Sheaf Cohomology of Linear Predictive Coding Networks

Jeffrey Seely JEFFREY@SAKANA.AI
Sakana AI

Editors: List of editors' names

Abstract

Predictive coding (PC) replaces global backpropagation with local optimization over weights and activations. We show that linear PC networks admit a natural formulation as cellular sheaves: the sheaf coboundary maps activations to edge-wise prediction errors, and PC inference is diffusion under the sheaf Laplacian. Sheaf cohomology then characterizes irreducible error patterns that inference cannot remove. We analyze recurrent topologies where feedback loops create internal contradictions, introducing prediction errors unrelated to supervision. Using a Hodge decomposition, we determine when these contradictions cause learning to stall. The sheaf formalism provides both diagnostic tools for identifying problematic network configurations and design principles for effective weight initialization for recurrent PC networks.

Keywords: predictive coding, cellular sheaf theory, cohomology, Hodge theory, applied topology, deep linear networks

1. Introduction

Predictive coding (PC) recasts neural network training as an optimization problem over weights and *activations* (Salvatori et al., 2025). This replaces a global loss with a series of layer-local losses. Optimizing over activations (PC inference) enables one to handle arbitrary recurrent topologies without unrolling the computational graph or relying on backpropagation through time (Salvatori et al., 2022a). But it creates a fundamental ambiguity that, to our knowledge, has been overlooked: a node deep in a recurrent network receives error signals from all its connections—some from supervision, others from contradictory feedback loops—with no way to distinguish between them.

We formalize linear PC networks as cellular sheaves (Curry, 2014) to analyze this problem systematically. A *predictive coding sheaf* assigns vector spaces to vertices (neuron activations) and edges (prediction errors) of the underlying computational graph, with restriction maps encoding the network weights. The sheaf coboundary operator δ^0 computes all prediction errors simultaneously. Sheaf cohomology represents the network's capacity to resolve inconsistencies. At a high level, sheaf theory studies how local consistency patches together into global coherence (Rosiak, 2022)—detecting when this local-to-global transfer *succeeds* (captured by the zeroth cohomology H^0) and when it *fails* (captured by H^1). This theme matches the predictive coding ambition: how can locally informed layers coordinate to solve a global task, even amid feedback loops far from any supervision?

We illustrate the correspondence between predictive coding networks and sheaf theory through simple examples (Sections 2 and 3). When input and output nodes are clamped to data and supervision targets, a Hodge decomposition characterizes precisely how the

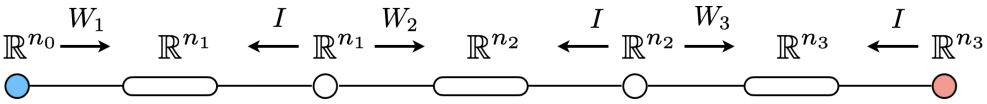

Figure 1: Cellular sheaf over a graph. Vector spaces are assigned to nodes and edges, linear restriction maps are assigned to vertex-to-edge pairs. Layers $h_i = W_i h_{i-1}$ are encoded via a restriction pair (W_i, I). This sheaf represents the deep linear network $y = W_3 W_2 W_1 x$.

supervision signal distributes prediction errors across edges, and how it distributes neuron activations across vertices (Section 4). This framework reveals why certain recurrent networks fail to learn: feedback loops can create internal contradictions that dominate supervision signals. Experiments demonstrate orders-of-magnitude differences in learning convergence rates based on weight initialization (Section 5); importantly, the stark difference is not due to weight magnitude (all our initializations are orthonormal) but due to how the orientations of the weights create *tension* or *resonance* when traced around feedback loops within the network. We focus on linear networks (where sheaf cohomology and Hodge decompositions can be readily computed), though what follows can be regarded as a linear analysis of nonlinear PC networks evaluated at equilibria.

2. Deep Linear Networks as Constraints and Penalties

We will establish the connection between linear PC networks and cellular sheaves by way of example.

2.1. A three-layer warm-up

Let $x \in \mathbb{R}^{n_0}$, $h_1 \in \mathbb{R}^{n_1}$, $h_2 \in \mathbb{R}^{n_2}$, and $y \in \mathbb{R}^{n_3}$. A three-layer deep linear network optimizes

$$\min_{W_1, W_2, W_3} \ \tfrac{1}{2} \left\| y - W_3 W_2 W_1 x \right\|_2^2 . \tag{1}$$

Alternatively, we can write Equation (1) as a constrained optimization problem,

$$\min_{W_1, W_2, W_3, h_1, h_2} \ \tfrac{1}{2} \left\| y - W_3 h_2 \right\|_2^2 \quad \text{s.t.} \quad h_1 = W_1 x, \quad h_2 = W_2 h_1, \tag{2}$$

whereby the key feature is that now h_1 and h_2 are optimization variables. When constraints are satisfied exactly, the problem is equivalent to Equation (1).

2.2. Predictive coding as a penalty method

One interpretation of predictive coding is that it is a soft relaxation of Equation (2) via quadratic regularization terms, otherwise known as a penalty method.[1] The resulting loss

1. To our knowledge, PC as a penalty method is not standard pedagogy, though we find this perspective instructive for what follows.

function, called the total energy E_{PC}, is the sum of squared residuals (or prediction errors):

$$E_{\text{PC}}(h_1, h_2, W_1, W_2, W_3) = \tfrac{1}{2}\big(\|r_1\|_2^2 + \|r_2\|_2^2 + \|r_3\|_2^2\big), \tag{3}$$

$$r_1 = h_1 - W_1 x, \quad r_2 = h_2 - W_2 h_1, \quad r_3 = y - W_3 h_2. \tag{4}$$

PC is a bilevel optimization problem: **Inference** fixes W_is and minimizes E_{PC} over h_is; **learning** then fixes h_is and updates W_is. In practice, inference amounts to multiple gradient descent steps (w.r.t. h_is) and learning is a single gradient descent step (w.r.t. W_is) before h_is are initialized again for the next minibatch; see Salvatori et al. (2022b) for other update schedules.

