test at airfoil
1073 self-noise regression task as an important industrial application.

G.1 Synthetic Control Tasks

1074

1075 Six canonical univariate time-series patterns—*normal*, *cyclic*, *increasing trend*, *decreasing trend*,
1076 *upward shift*, *downward shift*—are procedurally generated. We obtain 600 sequences of length 60
1077 (100 per class), normalised to the interval $[-1, 1]$, with an 80:20 split for training and test.

1078 **Models & set-up.** A lightweight vanilla Trans-
1079 former (32-d model, 4 heads, 2 layers) is compared
1080 with an identically sized Copresheaf Transformer,
1081 where multi-head attention is replaced by a gated
1082 outer-product tensor-attention layer with orthogonal-
1083 ity ($\lambda = 0.01$) and sparsity ($\lambda = 10^{-4}$) regularisers.

1084 Both models share sinusoidal-with-linear-decay positional encodings, use Adam (10^{-3}), batch 32,
1085 train for 15 epochs, and each experiment is repeated with three random seeds.

1086 **Results.** As seen in Table 8 the Copresheaf Transformer yields a consistent improvement of +0.8–1.0
1087 percentage points (pp) over the vanilla Transformer while remaining lightweight and training in
1088 comparable wall-clock time (< 1 min/run on a single GPU), highlighting the benefit of richer token-
1089 pair transformations for recognition tasks.

G.1.1 Structure Recognition datasets

1090

1091 In this experiment we consider two synthetic image datasets containing oriented ellipses or hierarchi-
1092 cal triangles.

1093 **Dataset (Oriented Ellipses).** Each 32×32 RGB image contains a single black ellipse on a white
1094 background. The horizontal and vertical *semi-axes* a, b are drawn uniformly from $\{4, 5, \dots, 12\}$
1095 pixels, and the ellipse is rotated by a random angle in $[0, 180^\circ)$. The task is to predict the coarse ori-
1096 entation bin (4 bins of 45° each). We synthesise 6,000 images, keep 5,000:1,000 for train/validation,
1097 and rescale pixels to $[-1, 1]$.

1098 **Dataset (Hierarchical Triangles).** Each 32×32 RGB image contains six coloured circles—*red*,
1099 *green*, *blue*, *yellow*, *cyan*, *magenta*—placed on two nested equilateral triangles (inner radius 8 px,
1100 outer 12 px). Colours are randomly permuted. A hand-crafted hierarchy of linear maps (inner-triangle,

outer-triangle, cross-level) is applied to the circles’ one-hot colour vectors; the image is labelled 1 when the resulting scalar exceeds a fixed threshold, else 0. We generate 6,000 images and keep 5,000:1,000 for train/validation.

Models & set-up (shared). Both tasks use the same compact Vision-Transformer backbone: 32-dim patch embeddings (patch size 8), 4 heads, 2 layers, learnable positional embeddings, AdamW (3×10^{-4}). The baseline is a **Regular ViT**; its counterpart is an identically sized **Copresheaf ViT** in which multi-head attention is replaced by an outer-product copresheaf mechanism (stalk-dim = 8). Oriented Ellipses is trained with batch 128; Hierarchical Triangles with batch 64. All runs use 30 epochs and three independent seeds.

Results. Across both synthetic vision tasks the Copresheaf ViT consistently surpasses the Regular ViT: a dramatic +12.1 pp gain on Oriented Ellipses and a subtler yet statistically tighter +1.4 pp on Hierarchical Triangles, while also cutting variance by an order of magnitude in the latter case (see Table 9). These outcomes underscore that replacing standard attention with copresheaf-guided outer-product maps yields robust improvements for both low-level geometric orientation recognition and higher-level nested-structure reasoning, all without a significant increase of model size or training budget.

Table 9: Validation accuracy on both synthetic vision tasks (mean \pm std over three seeds).

Dataset	Regular ViT	Copresheaf ViT
Oriented Ellipses	84.13 ± 4.12	96.23 ± 0.33
Hierarchical Triang.	95.47 ± 1.31	96.87 ± 0.26

G.1.2 Classifying Hierarchical Polygons

Similar to the previous section, we now synthesize a hierarchy of nested regular polygons. In particular, each 32×32 RGB image contains a variable number $n \in \{6, 8, 10\}$ of coloured circles arranged on two nested regular polygons (inner radius 8 px, outer 12 px). The first $n/2$ circles form the inner polygon, the remainder the outer; colours are drawn from a palette of n distinct hues and randomised per sample. A hierarchy of hand-crafted linear maps is applied to one-hot colour vectors: pairwise maps on the inner polygon (F_{inner}), on the outer polygon (G_{outer}), and a cross-level map H . The image is labelled 1 when the resulting scalar exceeds a threshold, else 0. For each n we synthesise 6,000 images, keep 5,000:1,000 for train/validation, and normalise pixels to $[-1, 1]$.

Models & training. We reuse the compact ViT backbone (32-dim patch embeddings, patch size 8, 4 heads, 2 layers, learnable positional embeddings). The **Regular ViT** is compared with an identically sized **Copresheaf ViT**, which replaces multi-head attention with rank-restricted copresheaf outer-product maps (stalk-dim = 8). Both networks are trained for 10 epochs with AdamW (3×10^{-4}), batch 64; each configuration is run three times.

