A Generalized Tikhonov Layer for Interpretable-by-design Graph Neural Networks

Nicolas Tremblay
UiT the Arctic University of Norway
& CNRS, France
nicolas.tremblay@cnrs.fr

Benjamin Ricaud
UiT the Arctic University of Norway
benjamin.ricaud@uit.no

Filippo Maria Bianchi
UiT the Arctic University of Norway
& NORCE Norwegian Research Centre
filippo.m.bianchi@uit.no

Abstract

We propose the *Tikhonov layer*, a graph neural network layer that is interpretable by design: once trained, its learned parameters reveal which node features and which aspects of the graph topology were leveraged for prediction. In practice, the layer's propagation matrix takes the closed-form $\mathsf{R} = (p(\mathsf{L}) + \mathsf{Q})^{-1}\mathsf{Q}$, where L is the normalized graph Laplacian, $\mathsf{Q} = \mathrm{diag}(q_1, \ldots, q_n)$ a learnable diagonal matrix of positive node-importance scores, and $p(\cdot)$ a learnable polynomial. For any input feature $\boldsymbol{x}$, the layer output $\mathsf{R}\boldsymbol{x}$ is the exact minimizer of a generalized graph Tikhonov problem that trades off node-level data fidelity against a topology-driven regularization penalty. The learned pair $\{\{q_i\}, p\}$ constitutes a built-in explanation: large q_i indicates that node i's own features drive the prediction, while small q_i signals reliance on the local graph topology; the shape of p reveals whether homophily, heterophily, or a band-pass response is exploited. Expressivity is preserved by routing complexity through a dedicated, arbitrarily deep Q-*network* that produces the importance scores, while the Tikhonov layer itself remains transparent. We prove that distinct node-importance matrices yield distinct propagation operators, structurally coupling the explanation to the computation. Additionally, the Tikhonov layer provides, in a single layer, a global receptive field, mitigating both oversmoothing and oversquashing. Experiments on standard graph classification benchmarks confirm that the model matches (and sometimes outperforms) opaque baselines while producing interpretable and faithful explanations.

1 Introduction

Graph neural networks (GNNs) have seen rapid development over the last decade, advancing expressivity and performance on tasks such as graph and node classification. In graph machine learning, the data is twofold. On the one hand is the structure, often represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of $n = |\mathcal{V}|$ nodes interconnected by $m = |\mathcal{E}|$, possibly weighted, edges. On the other hand are the features, a matrix $\mathsf{X} \in \mathbb{R}^{n \times d}$ of which the i-th row represents the d-dimensional data defined over node i. In the standard layer formulation (with σ a point-wise non-linearity, W a matrix of learnable weights):

$$\mathsf{X}^{(k+1)} = \sigma\left(\mathsf{R}\mathsf{X}^{(k)}\mathsf{W}^{(k)}\right), \tag{1}$$

the graph propagation matrix $\mathsf{R} \in \mathbb{R}^{n \times n}$ encodes how features from neighboring nodes are aggregated, and layers are stacked to progressively widen each node's receptive field. Early choices for R were

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

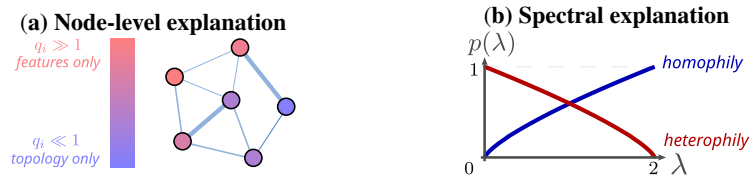


Figure 1: **The explanation** $\{\{q_i\}_{i\in\mathcal{V}}, p(\cdot)\}$**. (a)** After training, the learned q values provide a node-level explanation that can be plotted as a heatmap over $\mathcal{G}$: warm nodes (large q_i) rely on their own features, cool nodes (small q_i) are governed by graph topology. **(b)** The shape of the learned polynomial $p(\lambda)$ reveals how the graph topology influences the nodes with small q_i. For instance, an increasing profile penalizes high frequencies (favors homophily), a decreasing one penalizes low frequencies (favors heterophily). Together, $\{\{q_i\}_{i\in\mathcal{V}}, p(\cdot)\}$ *is* the explanation.



Figure 2: **Illustration of** $\mathsf{R}\boldsymbol{x}$**. (a)** large q_i everywhere. $\mathsf{R} \approx \mathsf{I}$ regardless of $p(\lambda)$: the model ignores topology and behaves like an MLP. **(b)** mixed q_i. Nodes with large q_i keep their features. Nodes with small q_i are influenced by their neighborhood (according to $p(\lambda)$). Here, with $p(\lambda)$ increasing, high frequencies are penalized, and homophily is encouraged.

normalized versions of the adjacency or Laplacian matrices, or polynomials of these matrices; accounting for the class of so-called *spectral* GNNs. More versatile models quickly appeared, under the generic name of *message-passing* GNNs[1], where all entries R_{ij} could be independently learned. For graph-level tasks, a permutation-invariant pooling operation (e.g., max, sum, or mean) followed by an MLP head maps node representations to predictions. We focus on graph-level classification and regression tasks for conciseness, though the setting also directly applies to node-level tasks.