3. Sheaf Preliminaries for Predictive Coding

We formalize a linear PC network as a cellular sheaf (Curry, 2014), establishing nomenclature along the way.

From activations to cochains. Following the three-layer example, we organize (concatenate) activations into a **0-cochain**:

$$s = (x, h_1, h_2, y) \in C^0 := \mathbb{R}^{n_0} \oplus \mathbb{R}^{n_1} \oplus \mathbb{R}^{n_2} \oplus \mathbb{R}^{n_3}. \tag{5}$$

The vector-concatenation of prediction errors forms a **1-cochain**:

$$r = (r_1, r_2, r_3) \in C^1 := \mathbb{R}^{n_1} \oplus \mathbb{R}^{n_2} \oplus \mathbb{R}^{n_3}. \tag{6}$$

The coboundary operator. The coboundary $\delta^0 : C^0 \to C^1$ computes all prediction errors simultaneously from the activations:

$$\delta^0 s = \begin{bmatrix} -W_1 & I_{n_1} & 0 & 0 \\ 0 & -W_2 & I_{n_2} & 0 \\ 0 & 0 & -W_3 & I_{n_3} \end{bmatrix} \begin{bmatrix} x \\ h_1 \\ h_2 \\ y \end{bmatrix} = \begin{bmatrix} h_1 - W_1 x \\ h_2 - W_2 h_1 \\ y - W_3 h_2 \end{bmatrix}, \tag{7}$$

where we note the per-edge sign convention.

PC Inference is sheaf diffusion. From Equation (7) it is clear that PC energy Equation (3) can be written compactly as $E_{\text{PC}}(s) = \frac{1}{2}\|\delta^0 s\|^2$. Minimizing the energy E_{PC} with respect to s via gradient descent yields the gradient flow $\dot{s} = -Ls$ (or its discrete-time variant), where $L = (\delta^0)^\top \delta^0$ is the **sheaf Laplacian**. The gradient flow is known as **sheaf diffusion**, as it generalizes notions of graph diffusion to sheaves (Hansen and Ghrist, 2018).

Network sheaves. More generally, for an arbitrary graph $G = (V, E)$, a *network sheaf* $\mathcal{F}$ over G assigns vector spaces $\mathcal{F}(v) = \mathbb{R}^{n_v}$ to vertices and $\mathcal{F}(e) = \mathbb{R}^{m_e}$ to edges. Each edge $e = (u, v)$ carries linear restriction maps $\rho_{e \leftarrow u} : \mathcal{F}(u) \to \mathcal{F}(e)$ and $\rho_{e \leftarrow v} : \mathcal{F}(v) \to \mathcal{F}(e)$.

Generalizing the three-layer example from the previous section, we define a *predictive coding sheaf* to be a network sheaf with the following convention: $e = (u \to v)$ denotes a forward computation $s_v = W_e s_u$, encoded via restrictions $\rho_{e \leftarrow u} = W_e$ and $\rho_{e \leftarrow v} = I$. The coboundary then computes the prediction error at each edge:

$$(\delta^0 s)_e = \rho_{e \leftarrow v} s_v - \rho_{e \leftarrow u} s_u = s_v - W_e s_u. \tag{8}$$

This definition applies for a graph G with multiple edges between vertices. Formally, our graph is undirected but where we use notation $e = (u \to v)$ to specify the placement of the W_e and I restriction maps.

Cohomology. The sheaf coboundary induces two cohomology groups that characterize network consistency:

$$H^0(G, \mathcal{F}) = \ker \delta^0 = \{s \in C^0 : \delta^0 s = 0\}, \tag{9}$$

$$H^1(G, \mathcal{F}) = C^1 / \operatorname{im} \delta^0 \cong \ker(\delta^0)^\top. \tag{10}$$

Here H^0 consists of activation patterns (on vertices) with zero prediction error everywhere, while H^1 represents prediction error patterns (on edges) that cannot arise from any activation choice. In finite dimensions with inner products we can represent $H^1(G, \mathcal{F})$ by $\ker(\delta^0)^\top$. In our setting the predictive coding sheaf $\mathcal{F}$ depends on the current network weights, so H^0 and H^1 should be understood as weight-dependent linear subspaces of C^0 and C^1, not as topological invariants of the underlying graph.

4. Relative Systems and Clamped Networks

In predictive coding, one typically does not work with the full unconstrained system. Instead, one clamps input x to data and output y to supervision targets during network training. In sheaf terms, this corresponds to a relative system. Let $V = V_{\text{free}} \cup V_{\text{clamped}}$ partition the vertices. The relative coboundary D extracts the columns of δ^0 corresponding to free vertices. In the three-layer network, we have:

$$D = \begin{bmatrix} I_{n_1} & 0 \\ -W_2 & I_{n_2} \\ 0 & -W_3 \end{bmatrix}, \quad b = \begin{bmatrix} -W_1 x \\ 0 \\ y \end{bmatrix}. \tag{11}$$

During inference, we optimize only over internal activations $z = (h_1, h_2) \in \mathbb{R}^{n_1} \oplus \mathbb{R}^{n_2}$. The effect of clamping data nodes can be interpreted as creating a "target prediction error" b as a 1-cochain.