Results. Figure 5 shows that the Copresheaf ViT consistently exceeds the Regular ViT at $n=6$ (0.72 vs 0.66) and regains a clear lead at $n=10$ (0.63 vs 0.57) despite both models dipping at $n=8$. The Copresheaf curve displays narrower uncertainty bands at the hardest setting, indicating greater run-to-run stability. Overall, copresheaf-guided attention scales more gracefully with combinatorial complexity, capturing cross-level dependencies that standard self-attention struggles to model.

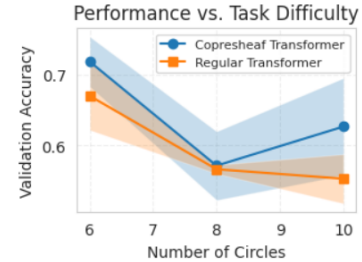


Figure 5: Validation accuracy (mean \pm 1 s.d., 3 runs) as task difficulty increases.

G.1.3 Airfoil Self-Noise Regression

The UCI airfoil dataset (1 503 rows) maps five continuous descriptors—frequency, angle of attack, chord length, free-stream velocity, Reynolds number—to the sound-pressure level (dB). Inputs and target are min-max scaled to $[0, 1]$; we keep only 400:100 train/test samples for a low-data setting.

Table 10: Test MSE (mean \pm std over two runs).

Model	MSE \downarrow
Regular Transformer	0.0223 ± 0.0001
Copresheaf Transf.	0.0208 ± 0.0002

Models Both regressors share a minimalist backbone: 64-d token embedding \rightarrow 2-layer, 4-head Transformer \rightarrow mean pooling \rightarrow scalar head. The Copresheaf variant swaps dot-product attention for learned outer-product maps ρ_{ij} that depend on each token pair, whereas the Regular baseline keeps standard self-attention. Training uses Adam (10^{-4}), 1000 epochs, batch 32.

Results. On the small 100-sample test set the Copresheaf regressor lowers MSE by 6.7% relative to the regular Transformer and maintains sub- 10^{-4} run-to-run variance (see Table 10), confirming that pair-specific linear transports help model heterogeneous feature interactions even in data-scarce regimes.

G.2 Pixelwise Regression Tasks: Evaluating CopresheafConv2D Layers

We evaluate neural network models incorporating *CopresheafConv2D* layers—custom convolutional layers with patch-wise trainable linear morphisms—against standard convolutional models across four synthetic pixelwise regression tasks: PDE regression (Bratu and convection–diffusion equations), image denoising, distance transform regression, and edge enhancement. In all tasks, Copresheaf-based models consistently achieve lower Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) compared to standard convolutional models, suggesting improved modeling of spatial structures and relationships.

Task Definitions.

• PDE Regression:

- **Bratu Equation:** A nonlinear reaction–diffusion PDE:

$$-\Delta u = g(x, y) e^u, \quad u|_{\partial\Omega} = 0,$$

where $g(x, y)$ is a source intensity.

- **Convection–Diffusion Equation:** A transport PDE:

$$-\nu \Delta u + c_x \partial_x u + c_y \partial_y u = g(x, y), \quad u|_{\partial\Omega} = 0,$$

with diffusion ν and velocities c_x, c_y .

- **Image Denoising:** Recovering clean structured images (sinusoidal patterns with a Gaussian bump, normalized to $[0, 1]$) from Gaussian noise ($\sigma = 0.3$).
- **Distance Transform Regression:** Predicting the normalized Euclidean distance transform of a binary segmentation (thresholded at 0.5) of structured images.
- **Edge Enhancement:** Predicting edge maps from structured images using a difference-of-anisotropic-Gaussians (DoG) transformation.

Model and Training Setup. For PDE regression and distance transform tasks, we use U-Net variants: *CopresheafUNet* (with CopresheafConv2D layers) and *ConvUNet* (with standard Conv2d layers), both with a four-level backbone ($64 \rightarrow 128 \rightarrow 256 \rightarrow 512$ channels). For image denoising and edge enhancement, we use four-layer convolutional networks: *CopresheafNet* and *ConvNet* ($1 \rightarrow 8 \rightarrow 16 \rightarrow 8 \rightarrow 1$ channels). All models are trained on 64×64 inputs using the Adam optimizer and MSE loss, with task-specific settings (learning rates 10^{-3} or 10^{-4} , batch sizes 8 or 16, 80–300 epochs). Results are averaged over 3 random seeds.

Table 11: Mean (\pm std over 3 seeds) of MSE and RMSE across all tasks.