Research questions. In the context of GNNs, interpretability is arguably even more complex than in classical neural networks: one must explain not only how the input features are progressively transformed and leveraged throughout the network, but also how the graph topology itself is exploited during the process. Are both the features and the topology used to solve the task? Or perhaps only the features or only the topology? Or perhaps only the features on some parts of the graph and the topology elsewhere? When the topology is used, how is it leveraged?

Proposed contribution. We address these questions via the design of the propagation matrix R itself. In its simplest, mono-channel form, our contribution constrains R to the form $\mathsf{R} = (p(\mathsf{L}) + \mathsf{Q})^{-1}\mathsf{Q}$, with L the graph's normalized Laplacian[2], $\mathsf{Q} = \mathrm{diag}(q_1, \ldots, q_n)$ a learnable diagonal matrix with positive entries, and $p(\cdot)$ a learnable polynomial satisfying $0 < p(\lambda) < 1$ on $[0, 2]$. This is not arbitrary: $\forall \boldsymbol{x} \in \mathbb{R}^n$, $\mathsf{R}\boldsymbol{x}$ is the unique closed-form solution of the generalized graph Tikhonov problem:

$$\mathsf{R}\,\boldsymbol{x} = \underset{\boldsymbol{z}\in\mathbb{R}^n}{\arg\min} \sum_{i\in\mathcal{V}} q_i(x_i - z_i)^2 + \boldsymbol{z}^\top p(\mathsf{L})\boldsymbol{z}, \tag{2}$$

in which each q_i controls the relative importance of node i's own features against the pressure exerted by the topology term $\boldsymbol{z}^\top p(\mathsf{L})\boldsymbol{z}$. When q_i is large, the original feature at node i is preserved; when q_i is small, the information at that node is mostly discarded in favor of a topology-driven solution. The learned p governs the nature of this influence [30]: an increasing p penalizes high frequencies, pulling outputs toward smooth, homophilic values; a decreasing p penalizes low frequencies, enabling heterophily; higher-degree polynomials yield band-pass responses.

[1] The matrix form of equation (1) is in fact too restrictive to encompass the full versatility of the message-passing paradigm, but it suffices for our purpose here and is much easier to read.

[2] $\mathsf{L} = \mathsf{I} - \mathsf{D}^{-1/2}\mathsf{A}\mathsf{D}^{-1/2}$ with A the adjacency matrix and D the diagonal matrix of degrees. If i is an isolated node (hence with non-invertible degree), we set $\forall j, \mathsf{L}_{ij} = \mathsf{L}_{ji} = 0$. Recall that the spectrum of L is contained in $[0, 2]$.

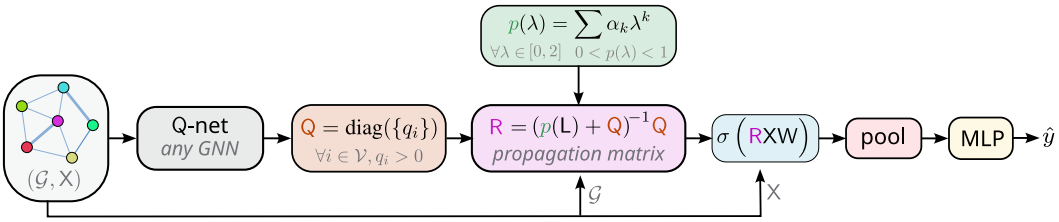


Figure 3: **Overview of the Tikhonov layer.** A Q-network (any GNN) produces the node importances $\mathsf{Q} = \mathrm{diag}(\{q_i\})$; a single Tikhonov layer propagates features globally; pooling + an MLP head yield the prediction (here for graph classification). The learnable parameters are those of the Q-network, the coefficients of the polynomial $p(\lambda)$, the weights W, and the weights of the MLP head.

Crucially, our architecture consists of *a single Tikhonov layer* of this form. The final node features, e.g., those fed to the global pooler followed by the MLP, are simply obtained as

$$\sigma\left(\mathsf{RXW}\right) = \sigma\left((p(\mathsf{L}) + \mathsf{Q})^{-1}\mathsf{QXW}\right).$$

This deliberate simplicity is what makes the model interpretable by design: the learned values $(q_i)_{i\in\mathcal{V}}$ provide a direct and faithful answer to the research questions above. By inspecting the q_i's after training, one can identify which nodes rely on their own features (those with large q_i), which nodes are governed by the graph topology (those with small q_i), and in such cases, how the topology affects the features (via the learned polynomial p). One can, in particular, directly see whether the topology is exploited at all (if all $q_i \gg 1$, R tends to the identity matrix and the Tikhonov layer reduces to an MLP). This is done *without* any post-hoc approximation, since the explanation *is* the model itself. As such, we will refer to the pair $\{\{q_i\}_{i\in\mathcal{V}}, p(\cdot)\}$ as the *explanation*: see Figures 1 and 2 for illustrations.