More precisely, clamping induces the inhomogeneous least-squares system

$$E_{\text{rel}}(z) = \tfrac{1}{2}\|Dz + b\|^2, \quad z \in C^0_{\text{free}}. \tag{12}$$

When D has full column rank (as in all our examples), the minimizer is unique: $z^\star = -D^\dagger b$, where $\bullet^\dagger$ indicates the pseudoinverse. More generally, the solution set is the affine space $\{z^\star + z_{\text{null}} : z_{\text{null}} \in \ker D\}$, and gradient descent from $z = 0$ converges to the minimum-norm representative $z^\star$. At any minimizer, the residual satisfies

$$r^\star := Dz^\star + b \in \ker D^\top, \qquad D^\top r^\star = 0. \tag{13}$$

Equivalently, the target b admits an orthogonal (Hodge) decomposition

$$b = \underbrace{(-Dz^\star)}_{\in \operatorname{im} D} + \underbrace{r^\star}_{\in \ker D^\top}. \tag{14}$$

This decomposition makes explicit the part of the "target prediction error" b that can be eliminated by inference ($-Dz^\star$) and the part that cannot ($r^\star$).

Harmonic and diffusive operators. The orthogonal decomposition of b allows us to precisely state how b influences edge patterns and activation patterns at optimal inference. Define the harmonic projector $\mathcal{H}$ and diffusive operator $\mathcal{G}$ as

$$\mathcal{H} := I - DD^{\dagger}, \qquad \mathcal{G} := L_{\text{rel}}^{\dagger} D^{\top} = D^{\dagger}, \tag{15}$$

where $L_{\text{rel}} = D^{\top}D$ is the relative sheaf Laplacian. Then the minimum-norm solution of Equation (12) yields the compact form,

$$r^{\star} = \mathcal{H}b, \qquad z^{\star} = -\mathcal{G}b, \qquad E_{\text{rel}}(z^{\star}) = \tfrac{1}{2}\left\|\mathcal{H}b\right\|_2^2. \tag{16}$$

We can also think of b as an "excitation vector" for the network, or as boundary conditions. The projector $\mathcal{H}$ distributes that excitation across edges into the harmonic subspace of C^1, $\ker D^{\top}$. The operator $\mathcal{G}$ diffuses the same excitation to internal vertices, determining which sources are actually active after inference/diffusion has converged.

Learning requires harmonic-diffusive overlap. For a trainable edge $e = (u \to v)$ with weight W_e and source activation s_u, the per-edge gradient is

$$\frac{\partial E_{\text{rel}}}{\partial W_e} = (W_e s_u - s_v) s_u^{\top} = -\, r_e\, s_u^{\top}. \tag{17}$$

Learning requires both a residual and a source. If either r_e or s_u vanishes, the update at edge e is zero. At optimal inference, we can specify where in the network this occurs. Define the full activation vector $s^{\star} \in C^0$ with free vertices from $z^{\star}$ and clamped vertices at their fixed values: $(s^{\star})_{\text{free}} = z^{\star}$, $(s^{\star})_{\text{clamped}} = (x, y)$. Using Equation (16),

$$\left.\frac{\partial E_{\text{rel}}}{\partial W_e}\right|_{z^{\star}} = -\,(\mathcal{H}b)_e\,(s^{\star})_u^{\top}. \tag{18}$$

When u is a free vertex, $(s^{\star})_u = -(\mathcal{G}b)_u$, yielding

$$\left.\frac{\partial E_{\text{rel}}}{\partial W_e}\right|_{z^{\star}} = \underbrace{(\mathcal{H}b)_e}_{\text{harmonic (edge)}} \quad \underbrace{(\mathcal{G}b)_u^{\top}}_{\text{diffusive (vertex)}}. \tag{19}$$

4.1. Relative cohomology

To connect our clamped setting to sheaf theory more formally, we point the reader to Hansen and Ghrist (2020, 2018). In brief: the clamped setting yields a relative cochain complex $(C^{\bullet}(G, V_{\text{clamped}}; \mathcal{F}), D)$ with target b defining a cohomology class $[b] \in H^1_{\text{rel}} \cong \ker D^{\top}$ with $r^{\star} = \mathcal{H}b$ its harmonic representation. In this language, PC inference solves a discrete Dirichlet problem—harmonically extending boundary values to the interior with H^1_{rel} measuring obstructions to this extension.

4.2. A minimal recurrent example

Consider augmenting our three-layer network with a feedback edge $h_2 \to h_1$, creating a cycle. The coboundary and relative coboundary gain an additional row. The relative system is

$$D = \begin{bmatrix} I_{n_1} & 0 \\ -W_2 & I_{n_2} \\ 0 & -W_3 \\ I_{n_1} & -W_2^{\text{FB}} \end{bmatrix}, \quad b = \begin{bmatrix} -W_1 x \\ 0 \\ y \\ 0 \end{bmatrix}. \tag{20}$$

The cycle induces a monodromy $\Phi = W_2^{\mathrm{FB}} W_2 : \mathbb{R}^{n_1} \to \mathbb{R}^{n_1}$. During inference, h_1 receives prediction errors from three sources: the forward path ($h_1 - W_1 x$), the next layer ($h_2 - W_2 h_1$), and the feedback loop ($h_1 - W_2^{\mathrm{FB}} h_2$). Consider two extreme cases of the weight initialization of the system:

Resonance ($\Phi \approx I$). In this case, the feedback loop reinforces itself—changes in h_1 return unchanged. We show below that this may introduce slow modes during inference. But once converged, the network has no internal contradictions to resolve.