Task	Model	MSE	RMSE
Bratu Equation	CopresheafUNet	0.0001 ± 0.0002	0.0108 ± 0.0003
	ConvUNet	0.0003 ± 0.0002	0.0183 ± 0.0007
Convection–Diffusion	CopresheafUNet	0.0004 ± 0.0001	0.0205 ± 0.0010
	ConvUNet	0.0006 ± 0.0002	0.0232 ± 0.0012
Image Denoising	CopresheafNet	0.0010 ± 0.0001	0.0310 ± 0.0010
	ConvNet	0.0011 ± 0.0002	0.0336 ± 0.0015
Distance Transform	CopresheafUNet	0.0001 ± 0.00002	0.0105 ± 0.0003
	ConvUNet	0.0002 ± 0.00003	0.0156 ± 0.0005
Edge Enhancement	CopresheafNet	0.0008 ± 0.0001	0.0283 ± 0.0010
	ConvNet	0.0009 ± 0.0002	0.0300 ± 0.0015

Take-away. Across all tasks, replacing standard convolutional layers with CopresheafConv2D layers results in lower MSE and RMSE (see Table 11). This consistent improvement suggests that patch-wise linear maps enhance the models’ ability to capture complex spatial patterns. These findings highlight the potential of Copresheaf-based architectures for pixelwise regression problems. Subsequently, we address two challenges related to token classification: real/fake token sequence detection and

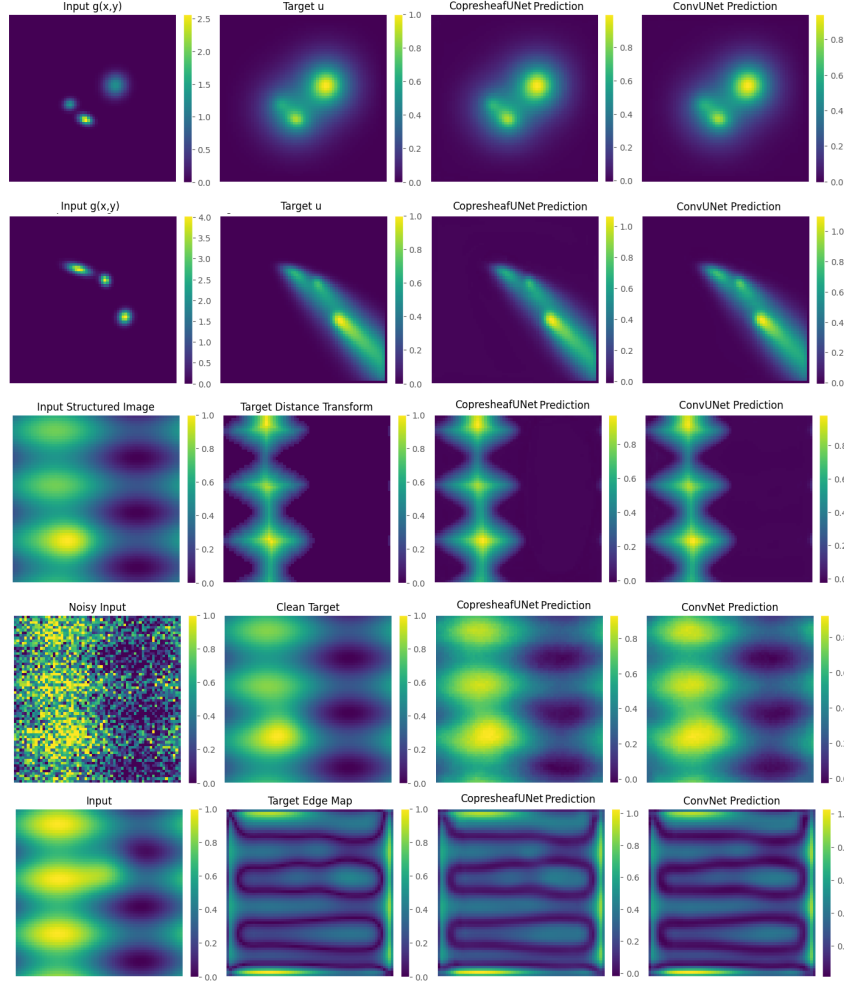


Figure 6: Model outputs across tasks. *A*: Bratu equation—input g , target u , CopresheafUNet vs. ConvUNet predictions. *B*: Convection–diffusion—input g , target u , CopresheafUNet vs. ConvUNet predictions. *C*: Distance transform—input image, target transform, CopresheafUNet vs. ConvUNet predictions. *D*: Image denoising—noisy input, clean target, CopresheafNet vs. ConvNet predictions. *E*: Edge enhancement—input image, target edge map, CopresheafNet vs. ConvNet predictions. Copresheaf-based models show subtle improvements in detail recovery.

segment-wise token classification. Finally, we conduct a preliminary study on shape classification using copresheaf-augmented attention and graph classification a molecular benchmark, MUTAG. These are followed by applications in graph connectivity classification and text classification on TREC coarse label benchmark.

G.2.1 Learning Token-Relations with Copresheaf Attention

We study five problems that differ only in the (non)linear operator *unknown* applied to the first half of a random token sequence (or to a second related sequence). The classifier must decide whether the tail is just a noisy copy (label 0) or a transformed version of the head (label 1).

- **Orthogonal block.** Eight 16-d “head” tokens are either copied (+0.05 noise) or rotated by a sample-specific orthogonal matrix before adding the same noise.
- **Per-token scaling.** As above, but the tail is $\alpha_i x_i + \text{noise}$ with $\alpha_i \sim \text{U}[0.4, 1.6]$.
- **Rotated copy (embedded 2-D).** Six 2-D points are mapped to 16 d by a fixed linear embed, duplicated to a 12-token sequence; the tail is either a noisy copy or the points after a random planar rotation.