Remarkably, expressivity is not sacrificed for the sake of interpretability. Rather than learning the $(q_i)_{i\in\mathcal{V}}$ as free parameters, they are produced from $(\mathcal{G}, \mathsf{X})$ by a dedicated GNN — which we call the Q-*network* — that can be made arbitrarily deep and expressive. This design enables the model to naturally handle graphs of varying sizes, and conditions the scores q_i on the graph's topology and node features. This Q-network concentrates the majority of the representational power of the model in a single, well-defined place. The rest of the architecture (the Tikhonov layer and the final MLP) can thus keep its transparency and interpretability. In this sense, our model follows a *separation of concerns*: complexity lives in the Q-network, interpretability lives in the Tikhonov layer. See Fig. 3 for an illustration of the full architecture.

Additional properties. Beyond interpretability, the Tikhonov layer naturally mitigates the effect of two notorious pathologies of deep GNNs, namely *oversmoothing* and *oversquashing*.

Oversmoothing. Stacking many message-passing layers causes all node representations to collapse [19]. Our architecture avoids involuntary oversmoothing in two ways. First, using a *single* Tikhonov layer avoids depth-induced accumulation. Second, since $p(\lambda) > 0$ on all of $[0, 2]$, the operator $p(\mathsf{L})$ is invertible and $\mathsf{R} = (p(\mathsf{L}) + \mathsf{Q})^{-1}\mathsf{Q} \to \mathbf{0}$ as all $q_i \to 0$: the only way to collapse node representations is to drive all importance scores to zero, which is a deliberate model choice rather than an unavoidable pathology.

Oversquashing. In message-passing GNNs, long-range sensitivity $|\partial(\mathsf{R}\boldsymbol{x})_i/\partial x_j|$ decays exponentially with the hop-distance $d_{\mathcal{G}}(i, j)$ at a rate that is fixed by the architecture, regardless of model width [1]. In the Tikhonov layer the analogous quantity is $|\mathsf{R}_{ij}|$, which decays exponentially with $d_{\mathcal{G}}(i, j)$ at rate $2/(K\sqrt{\kappa})$ with K the degree of $p(\cdot)$ and $\kappa = \mathrm{cond}_2(\mathsf{I} + \mathsf{Q}^{-1/2}p(\mathsf{L})\mathsf{Q}^{-1/2})$ (Proposition 3.5). Crucially, κ is *learnable*: when the model drives some q_i's small, κ grows and $2/(K\sqrt{\kappa}) \to 0$, mitigating the decay. The model thus learns to adjust its spatial reach end-to-end, trading local precision against long-range sensitivity through the q_i's.

Note that oversquashing and oversmoothing can occur in the Q-network as it is a standard potentially deep GNN: mitigation strategies from the literature should be applied to address them as needed.

2 Related works

Spectral and Laplacian-based GNNs. Early spectral GNNs [6] defined convolutions via the eigendecomposition of L; Defferrard et al. [10] replaced this with Chebyshev polynomial approximations, and Kipf and Welling [17] further simplified to a first-order filter. All of these use polynomials of L as propagation operators. Our $\mathsf{R} = (p(\mathsf{L}) + \mathsf{Q})^{-1}\mathsf{Q}$ is instead the *regularized inverse* of a polynomial of L, corresponding to the closed-form solution of a Tikhonov problem. APPNP [14] uses a personalized PageRank resolvent $\alpha(\mathsf{I} - (1-\alpha)\hat{\mathsf{A}})^{-1}$, structurally close to our formulation when $\mathsf{Q} = \alpha\mathsf{I}$, but with a fixed scalar α; GDC [15] similarly uses fixed diffusion kernels. The crucial distinction is our *learnable, node-wise* $\mathsf{Q} = \mathrm{diag}(q_1, \ldots, q_n)$, which enables interpretability. ARMA graph convolutions [5] also implement rational spectral filters; our homogeneous case is rational as well, but the learnable node-wise Q gives a variational and interpretable non-stationary extension.

Graph signal processing and Tikhonov regularization. In graph signal processing [27, 25], the Laplacian quadratic form $\boldsymbol{z}^\top \mathsf{L}\boldsymbol{z}$ measures signal smoothness. Graph Tikhonov regularization [34, 28] denoises a signal by minimizing $\|\boldsymbol{x} - \boldsymbol{z}\|_{\mathsf{Q}}^2 + \boldsymbol{z}^\top \mathsf{L}\boldsymbol{z}$, whose solution is $(\mathsf{L} + \mathsf{Q})^{-1}\mathsf{Q}\boldsymbol{x}$, and has been studied in semi-supervised learning [35]. To our knowledge, this has not been used as a *learnable* GNN layer with a node-wise Q. When Q is heterogeneous with sparse large entries and p penalizes high frequencies, the operator is also related to *graph inpainting* [24, 7]: nodes with large q_i act as anchors, and the rest are interpolated by topology.

Attention and node importance in GNNs. Graph attention networks [31] learn edge-level importance weights. Our q_i plays an analogous but distinct role: it governs the balance between node i's own features and topological pressure at the *node* level, through a variational formulation. Unlike an attention coefficient, which only affects its two endpoints, a single q_i influences the solution at every other node through the resolvent $(p(\mathsf{L}) + \mathsf{Q})^{-1}$, giving it a global footprint.