Internal tension ($\Phi \approx -I$). The feedback loop contradicts itself—changes in h_1 return negated. During inference, h_1 compromises between forward predictions and contradictory feedback. Learning thus requires resolving two tasks: match the supervision signal and resolve the internal contradiction.

How *tension* and *resonance* enter the harmonic and diffusion operators follows from their effect on the linear dependence of the block rows of D. While the algebraic story is tractable, we pursue empirical demonstration in the next section, using a 10-node network to concretely show how monodromy shapes these operators and ultimately learning itself.

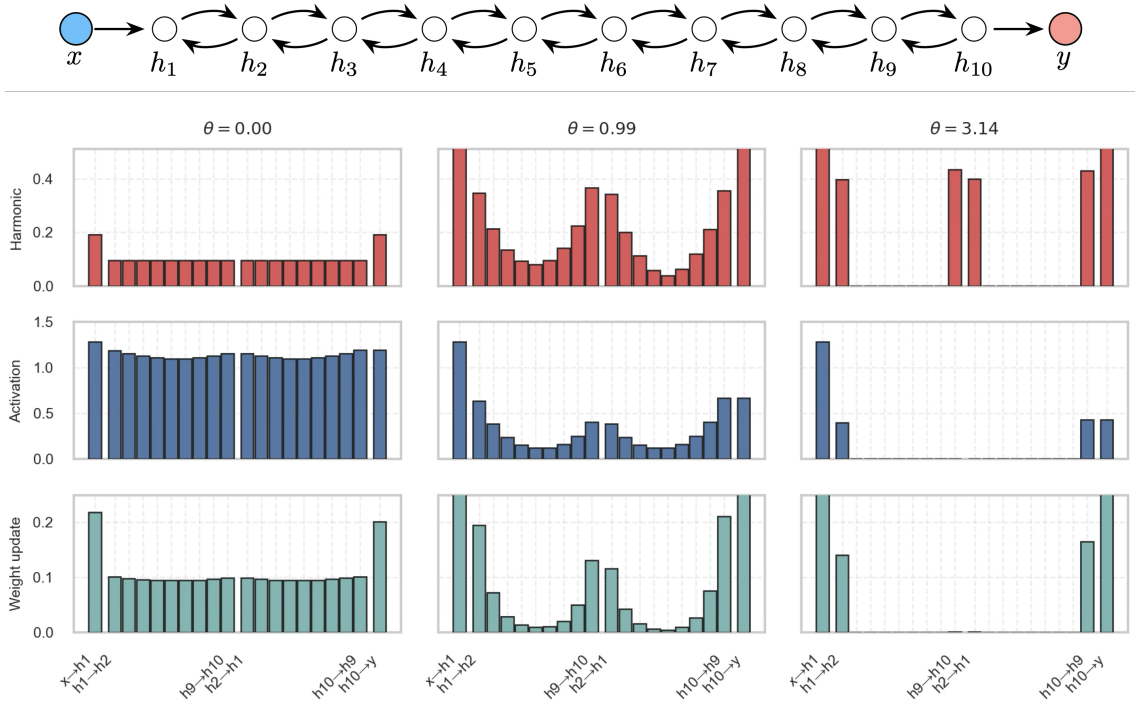

Figure 2: Harmonic-diffusive separation in a 10-node network for three initializations. Top: sample-averaged harmonic load per edge. Middle: sample-averaged diffusive activation per vertex. Bottom: weight update magnitude. See Section 5 for details. For $\theta = 0$ (resonant), harmonic load and diffusion overlap broadly. For $\theta = \pi$ (contradictory), harmonic load concentrates on internal edges where diffusion does not reach, starving the corresponding updates.

5. A Canonical Knotted Network

Consider the network in Figure 2: a 10-layer deep linear network with feedback cycles between each layer. We initialize forward weights W_i as random 2×2 orthonormal matrices. Feedback weights W_i^{FB} are initialized so the monodromy $\Phi_i(\theta) = W_i^{\mathrm{FB}} W_i$ induces a net rotation of θ. Thus $\theta = 0$ sets $W_i^{\mathrm{FB}} = W_i^{-1}$ (resonance) and $\theta = \pi$ yields $\Phi_i(\theta) = -I$ (strong internal tension).

We run PC to disentangle the network and recover the identity mapping. We draw samples $x \in \mathbb{R}^2$ from a standard normal and set $y = x$ (optionally with small i.i.d. Gaussian noise). Let $B = [b^{(1)}, \ldots, b^{(N)}]$ denote a batch of $N = 128$ target vectors. We plot: (i) **harmonic load** per edge: $\frac{1}{N} \sum_{i=1}^{N} \|(\mathcal{H} b^{(i)})_e\|_2$; (ii) **diffusive activation** per vertex: $\frac{1}{N} \sum_{i=1}^{N} \|(z^{\star(i)})_v\|_2$; (iii) **weight gradient magnitude**: $\|\frac{1}{N} \sum_{i=1}^{N} r_e^{(i)} (s_u^{(i)})^\top\|_F$.

For $\theta = 0$ initialization (Figure 2, left column), harmonic load distributes evenly across all edges. Sheaf diffusion/inference near-uniformly spreads activations across vertices, yielding strong weight updates everywhere. For $\theta = \pi$ (Figure 2, right column), harmonic load concentrates only on internal edges to/from nodes h_1 and h_{10}. But diffusion is blocked at h_1 and h_{10}, starving updates on the adjacent edges because h_2 and h_9 remain inactive: contradictory feedback induces resistance to the diffusion flow.