- **Query \rightarrow context linearity.** Two parallel sequences (50×16 “query”, 50×24 “context”). Class 0: context is a global affine transform of the query with partly correlated semantics. Class 1: context comes from a quadratic warp and weak semantic correlation.
- **Affine vs. Quadratic Token Relations .** Two parallel sequences (length-6, query dim 16, context dim 24). Class 0: context is a linear spatial transformation (rotation + translation) of the query plus correlated semantic noise. Class 1: context is generated via a spatial quadratic (nonlinear) transformation with weaker semantic correlation.

Data. Tasks 1–2 use 16 tokens, task 3 uses 12, task 4 uses two length-50 sequences, task 5 uses two length-6 sequences. For each of three seeds we draw 4,096:1,024 train/test sequences (task 4–5: 320:80).

Backbone & training. A tiny Transformer encoder (4 heads, token dim 16, stalk-dim 4) \rightarrow mean-pool \rightarrow 2-way classifier. *Classic* uses vanilla dot-product attention; *Copresheaf* augments it with learned token-pair copresheaf maps (we chose General Copresheaf for the first two tasks, and Non-linear MLP for tasks 3–5). We train for 8 / 12 / 10 / 10 / 10 epochs respectively with Adam (10^{-3}), bat

Take-away. Across in-sequence, element-wise, embedded-geometric, and cross-sequence settings—including varying degrees of spatial and semantic correlation—injecting copresheaf transports into self-attention consistently lifts accuracy significantly (up to +38 pp, as seen in Table 12). This highlights a general principle: tasks whose signals reside in *relations between tokens* rather than in absolute token content strongly benefit from explicitly modeling these relations through learnable copresheaf-induced attention.

Limited Attention Capacity. We study the impact of attention capacity on relational reasoning, by varying the number of heads in a small transformer and testing its ability to classify Query \rightarrow context dataset provided in Section G.2.1.

We evaluate three setups: a baseline transformer, the same model augmented with positional encoding (PE), and a copresheaf-augmented transformer with 2 heads. Figure 7 shows accuracy as a function of attention capacity.

While positional encoding improves baseline accuracy slightly, the copresheaf-augmented attention with just two heads outperforms all classic models, even those with eight times more heads. This highlights the value of inductive relational structure over brute-force capacity scaling.

G.2.2 Segment-wise Token Classification

We test whether copresheaf attention improves token-level classification in a sequence partitioned into contiguous segments with distinct patterns. The classifier must assign a segment label (0, 1, or 2) to each token based on its local context.

Data.

Each input is a sequence of 100 tokens, where each token is a 16-dimensional feature vector. The sequence is divided into three contiguous segments, each following a different pattern: (i) a cosine oscillation, (ii) a linearly increasing ramp, or (iii) an exponentially decreasing signal, with additive noise. The task is to predict the correct segment label for each token. We generate 300 training sequences and evaluate on three random seeds.

Table 12: Mean accuracy (\pm std, 3 seeds).

Task	Classic	Copresheaf
Orthogonal block	0.732 ± 0.009	0.928 ± 0.007
Per-token scaling	0.521 ± 0.005	0.707 ± 0.004
Rotated copy (2-D)	0.739 ± 0.010	0.896 ± 0.033
Query \rightarrow context	0.608 ± 0.046	0.992 ± 0.012
Affine vs. Quadratic Relations	0.588 ± 0.047	0.900 ± 0.027

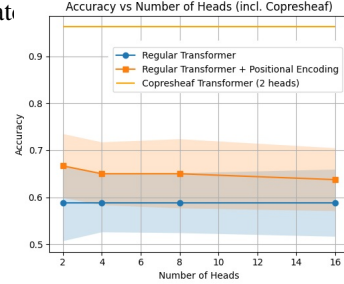


Figure 7: Accuracy as a function of number of attention heads. Even with low capacity, the copresheaf-augmented model perfect generalization of the Query to Context task.

Table 13: Mean segmentation accuracy (\pm std, 3 seeds).

Model	Accuracy
Classic	0.705 ± 0.010
Copresheaf-FC	0.833 ± 0.015
Copresheaf-MLP	0.831 ± 0.007
Copresheaf-SPD	0.743 ± 0.017

Backbone & training We use a 2-layer encoder with 4 heads, token dim 16, and stalk-dim 4. A linear classifier maps each token to one of 3 segment labels. *Classic* is standard attention; *Copresheaf* augments each attention head with learned per-token transport maps. We train for 10 epochs using Adam (1e-3), with batch size 32.

Take-away. Injecting copresheaf structure into attention substantially improves token-wise segmentation accuracy, especially with expressive MLP kernels (see Table 13). This demonstrates that local consistency constraints enforced by per-token transport maps help resolve semantic boundaries even in noisy, positionally ambiguous settings.

G.2.3 Topological Shape Classification

We evaluate copresheaf-augmented attention on a synthetic 3D point cloud classification task. Each input is a set of 128 points in \mathbb{R}^3 sampled from one of four classes: cube, sphere, torus, and twisted torus. Rotations are applied to remove alignment bias.