Self-explainable and interpretable GNNs. Our model is a *self-explainable* GNN (SE-GNN): interpretability is built into the architecture, not recovered post-hoc via methods such as GNNEXPLAINER [32], PGEXPLAINER [21], or gradient saliency [26], which produce explanations without faithfulness guarantees (see Longa et al. [20] for a survey; Azzolin et al. [2] for a critical analysis). Typical SE-GNNs such as SE-GNN [8] or PROTGNN [33] follow a two-module design $g = f \circ q$: an explanation extractor q identifies a subgraph, and a classifier f maps it to a prediction. This factorization is critically vulnerable: f can achieve optimal risk while the explanation produced by q is provably unrelated to the model's actual computations [3]. Our architecture avoids this two-module failure mode since there is no detached explanatory subgraph passed to a classifier as an arbitrary code. The prediction pipeline is a *single computation* $\mathrm{MLP}(\mathrm{pool}(\sigma((p(\mathsf{L}) + \mathsf{Q})^{-1}\mathsf{Q}\,\mathsf{X}\mathsf{W})))$, where Q and p are constitutive parts of the propagation operator R. Proposition 3.6 formalizes this structural coupling: for fixed p, the map $\mathsf{Q} \mapsto \mathsf{R}$ is injective, so changing the node-importance explanation necessarily changes the propagation operator. While the Q-network may still learn shortcut or label-encoding patterns, the explanation is faithful to the Tikhonov propagation induced by the model. Also, rather than selecting a discrete subgraph, the Tikhonov layer assigns a continuous scalar importance q_i to each node, directly quantifying the local balance between features and topology: a graded, spatially resolved answer better suited to interpretability than a binary selection.

3 Methodology

In its general form, the multi-channel Tikhonov layer we propose can be written as a J-channel layer: $\sigma\left(\mathsf{R}_1\mathsf{X}\mathsf{W}|\cdots|\mathsf{R}_J\mathsf{X}\mathsf{W}\right)$, where (i) the notation $(\mathsf{M}_1|\cdots|\mathsf{M}_J)$ represents column-wise concatenation of the matrices $\{\mathsf{M}_j\}$; (ii) R_j reads $\mathsf{R}_j = \left(p_j(\mathsf{L}) + \mathsf{Q}_j\right)^{-1}\mathsf{Q}_j$ where $p_j(\cdot)$ are learnable polynomials verifying $0 < p_j(\lambda) < 1$ on $[0, 2]$. Each channel enjoys the variational interpretation of Eq. (2); (iii) The Q_j's are diagonal matrices of positive node-importance parameters. Each Q_j is built via a dedicated Q-network: there are thus J independent Q-networks; (iv) there is only one learnable matrix weight W. This uniqueness enables identifiability even when $J > 1$ (see Proposition 3.8).

Each channel j learns a global graph filter $p_j(\lambda)$ along with where on the graph and with what intensity it should be applied on the features, via the diagonal entries of Q_j. If the data exhibits both homophily and heterophily at different locations and/or scales, the multi-channel design enables the model to learn different local behaviors. This is similar to the "multi-head attention" in transformers, where each head can learn different patterns. In our experimental section, we will set $J = 1$.

3.1 Theoretical properties

We establish the fundamental properties of the Tikhonov layer. Unless stated otherwise, all results are for the single-channel case (extensions to multi-channel are immediate); proofs are in appendix A. Let us recall our notation: L is the normalized Laplacian with eigenvalues $\{\lambda_k\}$, $p(\cdot)$ a polynomial of order $K > 0$ verifying $0 < p(\lambda) < 1$ for $\lambda \in [0, 2]$, Q a diagonal matrix of strictly positive entries, $\mathsf{R} = (p(\mathsf{L}) + \mathsf{Q})^{-1}\mathsf{Q}$ the Tikhonov operator, and $d_{\mathcal{G}}(i, j)$ the hop distance between i and j in $\mathcal{G}$.

Definition 3.1 (Complete K-hop support)**.** *Let $\mathbf{P} = p(\mathbf{L})$. It is well-known that $\mathbf{P}$ is K-localized: $d_{\mathcal{G}}(i, j) > K \implies P_{ij} = 0$. We say that $\mathbf{P}$ has* complete K-hop support *if the converse also holds:*

$$P_{ij} \neq 0 \iff d_{\mathcal{G}}(i, j) \leq K.$$

Assuming that P has complete K-hop support is a very mild condition. For instance, if the coefficients of p are drawn at random from a continuous distribution—as is typically done during initialization before training—then P has complete K-hop support almost surely; see Lemma A.1 for a precise statement. Moreover, we only use this assumption to simplify claims (iii) of Propositions 3.1 and 3.4.

Proposition 3.1 (Basic properties of R)**.** *(i) The inverse of $p(\mathsf{L}) + \mathsf{Q}$ exists: R is well-defined. (ii) The spectrum of R is contained in $(0, 1)$; in particular, R is injective. (iii) Under complete K-hop support, R is symmetric if and only if the q_i's are constant on each connected component of $\mathcal{G}$.*

Point (iii) means that heterogeneous q_i's make the influence of node j on node i differ from that of node i on node j: this asymmetric coupling lets the model represent direction-dependent influence.