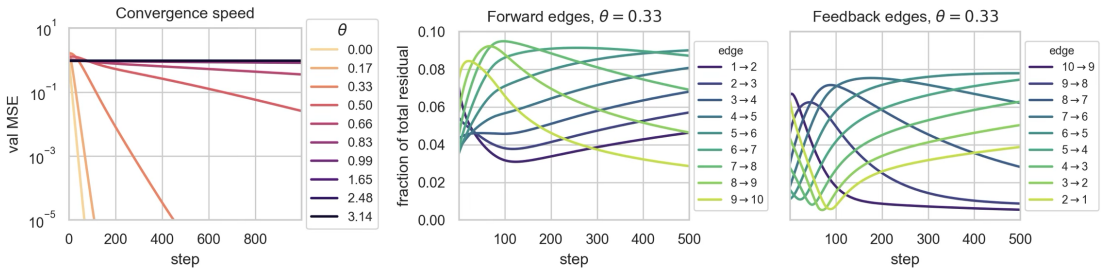

Figure 3: *Left:* validation MSE of the 10-node network under different weight initializations parametrized by θ on a single validation batch B_{val} (from the same distribution as training batch B). Validation MSE is computed with the y node unclamped (i.e., treated as a free vertex, updated during inference rather than fixed to target values). Networks with contradictory feedback ($\theta > 0.4$) exhibit extremely slow convergence. *Right:* Temporal evolution of harmonic load during training shows progressive "unknotting" dynamics for an intermediate $\theta = 0.33$.

To visualize training dynamics, we train for 1000 steps at a fixed learning rate $\eta = 0.1$. For each step, we solve for the optimal (and unique) $z^\star$ to study the effects of weight initialization purely on learning dynamics instead of inference dynamics. For learning dynamics, only networks with $\theta < 0.4$ achieve ≤ 0.001 validation MSE after 1000 steps. Figure 3 shows harmonic load evolution for an intermediate $\theta = 0.33$. The choice of orthonormal initialization ensures identical per-weight norms across layers, emphasizing that learnability depends on the global wiring of the feedback loops rather than local weight scales.

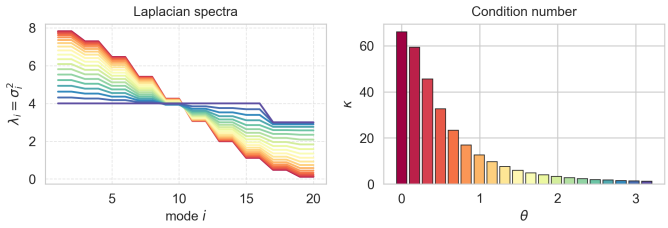

Figure 4: Spectrum of $L = D^\top D$ vs. feedback angle θ. Resonant feedback can slow inference (large $\kappa = \lambda_{\max}/\lambda^+_{\min}$) but still allows learning once converged (see Figure 3).

Different learning rates η can enable convergence for larger θ values, but the stark difference in convergence rates in Figure 3 demonstrates marked sensitivity to the properties of D despite identical network architecture. In the appendix, we show similar analysis and results for an all-to-all predictive coding network (Appendix C).

6. Discussion

Our main contribution is to show that linear predictive coding networks fit naturally within a cellular sheaf framework, recasting intuitions about prediction errors in recurrent PC networks as precise questions in the linear algebra and geometry of cellular sheaves.

From linear to nonlinear. The present development can be reinterpreted by replacing each occurrence of W with the Jacobian of a nonlinear layer f evaluated at an equilibrium of the inference dynamics. Alternatively, introducing edgewise nonlinearities yields a nonlinear sheaf Laplacian (Zaghen et al., 2024; Zaghen, 2024; Hanks et al., 2025).

Inference dynamics. Our current focus is on learnability given converged or optimal inference. In practice, and with nonlinear systems, one takes multiple gradient descent steps, where the conditioning of the inference dynamics plays a more important role. See Appendix A for a preconditioned sheaf diffusion that preserves the minimizer set while improving conditioning.

Predictive coding variations. There exist multiple formulations of PC networks in the literature, which differ in how prediction errors are aggregated. Classical formulations typically associate a single prediction-error unit to each neuron or layer: all incoming predictions are combined and compared to the current state, yielding a neuron-wise squared error term in the energy (Friston, 2005; Whittington and Bogacz, 2017; Millidge et al., 2021). By contrast, our formulation uses an edge-wise factorized energy: each incident connection contributes its own squared prediction error term $\|s_v - W_e s_u\|^2$, so a neuron state is constrained simultaneously by independent consistency conditions along all of its edges. Architectures with multiple error channels per unit or auxiliary local losses in deep PC networks are empirically close in spirit to this edge-wise view (Han et al., 2018; Tschantz et al., 2023; Oliviers et al., 2025). Extending our analysis to the neuron-wise setting amounts to

working with a sheaf over a hypergraph, which still supports sheaf diffusion, Laplacians, and cohomology (Anton et al., 2025). Finally, many PC models introduce precision (inverse-variance) weights on prediction errors (Friston, 2005; Millidge et al., 2021); in our sheaf view these correspond to modifying the inner products on edge and vertex stalks, a direction we leave open.