Data. The dataset consists of 480:160 train/test samples, balanced across the four classes. Each point cloud is processed as a sequence of 128 points with 3D coordinates.

Backbone & training. Both models use a 4-layer point transformer with 4 heads and head dimension 32. The *Classic* model uses standard self-attention. The *Copresheaf* model augments attention with diagonal copresheaf morphisms. We train each model for 50 epochs using AdamW and cosine learning rate decay across 3 random seeds.

Table 14: Mean accuracy (\pm std, 3 seeds).

Model	Accuracy
Classic	0.708 ± 0.031
Copresheaf	0.746 ± 0.034

Take-away. Copresheaf-augmented attention improves accuracy on 3D shape classification by enhancing sensitivity to latent geometric structure (see Table 14).

G.3 TREC Text Classification Task

We evaluate two transformer-based models on the TREC coarse-label question classification task, which involves categorizing questions into 6 classes (e.g., abbreviation, entity, description). The models are: *Classic*, a standard transformer with multi-head self-attention; and *Copresheaf-FC*, which incorporates a GeneralSheafLearner to model stalk transformations.

Task Definition.. The TREC dataset consists of questions labeled with one of 6 coarse categories. Inputs are tokenized questions truncated to 16 tokens, mapped to a vocabulary of size $|\mathcal{V}|$, and embedded into an 8-dimensional space. The task is to predict the correct class label for each question.

Model and Training Setup. Both models use a single transformer block with 2 attention heads, an embedding dimension of 8, and a stalk dimension of 4 for the Copresheaf-based model. The architecture includes an embedding layer, a transformer block with attention and feed-forward components, adaptive average pooling, and a linear classifier. Compared to state-of-the-art (SOTA) models, which often employ multiple transformer layers and high-dimensional embeddings for maximal performance, our networks

Table 15: Mean (\pm std over 4 seeds) of test accuracy for the TREC classification task.

Model	Test Accuracy
Classic	0.7320 ± 0.0080
Copresheaf-FC	0.7500 ± 0.0150

are intentionally small, using a single block and low embedding dimension to prioritize computational efficiency and controlled experimentation. We train on the TREC training set (5452 samples) over 30 epochs with a batch size of 32, using the Adam optimizer with a learning rate of 10^{-3} and cross-entropy loss. The test set (500 samples) is used for evaluation. Each experiment is repeated over 4 random seeds. As seen in Table 15, the copresheaf models outperform their SOTA counterparts.

G.4 Catalogue of copresheaf maps

We also provide the table of copresheaf maps that we used throughout our experiments in Table 16.

Map family	Copresheaf map $\rho_{y \rightarrow x}$ (per head)	Learnable params
General Copresheaf	$\rho_{y \rightarrow x} = \tanh\left(W \begin{bmatrix} q_x \\ k_y \end{bmatrix}\right)$	$W \in \mathbb{R}^{2d \times d^2}$
Pre-Linear Map	$\rho_{y \rightarrow x} = q_x k_y^\top$	none
Diagonal MLP Map	$\rho_{y \rightarrow x} = \text{diag}(\sigma(\text{MLP}[q_x, k_y]))$	2-layer MLP
Graph Attention Map	$\rho_{y \rightarrow x} = \sigma(\text{MLP}[q_x, k_y]) I_d$	2-layer MLP
Vision Spatial Map	$\rho_{y \rightarrow x} = \sigma(\text{MLP}(p_x - p_y))$ ($p_x, p_y \in [0, 1]^2$ pixel coords)	2-layer MLP
Outer-Product Map	$\rho_{y \rightarrow x} = W_q q_x (W_k k_y)^\top$	$W_q, W_k \in \mathbb{R}^{d \times d}$
Non-linear MLP Map	$\rho_{y \rightarrow x} = \text{reshape}(\text{MLP}[q_x, k_y])$	2-layer MLP ($2d \rightarrow 2d \rightarrow d^2$)
Gaussian RBF Map	$\rho_{y \rightarrow x} = e^{-\ q_x - k_y\ ^2 / 2\sigma^2} I_d$	σ (scalar)
Dynamic Map	$\rho_{y \rightarrow x} = \text{reshape}(W_f q_x)$	$W_f \in \mathbb{R}^{d \times d^2}$
Bilinear Map	$\rho_{y \rightarrow x} = (b^\top (q_x, k_y)) I_d$	$b \in \mathbb{R}^{d \times d}$
SheafFC Map	$\rho_{y \rightarrow x} = I_d + \tanh\left(W \begin{bmatrix} q_x \\ k_y \end{bmatrix}\right)$	$W \in \mathbb{R}^{2d \times d^2}$ (zero init)
SheafMLP Map	$\rho_{y \rightarrow x} = I_d + \tanh(\text{MLP}[q_x, k_y])$	2-layer MLP (last layer zero init)
SheafSPD Map	$\rho_{y \rightarrow x} = I_d + QQ^\top, \quad Q = W \begin{bmatrix} q_x \\ k_y \end{bmatrix}$	$W \in \mathbb{R}^{2d \times d^2}$ (no bias)

Table 16: Catalogue of copresheaf maps $\rho_{y \rightarrow x}$ used in our training our copresheaf transformer model. All maps act stalk-wise and are evaluated independently for each attention head; σ is the logistic function.