Proposition 3.2 (Effect of q_i on feature preservation and local $p(L)$-harmonicity)**.** *Let $p_{max} = \max_k p(\lambda_k)$. Set $\boldsymbol{z} = \mathsf{R}\boldsymbol{x}$. For every $i \in \mathcal{V}$:*

$$|z_i - x_i| \leq \frac{p_{max}}{q_i} \|\boldsymbol{x}\|_2, \qquad |[p(\mathsf{L})\boldsymbol{z}]_i| \;\leq\; \sqrt{q_i}\,\sqrt{p_{max}}\;\|\boldsymbol{x}\|_2 .$$

Thus, when $q_i \gg p_{\max}$, node i's features are preserved. Conversely, when $q_i \ll p_{\max}^{-1}$, node i's output is nearly $p(\mathsf{L})$-harmonic: its value is dictated by the graph topology rather than by its own feature.

Proposition 3.3 (Spectral density and strict generalization)**.** *The family of homogeneous Tikhonov operators $\{(p(\mathsf{L})+q\mathsf{I})^{-1}q\mathsf{I}\}$ with $q > 0$ and $0 < p(\lambda) < 1$ on $[0, 2]$ is dense in the uniform norm in the space of continuous spectral filters $\{g(\mathsf{L}) \mid g : [0, 2] \to (0, 1)\}$. Furthermore, $\mathsf{R} = (p(\mathsf{L}) + \mathsf{Q})^{-1}\mathsf{Q}$ corresponds to a spectral filter $g(\mathsf{L})$ if and only if Q is homogeneous ($\mathsf{Q} = q\mathsf{I}$). For heterogeneous Q, R generically does not commute with L, strictly extending the expressivity of spectral graph filters.*

In the homogeneous case ($\mathsf{Q} = q\mathsf{I}$), the Tikhonov operator's rational parameterization matches the expressivity of polynomial spectral GNNs. A heterogeneous Q breaks the stationarity assumption of traditional spectral methods, enabling flexible, highly expressive *node-adaptive filtering*.

Proposition 3.4 (Global receptive field)**.** *(i) $\mathsf{R}_{ij} = 0$ if i, j belong to different connected components. (ii) If $p(\lambda) = a + b\lambda$ with $b > 0$, then $\mathsf{R}_{ij} > 0$ for every (i, j) in the same connected component. (iii) More generally for any complete K-hop support p: for almost every Q (*i.e.*, for all Q outside a measure-zero exceptional set): $R_{ij} \neq 0$ for every (i, j) in the same connected component.*

Thus, generically in Q, the Tikhonov operator R has a global receptive field. Increasing the polynomial degree K enriches the set of representable spectral filters but does *not* enlarge the spatial reach, which is already global for $K = 1$. While non-zero entries are good news to mitigate oversquashing, they alone do not prevent it: the magnitude of R_{ij}, quantified by the next proposition, is what truly matters.

Proposition 3.5 (Spatial decay of R)**.** *Let the graph be connected and κ be the condition number of $\mathsf{I} + \mathsf{Q}^{-1/2}p(\mathsf{L})\,\mathsf{Q}^{-1/2}$. Then, for any pair (i, j):*

$$|\mathsf{R}_{ij}| \;\leq\; 2\sqrt{\frac{q_j}{q_i}}\;\exp\left(-\frac{2}{K\sqrt{\kappa}}d_{\mathcal{G}}(i, j)\right).$$

This bound makes the oversquashing behavior precise. Both the prefactor $2\sqrt{q_j/q_i}$ and the decay rate $2/(K\sqrt{\kappa})$ depend on the learned Q. When all q_i are large, $\kappa \approx 1$ and the decay is rapid, making the layer act locally, similarly to a standard spectral filter. Conversely, when some q_i entries are small, long-range sensitivity is boosted by two coupled effects: (i) the prefactor $\sqrt{q_j/q_i}$ amplifies distant contributions at destination nodes with small q_i; (ii) κ increases, weakening the exponential attenuation across the graph. Oversquashing is thus not absent by default, but *controllable by design*: the model learns the q_i's end-to-end and can allocate long-range sensitivity where needed.

Proposition 3.6 (Structural coupling of Q and R)**.** *The map $\mathsf{Q} \mapsto (p(\mathsf{L}) + \mathsf{Q})^{-1}\mathsf{Q}$ is injective on the set of diagonal matrices with strictly positive entries. In particular, two models sharing the same p, W, and MLP head but differing in their $\mathsf{Q} \neq \mathsf{Q}'$ produce distinct propagation operators.*

Unlike in two-module SE-GNN architectures, where a classifier can treat the extracted explanation as a separate label code, here Q directly defines the propagation operator seen by the MLP head. Thus, the explanation cannot be changed without changing the Tikhonov computation, which guarantees faithfulness of Q to the induced propagation operator.