7. Related Work

The difficulty in inference and learning in predictive coding has been well-studied (Qi et al., 2025; Frieder and Lukasiewicz, 2022), including in the case of deep linear networks (Innocenti et al., 2024). To improve learning in PC networks, recent methods have focused on depth-wise scaling (Ishikawa et al., 2024; Innocenti et al., 2025) for feedforward networks, showing that an appropriate scaling of the weight initialization can enable stable learning in feedforward PC networks with over 100 layers. PC networks with recurrent topologies have been studied in Oliviers et al. (2025); Ororbia et al. (2025); Han et al. (2018). Standard deep linear networks have proven informative for understanding learning dynamics (Saxe et al., 2013; Nam et al., 2025) with observations that extend to nonlinear networks.

Cellular sheaves have been used to model opinion dynamics on social networks and other distributed systems (Hansen and Ghrist, 2020, 2019; Hanks et al., 2025). Structural measures of internal tension in sheaves, such as effective resistance, have been studied in this context (Hansen and Ghrist, 2018), and a connection between 1-cochains and predictive-coding error units has been noted in Schmid (2025).

Sheaf neural networks generalize graph neural networks by using sheaf diffusion as the basic message-passing primitive (Hansen and Gebhart, 2020; Bodnar et al., 2022; Barbero et al., 2022a,b). These models operate on data sheaves over graphs (social networks, molecules, recommender graphs), and use the sheaf Laplacian to mitigate oversmoothing and to handle heterophily. By contrast, we apply sheaf theory to the *computational graph* of a predictive coding network, with no particular input graph structure.

References

Ayzenberg Anton, Gebhart Thomas, Magai German, and Solomadin Grigory. Sheaf theory: from deep geometry to deep learning. *arXiv preprint*, 2025. arXiv [math.AT].

Federico Barbero, Cristian Bodnar, Haitz Sáez de Ocáriz Borde, Michael Bronstein, Petar Veličković, and Pietro Liò. Sheaf neural networks with connection laplacians. *arXiv preprint*, 2022a. arXiv [cs.LG].

Federico Barbero, Cristian Bodnar, Haitz Sáez de Ocáriz Borde, and Pietro Lio. Sheaf attention networks. In *NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations*, 2022b.

Cristian Bodnar, Francesco Di Giovanni, Benjamin Paul Chamberlain, Pietro Lio, and Michael M Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in GNNs. In *Advances in Neural Information Processing Systems*, 2022.

Justin Curry. *Sheaves, Cosheaves and Applications*. Phd thesis, University of Pennsylvania, 2014.

Simon Frieder and Thomas Lukasiewicz. (non-)convergence results for predictive coding networks. *ICML*, 162:6793–6810, 2022.

Karl Friston. A theory of cortical responses. *Philos. Trans. R. Soc. Lond. B Biol. Sci.*, 360 (1456):815–836, 2005.

Kuan Han, Haiguang Wen, Yizhen Zhang, Di Fu, Eugenio Culurciello, and Zhongming Liu. Deep predictive coding network with local recurrent processing for object recognition. *Advances in Neural Information Processing Systems*, 31, 2018.

Tyler Hanks, Hans Riess, Samuel Cohen, Trevor Gross, Matthew Hale, and James Fairbanks. Distributed multi-agent coordination over cellular sheaves. *arXiv preprint*, 2025. arXiv [math.OC].

Jakob Hansen and Thomas Gebhart. Sheaf neural networks. *arXiv [cs.LG]*, 2020.

Jakob Hansen and Robert Ghrist. Toward a spectral theory of cellular sheaves. *arXiv preprint*, 2018. arXiv [math.AT].

Jakob Hansen and Robert Ghrist. Distributed optimization with sheaf homological constraints. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2019.

Jakob Hansen and Robert Ghrist. Opinion dynamics on discourse sheaves. *arXiv preprint*, 2020. arXiv [math.DS].

Francesco Innocenti, El Mehdi Achour, Ryan Singh, and Christopher L Buckley. Only strict saddles in the energy landscape of predictive coding networks? *arXiv preprint*, 2024. arXiv [cs.LG].

Francesco Innocenti, El Mehdi Achour, and Christopher L Buckley. μPC: Scaling predictive coding to 100+ layer networks. *arXiv [cs.LG]*, 2025.

Satoki Ishikawa, Rio Yokota, and Ryo Karakida. Local loss optimization in the infinite width: Stable parameterization of predictive coding networks and target propagation. *arXiv preprint*, 2024. arXiv [cs.LG].

Beren Millidge, Anil Seth, and Christopher L Buckley. Predictive coding: A theoretical and experimental review. *arXiv [cs.AI]*, 2021.

Yoonsoo Nam, Seok Hyeong Lee, Clementine Domine, Yea Chan Park, Charles London, Wonyl Choi, Niclas Goring, and Seungjai Lee. Position: Solve layerwise linear models first to understand neural dynamical phenomena (neural collapse, emergence, lazy/rich regime, and grokking). *arXiv [stat.ML]*, 2025.

Gaspard Oliviers, Mufeng Tang, and Rafal Bogacz. Bidirectional predictive coding. *arXiv preprint*, 2025. arXiv [cs.LG].

Alexander Ororbia, Karl Friston, and Rajesh P N Rao. Meta-representational predictive coding: Biomimetic self-supervised learning. *arXiv preprint*, 2025. arXiv [cs.NE].

Chang Qi, Matteo Forasassi, Thomas Lukasiewicz, and Tommaso Salvatori. Towards the training of deeper predictive coding neural networks. *arXiv preprint*, 2025. arXiv [cs.LG].

Daniel Rosiak. *Sheaf Theory through examples*. The MIT Press, 2022.