H Extended Related Work on Topological and Sheaf Neural Networks

Foundations of Sheaf Theory. Sheaf theory offers a unifying categorical framework across algebraic geometry, topology, and algebra [Bredon [1997]]. Early computer-science applications exploited its logical structure [Fourman et al. [1977], Goguen [1992]], and Srinivas generalized pattern matching via sheaves on Grothendieck topologies [Srinivas [1993]], later extended to NP-hard problems [Conghaile [2022], Abramsky [2022]]. Cellular sheaves, formalized in [Curry [2014]], underpin discrete topological data analysis and signal processing [Ghrist and Hiraoka [2011], Robinson [2014]]. Hansen & Ghrist introduced the sheaf Laplacian [Hansen and Ghrist [2019c]], learnable by convex optimization [Hansen and Ghrist [2019a]]. Connection sheaves model discrete vector bundles [Singer and Wu [2012]] and support manifold learning and Gaussian processes [Peach et al. [2024]].

Higher-Order Representations in Deep Learning. The growing interest in higher-order network models [Mendel [1991], Battiston et al. [2020], Bick et al. [2021]] has catalyzed geometric and topological deep learning. Techniques include simplicial, hypergraph, and cellular message-passing schemes [Gilmer et al. [2017], Ebli et al. [2020], Hayhoe et al. [2022], Hajij et al. [2020], Bunch et al. [2020]], skip connections [Hajij et al. [2022]], and convolutional operators [Jiang et al. [2019], Feng et al. [2019]].

Sheaf Neural Networks. In recent years, sheaf-based generalizations of graph neural networks (GNNs) have demonstrated notable improvements on tasks involving heterogeneous or non-Euclidean data. Hansen and Gebhart [Hansen and Gebhart [2020b]] first introduced sheaf neural networks (SNNs), which generalize graph neural networks (GNNs) by replacing neighborhood aggregation with learnable linear “restriction maps”, thereby customizing information flow between nodes. By allowing each edge to carry its own linear transformation, SNNs capture relationships in heterophilic graphs more effectively than degree-normalized convolutions. Building on this idea, Bodnar et al. [Bodnar et al. [2022b]] proposed Neural Sheaf Diffusion (NSD), which jointly learns both the underlying sheaf structure and the diffusion dynamics. NSD layers adaptively infer the sheaf Laplacian from data, mitigating the oversmoothing problem common in deep GNNs and achieving superior performance on a range of heterophilic benchmark datasets. Barbero et al. [Barbero et al. [2022c]] then combined NSD’s principled diffusion with attention mechanisms to formulate Sheaf

1328 Attention Networks (SANs). SANs modulate self-attention weights by the learned sheaf maps,
1329 preserving long-range dependencies while respecting local sheaf geometry.

1330 Alternative formulations include Bundle Neural Networks (BNNs) by Bamberger et al. [Bamberger
1331 et al. [2024]], which reinterpret propagation as parallel transport in a flat vector bundle rather than
1332 discrete message passing. Duta et al. [Duta et al. [2024]] extended sheaf methods to hypergraphs,
1333 defining linear and nonlinear hypergraph Laplacians that capture higher-order interactions among
1334 groups of nodes. On manifold-structured data, Tangent Bundle Neural Networks (TBNNs) proposed
1335 by Battiloro et al. [Battiloro et al. [2024]] treat features as elements of tangent spaces and propagate
1336 them along estimated geodesics, bridging continuous and discrete models.

1337 **Attention Mechanisms in Higher-Order Structures.**

1338 Attention mechanisms have been generalized to hypergraphs [Kim et al. [2020]], [Bai et al. [2021]] and
1339 simplicial complexes [Goh et al. [2022]], [Giusti et al. [2022]], [Battiloro et al. [2023]], among else via
1340 Hodge or Dirac operators. On combinatorial complexes, feature-lifting attention facilitate hierarchical
1341 information propagation [Giusti et al. [2023]], [Hajij et al. [2023b]].

1342 **Applications and Extensions.** These sheaf-theoretic architectures have found diverse applications,
1343 from multi-document summarization [Atri et al. [2023]] and recommendation systems [Purificato et al.
1344 [2023]] to community detection via sheaf cohomology [Wolf and Monod [2023]] and personalized
1345 federated learning with Sheaf HyperNetworks [Nguyen et al. [2024]], [Liang et al. [2024]]. In repre-
1346 sentation learning, many knowledge-graph embedding techniques have been reinterpreted as sheaf
1347 global-section problems [Gebhart et al. [2023]], [Kvinge et al. [2021]]. Together, these advances high-
1348 light the expressive power and flexibility of sheaf-based models in handling complex, heterogeneous,
1349 and higher-order data domains.

1350 **References for Supplementary Material**

1351 Samson Abramsky. Notes on presheaf representations of strategies and cohomological refinements of
1352 k -consistency and k -equivalence. *arXiv preprint arXiv:2206.12156*, 2022.