Proposition 3.7 (Joint identifiability of (p, Q))**.** *Let (p_1, Q_1) and (p_2, Q_2) be two pairs with diagonal $\mathsf{Q}_j \succ 0$ and polynomials $0 < p_j(\lambda) < 1$ on $[0, 2]$. If both pairs yield the same operator, $(p_1(\mathsf{L}) + \mathsf{Q}_1)^{-1}\mathsf{Q}_1 = (p_2(\mathsf{L}) + \mathsf{Q}_2)^{-1}\mathsf{Q}_2$, then there exists $\alpha > 0$ such that $\mathsf{Q}_2 = \alpha\,\mathsf{Q}_1$ and $p_2(\mathsf{L}) = \alpha\, p_1(\mathsf{L})$.*

The only operator-level non-identifiability is a global rescaling: $(p(\mathsf{L}), \mathsf{Q})$ and $(\alpha p(\mathsf{L}), \alpha\mathsf{Q})$ produce the same R since the factor cancels in $(p(\mathsf{L}) + \mathsf{Q})^{-1}\mathsf{Q}$. This is benign: the *relative* pattern of the q_i's, driving interpretation, is invariant to such rescaling.

Proposition 3.8 (Multi-channel identifiability)**.** *Let $\mathsf{H} = \mathsf{X}\mathsf{W}$ have rank n. Consider the pre-activation J-channel layer output $(\mathsf{R}_1\mathsf{H} \mid \cdots \mid \mathsf{R}_J\mathsf{H})$ with $\mathsf{R}_j = (p_j(\mathsf{L}) + \mathsf{Q}_j)^{-1}\mathsf{Q}_j$. If two parameter sets $\{(p_j, \mathsf{Q}_j)\}_{j=1}^J$ and $\{(p'_j, \mathsf{Q}'_j)\}_{j=1}^J$ yield the same output up to a permutation of the channel blocks, then there exists a permutation π of $\{1, \ldots, J\}$ and positive scalars α_j such that, for every j, $p'_j(\mathsf{L}) = \alpha_j\, p_{\pi(j)}(\mathsf{L})$ and $\mathsf{Q}'_j = \alpha_j\, \mathsf{Q}_{\pi(j)}$. In other words, up to the usual finite-spectrum ambiguity for polynomials, the only non-identifiabilities are per-channel rescaling and channel permutation.*

If J independent weight matrices $\{\mathsf{W}_j\}$ are used instead of the unique W above, a cross-channel compensation becomes possible: one channel's W_j can absorb work from another, breaking per-channel identifiability. Sharing W across channels eliminates this issue at a modest cost in expressivity, and is recommended in our framework, where interpretability is a priority.

3.2 Practical implementation details

Matrix inversion. Computing $\mathsf{R}\mathsf{X} = (p(\mathsf{L}) + \mathsf{Q})^{-1}\,\mathsf{Q}\mathsf{X}$ for a feature matrix $\mathsf{X} \in \mathbb{R}^{n\times d}$ requires solving d linear systems with the symmetric positive definite matrix $\mathsf{M} = p(\mathsf{L}) + \mathsf{Q}$. An exact sparse Cholesky factorization costs up to $\mathcal{O}\left(dn^2 + (Km)^{3/2}\right)$, where m is the number of edges and $K = \deg(p)$. While often cheaper than dense $\mathcal{O}(n^3)$ inversion, this is significantly more expensive than standard GNN layers (*e.g.*, GIN or GCN) operating in $\mathcal{O}(dm)$. We therefore use Preconditioned Conjugate Gradient (PCG). PCG solves $\mathsf{M}\boldsymbol{z} = \mathsf{Q}\boldsymbol{x}$ iteratively using matrix-vector products with M at $O(Km)$ per iteration. The number of iterations before convergence scales as $\mathcal{O}(\sqrt{\kappa(\mathsf{M})})$, where $\kappa(\mathsf{M})$ is the condition number. In practice, we use a very cheap positive diagonal preconditioner $\mathsf{P} = \mathrm{diag}\big(\max\{p(1) + q_i, \varepsilon_P\}\big)$, where $\varepsilon_P = 10^{-8}$ and the maximum is applied entrywise. It is trivial to compute and reduces the condition number, typically yielding convergence in a small number of iterations. For d right-hand sides, the total cost is $O(dTKm)$, where T is the number of PCG iterations, yielding much better scalability than sparse Cholesky on large graphs.

Remark 1 (Effective receptive field of PCG)**.** *With no preconditioner or a diagonal preconditioner, each solver iteration applies node-wise rescaling and multiplication by M. Since M mixes information across at most K hops, after T iterations the approximation's effective reach is at most $(T-1)K$ hops. Thus, while the exact Tikhonov operator is global (Proposition 3.4), a truncated PCG solve acts as a local polynomial approximation unless T is large. This reflects a solver limitation, not a message-passing bottleneck: each iterate remains linear in $\boldsymbol{x}$ without nonlinear compression. The true long-range sensitivity is governed by the learned Q (Proposition 3.5); T merely controls approximation accuracy. For tasks requiring long-range interactions, T should be set to large values (typically on the order of the graph diameter).*