Tommaso Salvatori, Luca Pinchetti, Beren Millidge, Yuhang Song, Tianyi Bao, Rafal Bogacz, and Thomas Lukasiewicz. Learning on arbitrary graph topologies via predictive coding. *Adv. Neural Inf. Process. Syst.*, 35:38232–38244, 2022a.

Tommaso Salvatori, Yuhang Song, Yordan Yordanov, Beren Millidge, Zhenghua Xu, Lei Sha, Cornelius Emde, Rafal Bogacz, and Thomas Lukasiewicz. A stable, fast, and fully automatic learning algorithm for predictive coding networks. *arXiv [cs.NE]*, 2022b.

Tommaso Salvatori, Ankur Mali, Christopher L Buckley, Thomas Lukasiewicz, Rajesh P N Rao, Karl Friston, and Alexander Ororbia. A survey on neuro-mimetic deep learning via predictive coding. *Neural Netw.*, 195(108161):108161, 2025.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv [cs.NE]*, 2013.

Eric Schmid. Applied sheaf theory for multi-agent artificial intelligence (reinforcement learning) systems: A prospectus. *arXiv preprint*, 2025.

Alexander Tschantz, Beren Millidge, Anil K Seth, and Christopher L Buckley. Hybrid predictive coding: Inferring, fast and slow. *PLoS Comput. Biol.*, 19(8):e1011280, 2023.

James C R Whittington and Rafal Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural Comput.*, 29(5):1229–1262, 2017.

Olga Zaghen. Nonlinear sheaf diffusion in graph neural networks. *arXiv preprint*, 2024. arXiv [cs.LG].

Olga Zaghen, Antonio Longa, Steve Azzolin, Lev Telyatnikov, Andrea Passerini, and Pietro Liò. Sheaf diffusion goes nonlinear: Enhancing GNNs with adaptive sheaf laplacians. In *Geometry-grounded Representation Learning and Generative Modeling Workshop (GRaM) at ICML 2024*, pages 264–276. PMLR, 2024.

Appendix A. Laplacian preconditioner

Following Hansen and Ghrist (2018), for the relative least-squares problem $E_{\text{rel}}(z) = \frac{1}{2}\|Dz + b\|^2$ with $L_{\text{rel}} = D^\top D$, use the block-Jacobi preconditioner

$$M := \text{blkdiag}(L_{\text{rel}}) \quad \text{(vertex blocks)}. \tag{21}$$

Run preconditioned gradient descent

$$z^{(t+1)} = z^{(t)} - \eta M^{-1} D^\top \big(D z^{(t)} + b\big). \tag{22}$$

Fixed points satisfy $M^{-1}D^\top(Dz+b) = 0 \iff D^\top(Dz+b) = 0$, so the minimizer set is unchanged. If L_{rel} is nonsingular, the unique optimizer remains $z^\star = -L_{\mathrm{rel}}^{-1} D^\top b$. If $\ker D \neq \{0\}$, the affine solution set is the same and the iteration selects the M-minimum-norm representative within it.

Equivalently, endow the vertices with the inner product $\langle u, v\rangle_M = u^\top M v$ and keep the standard inner product on edges. Then the adjoint of the coboundary is $D_M^* = M^{-1}D^\top$, so the preconditioned iteration is steepest descent for the same objective in the M-metric. The diffusion operator becomes $M^{-1}L_{\mathrm{rel}}$, and under the change of variables $y = M^{1/2}z$ this takes the normalized form $M^{-1/2}L_{\mathrm{rel}}M^{-1/2}$.

Appendix B. Gauss–Newton per-edge learning rates

At the inference optimum, the residual on edge $e = (u \to v)$ and source activation at u depend linearly on the boundary b:

$$r_e^\star = (R_e \mathcal{H})\, b, \qquad\qquad s_u^\star = -(S_u \mathcal{G})\, b, \tag{23}$$

where R_e, S_u extract edge and vertex blocks respectively.

For $b \sim \mathcal{N}(0, \Sigma_b)$ on C^1, the Gauss–Newton method requires the source covariance

$$\Sigma_{s_u} = \mathbb{E}\Big[s_u^\star (s_u^\star)^\top\Big] = S_u \mathcal{G}\, \Sigma_b\, \mathcal{G}^\top S_u^\top. \tag{24}$$

When $\Sigma_b = \sigma^2 I$,

$$\Sigma_{s_u} = \sigma^2\, S_u L_{\mathrm{rel}}^\dagger S_u^\top, \tag{25}$$

since $\mathcal{G}\mathcal{G}^\top = L_{\mathrm{rel}}^\dagger$.

Per-edge update rule. For mini-batch gradient $G_e := \sum_{i=1}^N r_e^{(i)} (s_u^{(i)})^\top$ at edge $e = (u \to v)$, the update is

$$W_e \leftarrow W_e - \gamma G_e \big(\Sigma_{s_u} + \varepsilon I\big)^{-1}, \quad 0 < \gamma < 2,\ \varepsilon > 0. \tag{26}$$

This uses only source statistics, omitting residual-side normalization.