1353 Yash Atri, Karan Rungta, Lili Mou, Kai-Wei Chang, and Nanyun Joshi. Promoting topic coherence
1354 and inter-document consorts in multi-document summarization via simplicial complex and sheaf
1355 graph. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language*
1356 *Processing*, pages 2154–2166, 2023.

1357 Song Bai, Feihu Zhang, and Philip H. S. Torr. Hypergraph convolution and hypergraph attention.
1358 *Pattern Recognition*, 110:107637, 2021.

1359 Jacob Bamberger, Michael Bronstein, Cristian Bodnar, and Pietro Liò. Bundle neural networks for
1360 message diffusion on graphs. *arXiv preprint arXiv:2405.15540*, 2024.

1361 Federico Barbero, Cristian Bodnar, Aurelien Kazi, Pietro Liò, Guido Montúfar, and Michael Bronstein.
1362 Sheaf attention networks. In *NeurIPS 2022 Workshop on Symmetry and Geometry in Neural*
1363 *Representations*, 2022.

1364 Claudio Battiloro, Lucia Testa, Lorenzo Giusti, Stefania Sardellitti, Paolo Di Lorenzo, and Sergio
1365 Barbarossa. Generalized simplicial attention neural networks. *arXiv preprint arXiv:2309.02138*,
1366 2023.

1367 Claudio Battiloro, Emanuele Rodolà, Pietro Liò, and Michael Bronstein. Tangent bundle convolutional
1368 learning: from manifolds to cellular sheaves and back. *IEEE Transactions on Signal Processing*,
1369 2024.

1370 Federico Battiston, Giulia Cencetti, Iacopo Iacopini, Vito Latora, Maxime Lucas, Alice Patania,
1371 Jean-Gabriel Young, and Giovanni Petri. Networks beyond pairwise interactions: structure and
1372 dynamics. *Physics Reports*, 874:1–92, 2020.

1373 Christian Bick, Elizabeth Gross, Heather A Harrington, and Michael T Schaub. What are higher-order
1374 networks? *arXiv preprint arXiv:2104.11329*, 2021.

1375 Cristian Bodnar, Fabrizio Frasca, Yuguang Wang, Nina Otter, Pietro Liò, Guido Montúfar, and
1376 Michael Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and over-
1377 smoothing in gnn. *Advances in Neural Information Processing Systems*, 35:18527–18541, 2022.

1378 Glen E. Bredon. *Sheaf Theory*, volume 170 of *Graduate Texts in Mathematics*. Springer, 1997.

1379 Eric Bunch, Qian You, Glenn Fung, and Vikas Singh. Simplicial 2-complex convolutional neural
1380 nets. *NeurIPS Workshop on Topological Data Analysis and Beyond*, 2020.

1381 Adam Ó Conghaile. Cohomology in constraint satisfaction and structure isomorphism. *arXiv preprint*
1382 *arXiv:2206.15253*, 2022.

1383 Justin Michael Curry. *Sheaves, Cosheaves and Applications*. PhD thesis, University of Pennsylvania,
1384 2014.

1385 Iulia Duta, Cristian Bodnar, and Pietro Liò. Sheaf hypergraph networks. *Advances in Neural*
1386 *Information Processing Systems*, 36, 2024.

1387 Stefania Ebli, Michaël Defferrard, and Gard Spreemann. Simplicial neural networks. *NeurIPS*
1388 *Workshop on Topological Data Analysis and Beyond*, 2020.

1389 Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks.
1390 *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3558–3565, 2019.

1391 Michael Fourman, Christopher Mulvey, and Dana Scott, editors. *Applications of Sheaves: Proceed-*
1392 *ings of the Research Symposium on Applications of Sheaf Theory to Logic, Algebra and Analysis*,
1393 volume 753 of *Lecture Notes in Mathematics*. Springer, 1977.

1394 Thomas Gebhart, Jakob Hansen, and Paul Schrater. Knowledge sheaves: A sheaf-theoretic framework
1395 for knowledge graph embedding. In *International Conference on Artificial Intelligence and*
1396 *Statistics*, pages 9094–9116. PMLR, 2023.

1397 Robert Ghrist and Yasuaki Hiraoka. Applications of sheaf cohomology and exact sequences to
1398 network coding. In *NOLTA*, 2011.

1399 Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural
1400 message passing for quantum chemistry. In *Int. Conf. Mach. Learn.*, pages 1263–1272. PMLR,
1401 2017.

1402 Lorenzo Giusti, Claudio Battiloro, Paolo Di Lorenzo, Stefania Sardellitti, and Sergio Barbarossa.
1403 Simplicial attention networks. *arXiv preprint arXiv:2203.07485*, 2022.

1404 Lorenzo Giusti, Claudio Battiloro, Lucia Testa, Paolo Di Lorenzo, Stefania Sardellitti, and Sergio
1405 Barbarossa. Cell attention networks. In *2023 International Joint Conference on Neural Networks*
1406 *(IJCNN)*, pages 1–8. IEEE, 2023.

1407 Joseph A. Goguen. Sheaf semantics for concurrent interacting objects. *Mathematical Structures in*
1408 *Computer Science*, 2(2):159–191, 1992.