To eliminate memory overhead and numerical instability, we use implicit differentiation rather than backpropagating through the solver iterations. Let $\boldsymbol{z} = \mathsf{M}^{-1}\mathsf{Q}\boldsymbol{x}$ be the forward output and $\bar{\boldsymbol{z}} = \partial\ell/\partial\boldsymbol{z}$ the incoming gradient from the loss ℓ. Differentiating $\mathsf{M}\boldsymbol{z} = \mathsf{Q}\boldsymbol{x}$ with respect to q_i (for instance) gives $\partial_{q_i}\mathsf{M}\boldsymbol{z} + \mathsf{M}\partial_{q_i}\boldsymbol{z} = \partial_{q_i}\mathsf{Q}\boldsymbol{x}$, *i.e.*: $\partial_{q_i}\boldsymbol{z} = \mathsf{M}^{-1}\partial_{q_i}\mathsf{Q}(\boldsymbol{x} - \boldsymbol{z})$. Applying the chain rule:

$$\frac{\partial\ell}{\partial q_i} = \bar{\boldsymbol{z}}^\top\mathsf{M}^{-1}\partial_{q_i}\mathsf{Q}(\boldsymbol{x} - \boldsymbol{z}) = u_i(x_i - z_i)$$

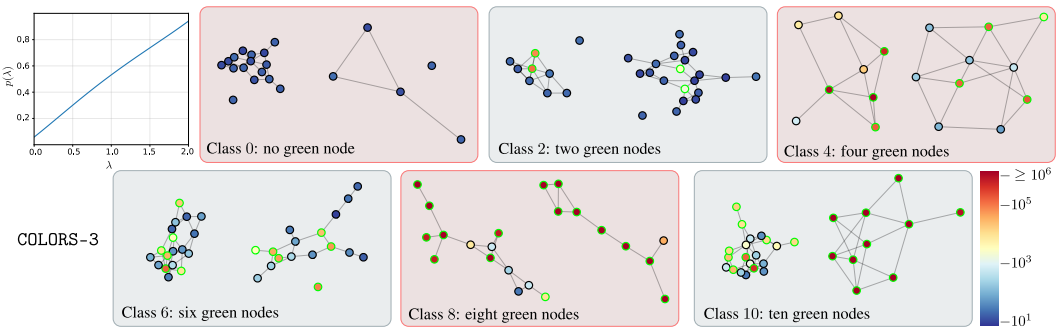


Figure 4: **Colors-3.** Node color encodes the learned q_i (log scale; blue: $q_i = 10$, red: $q_i \geq 10^6$); green circles mark ground truth. Top-left: learned $p(\lambda)$. Best run (in terms of validation loss) out of 5 runs (with different seeds), which all produced a test accuracy of 100%: perfect classification.

where $\boldsymbol{u} = \mathsf{M}^{-1}\bar{\boldsymbol{z}}$ is the adjoint variable. Computing $\boldsymbol{u}$ is the only expensive step and requires one additional PCG solve with the same matrix M. Gradients with respect to $\boldsymbol{x}$ and polynomial coefficients can also be expressed via $\boldsymbol{u}$. Thus, we run PCG in the forward pass without storing the computational graph, and use this single adjoint solve for all gradients in the backward pass. We refer to Appendix E for runtime experiments showing that our implementation of the Tikhonov layer incurs a computational overhead (only 2–4$\times$ slower, depending on the choice of PCG iterations T, compared to a comparable ChebNet baseline), maintains a comparable memory footprint, while providing interpretable results.

Bounding the q-values. The Q-network produces unconstrained scalar outputs $\tilde{q}_i \in \mathbb{R}$ for each node i. To map these to strictly positive values while ensuring numerical stability, we apply $q_i = \exp\big(\min(\tilde{q}_i, \log q_{\max})\big) + q_{\min}$ which clamps in log-space to prevent overflow in the exponential and adds a floor $q_{\min}$ to guarantee strict positivity. Despite this clamping, the condition number can still be substantial, making preconditioning essential for efficient CG convergence.

Enforcing $0 < p(\cdot) < 1$ on $[0, 2]$. This constraint is enforced via Bernstein polynomials, writing $p(\lambda) = \sum_{k=0}^{K} \mathrm{sigmoid}(\theta_k)\, b_{k,K}\big(\frac{\lambda}{2}\big)$ with $\{\theta_k\}$ real learnable parameters and Bernstein basis functions $b_{k,K}(t) = \binom{K}{k} t^k (1-t)^{K-k}$ that are all positive for $t \in [0, 1]$. Since $\mathrm{sigmoid}(\theta_k) \in (0, 1)$ and $\sum_k b_{k,K}(t) = 1$ on $t \in [0, 1]$, this parametrization naturally enforces $0 < p(\lambda) < 1$ for all $\lambda \in [0, 2]$.

4 Experiments

We evaluate the Tikhonov layer along two axes: *interpretability* (does the learned explanation $\{p, \{q_i\}\}$ reveal meaningful structure?) and *long-range propagation* (does the global receptive field help on tasks that require it?). To perform this evaluation, we need to investigate controlled synthetic datasets for which we have a known ground truth. We refer to Appendix F for experiments on two real-world datasets (ZINC and OGBG-MolHIV). Choice of hyperparameters is discussed in Appendix B. The code and environment used are available here: `https://github.com/ntrembla/Tikhonov_layer_code`.