Scalar fallback. If a matrix preconditioner is impractical, use a per-source scalar rate:

$$\eta_e^{\mathrm{spec}} \;=\; \frac{\gamma}{\varepsilon + \lambda_{\max}(\Sigma_{s_u})}, \qquad W_e \leftarrow W_e - \eta_e^{\mathrm{spec}} G_e. \tag{27}$$

Estimating Σ_{s_u} without explicit $\mathcal{G}, \mathcal{H}$. Hutchinson probes: draw $\xi_k \sim \mathcal{N}(0, \Sigma_b)$ in C^1 and solve

$$L_{\mathrm{rel}} y_k \;=\; D^\top \xi_k \quad (\text{or } (L_{\mathrm{rel}} + \lambda I) y_k = D^\top \xi_k \text{ if } L_{\mathrm{rel}} \text{ is singular}), \tag{28}$$

then accumulate

$$\widehat{\Sigma}_{s_u} \;\approx\; \frac{1}{q} \sum_{k=1}^{q} (S_u y_k)(S_u y_k)^\top. \tag{29}$$

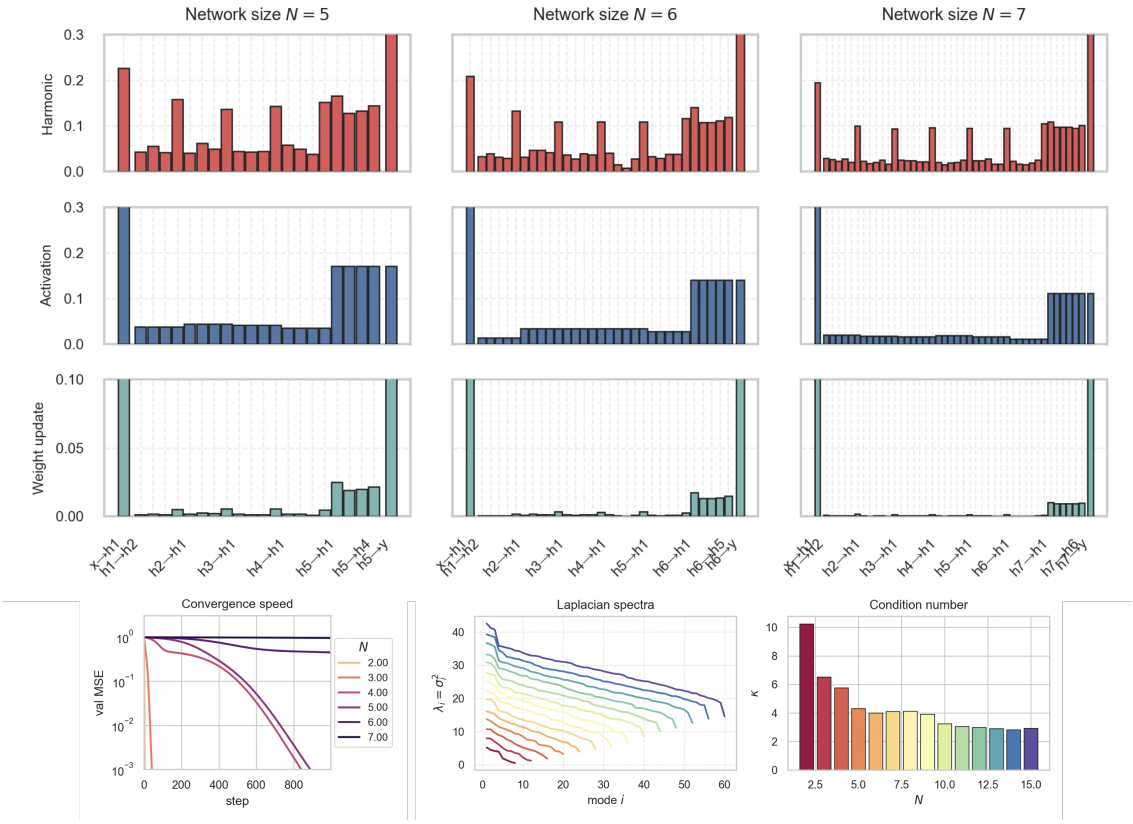

Figure 5: All-to-all variant: harmonic load concentrates on edges incident to the penultimate node h_f, while activations are weaker elsewhere, diminishing updates on $h_i \to h_f$ edges.

Spectral analysis. The selection operator S_u groups all edges from source vertex u under the same covariance Σ_{s_u}. With eigendecomposition $L_{\text{rel}} = U\Lambda U^\top$,

$$\Sigma_{s_u} = S_u U \Lambda_{\text{rel}}^\dagger U^\top S_u^\top \quad \text{(whitened case)}, \tag{30}$$

where small eigenvalues of L_{rel} inflate Σ_{s_u} and reduce the effective step size through $(\Sigma_{s_u} + \varepsilon I)^{-1}$. Poorly conditioned modes thus enforce conservative updates. The operator $\mathcal{G}$ determines vertex reachability under diffusion: vertices with small $s_u^\star$ yield weak gradients. The harmonic projector $\mathcal{H}$ filters the gradient through $r_e^\star = (R_e\mathcal{H})b$; when $\mathcal{H}$ concentrates residuals on edges from weakly activated sources, learning can stall despite preconditioning.

Appendix C. All-to-all network

We observe similar behavior in an all-to-all network. In Figure 5, we show training dynamics of an all-to-all network, where one x and y nodes connect to one of the h_i nodes each, and

every h_i node connects to every other node (bidirectionally) with orthonormally initialized weights, with each h_i 4-dimensional and x and y 2-dimensional. Harmonic load concentrates primarily on edges connected to the penultimate node (that which is connected to y, call it h_f), while activations are weaker on non-penultimate nodes, resulting in weaker updates on $h_i \to h_f$ edges. For network sizes ranging from 2 to 15 only networks of size 5 or less achieve ≤ 0.001 validation MSE after 1000 steps, again highlighting the effect of "internal tension" on convergence rates.