1409 Christopher Wei Jin Goh, Cristian Bodnar, and Pietro Lio. Simplicial attention networks. In *ICLR*
1410 *2022 Workshop on Geometrical and Topological Representation Learning*, 2022.

1411 Mustafa Hajij, Kyle Istvan, and Ghada Zamzmi. Cell complex neural networks. *NeurIPS 2020*
1412 *Workshop TDA and Beyond*, 2020.

1413 Mustafa Hajij, Karthikeyan Natesan Ramamurthy, Aldo Saenz, and Ghada Zamzmi. High skip net-
1414 works: a higher order generalization of skip connections. In *ICLR 2022 Workshop on Geometrical*
1415 *and Topological Representation Learning*, 2022.

1416 Mustafa Hajij, Ghada Zamzmi, Theodore Papamarkou, Nina Miolane, Aldo Guzmán-Sáenz,
1417 Karthikeyan Natesan Ramamurthy, Tolga Birdal, Tamal K. Dey, Soham Mukherjee, Shreyas N.
1418 Samaga, Neal Livesay, Robin Walters, Paul Rosen, and Michael T. Schaub. Topological deep
1419 learning: going beyond graph data. *arXiv*, 2023.

- 1420 Jakob Hansen and Thomas Gebhart. Sheaf neural networks. In *Workshop on Topological Data*
1421 *Analysis and Beyond*, 2020.
- 1422 Jakob Hansen and Robert Ghrist. Learning sheaf laplacians from smooth signals. In *ICASSP 2019-*
1423 *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages
1424 5446–5450. IEEE, 2019a.
- 1425 Jakob Hansen and Robert Ghrist. Toward a spectral theory of cellular sheaves. *Journal of Applied*
1426 *and Computational Topology*, 3(4):315–358, 2019b.
- 1427 Mikhail Hayhoe, Hans Riess, Victor M Preciado, and Alejandro Ribeiro. Stable and transferable
1428 hyper-graph neural networks. *arXiv preprint arXiv:2211.06513*, 2022.
- 1429 Jianwen Jiang, Yuxuan Wei, Yifan Feng, Jingxuan Cao, and Yue Gao. Dynamic hypergraph neural
1430 networks. In *IJCAI*, pages 2635–2641, 2019.
- 1431 Eun-Sol Kim, Woo Young Kang, Kyoung-Woon On, Yu-Jung Heo, and Byoung-Tak Zhang. Hyper-
1432 graph attention networks for multimodal learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*,
1433 pages 14581–14590, 2020.
- 1434 Henry Kvinge, Michael Robinson, Summet Smith, Danielle Shao, Thomas Martinez, Pablo Camacho-
1435 Torres, Chelsea Matson, and Rushil Anirudh. Sheaves as a framework for understanding and
1436 interpreting model fit. In *Proceedings of the IEEE/CVF International Conference on Computer*
1437 *Vision*, pages 4222–4230, 2021.
- 1438 Wenfei Liang, Bao Nguyen, Xiaomeng Dong, Andrey Razumov, Xiaojun Wan, Qianqian Chen,
1439 Hongyi Zhang, and Shilin Zhao. Fedsheafhn: Personalized federated learning on graph-structured
1440 data. *arXiv preprint arXiv:2405.16056*, 2024.
- 1441 Jerry M. Mendel. Tutorial on higher-order statistics (spectra) in signal processing and system theory:
1442 theoretical results and some applications. *Proceedings of the IEEE*, 79(3):278–305, 1991.
- 1443 Bao Nguyen, Wenfei Liang, Xiaomeng Dong, Andrey Razumov, Qianqian Chen, Hongyi Zhang,
1444 Xiaojun Wan, and Shilin Zhao. Sheaf hypernetworks for personalized federated learning. *arXiv*
1445 *preprint arXiv:2405.20882*, 2024.
- 1446 Robert Peach, Huiyuan Cheng, Dongxia He, Kaijie Zhao, Grace P. Pun, Hanwei Qi, Antoine Duneau,
1447 Arnab Bhowmik, Mikita Goyal, Himanshu Bahl, et al. Implicit gaussian process representation of
1448 vector fields over arbitrary latent manifolds. In *The Twelfth International Conference on Learning*
1449 *Representations*, 2024.
- 1450 Antonio Purificato, Vittorio Bombelli, Federico Grisostomi, Monica Bianchini, Marco Maggini, and
1451 Franco Scarselli. Sheaf4rec: Sheaf neural networks for graph-based recommender systems. *arXiv*
1452 *preprint arXiv:2304.09097*, 2023.
- 1453 Michael Robinson. *Topological signal processing*, volume 81. Springer, 2014.
- 1454 Amit Singer and H-T Wu. Vector diffusion maps and the connection laplacian. *Communications on*
1455 *pure and applied mathematics*, 65(8):1067–1144, 2012.
- 1456 Yellamraju V. Srinivas. A sheaf-theoretic approach to pattern matching and related problems.
1457 *Theoretical Computer Science*, 112(1):53–97, 1993.
- 1458 Arne Wolf and Anthea Monod. Topological community detection: A sheaf-theoretic approach. In
1459 *International Conference on Complex Networks and Their Applications*, pages 29–42. Springer,
1460 2023.