4.1 Interpretability on controlled tasks

We first study a few synthetic datasets for which the ground-truth role of features vs. topology is known by construction, so that the learned explanation can be objectively assessed.

Colors-3 [18]. This dataset requires the model to count green nodes in small graphs (we only consider the relevant first subset of the full dataset). The task is solvable by node features alone, but standard message-passing GNNs can be biased by topology. The Tikhonov layer reaches a test accuracy of 100% and, as shown in Figure 4, learns large q_i's everywhere ($\gg 1$), effectively ignoring the

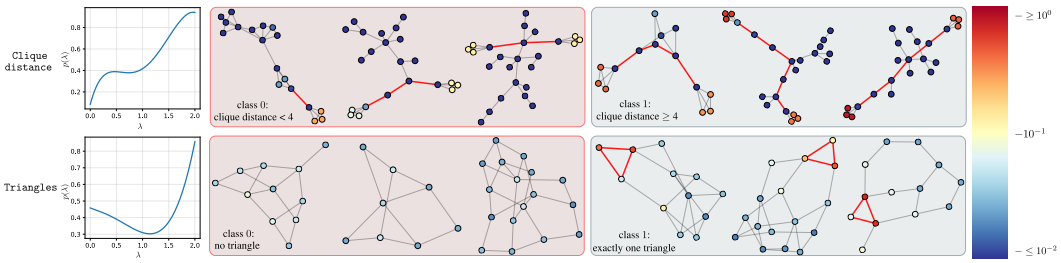


Figure 5: **Clique distance (top) and Triangles (bottom) datasets.** Node color encodes the learned q_i (log scale; blue: $q_i \leq 10^{-2}$, red: $q_i \geq 10^{0}$); red edges highlight ground-truth (triangle edges; shortest path between cliques). Left: learned polynomials p. Best (in terms of validation loss) of 5 runs. For `Clique distance`, the median test accuracy is 99.9% (the run shown has test accuracy of 100%). For `Triangles`, the median test accuracy is 95% (the run shown has test accuracy of 99.4%).

graph structure and nearly reducing to an MLP. More strikingly here in the plotted explanation [3], it assigns almost systematically higher q_i's to green nodes than to non-green ones: the explanation $\{q_i\}$ automatically singles out the nodes that carry the task-relevant feature.

Clique distance [29]. Each graph contains two cliques connected by a path of varying length. The task is to predict whether the inter-clique distance is below 4 (class 0) or at least 4 (class 1). Node features are uninformative (all-ones): the task is purely structural. As shown in Figure 5 (top row), the model learns an increasing $p(\lambda)$ and assigns very low q_i's to the path nodes connecting the cliques, while clique nodes are assigned much larger q_i's. The model thus exploits the topology of the connecting path to sense its length, with the low-q_i path nodes acting as topology-sensitive relays. The Tikhonov layer achieves a test accuracy of 100% ([29] reports GIN achieving 97%) and provides a meaningful explanation: cliques serve as anchors and diffusion propagates between them.

Triangles [29]. This dataset asks whether a graph contains exactly one triangle (class 1) or none (class 0). Node features are uninformative (all-ones), making the task purely structural. As shown in Figure 5 (bottom row), the model learns a mostly low-pass filter $p(\lambda)$ and small q_i's almost everywhere, hence relies primarily on topology. In class-1 graphs, however, triangle vertices receive noticeably larger q_i's, so the explanation pinpoints the motif without supervision. The model reaches 99.4% test accuracy, outperforming the best GAT baseline reported [29] for this dataset (94%, using degree and node-ID one-hot features) and nearly matching the Transformer accuracy (100%), while also revealing that the prediction is driven by identifying the triangle nodes.

CSBM (Contextual Stochastic Block Model). The CSBM [12] is a random graph model with node features, where topology and features are correlated with the same latent communities. Since it admits a known asymptotic recovery threshold, it provides a controlled setting to evaluate whether the learned explanation $\{q_i\}$ correctly identifies the source of the predictive signal (features, topology, or both); see Appendix C for details. The topology is sampled from a stochastic block model with signal strength λ, where edges are more likely within than across communities. Weak recovery from topology alone is possible iff $\lambda > 1$. Node features are drawn from a Gaussian mixture aligned with the labels, with signal strength $\mu/\sqrt{\gamma}$; weak recovery from features alone is possible iff $\mu/\sqrt{\gamma} > 1$. Joint weak recovery is possible [12] iff $\lambda^2 + \mu^2/\gamma > 1$. We introduce the following classification task: distinguish connected CSBM graphs (class 1) from a null model (class 0). For class 0, we sample a CSBM graph with the same λ, then randomly rewire its edges while preserving degree distribution and connectivity, removing community structure; features contain no signal ($\mu = 0$). We introduce this dataset to evaluate our method in a controlled yet challenging setting. For each $(\lambda, \mu/\sqrt{\gamma})$, we generate 2000 graphs (1000 per class) and use a 10-fold split. Graphs have $n = 100$ nodes split into two equal communities and average degree $d = 10$. We fix $\gamma = 25$.

[3] This is striking because the network does not need to do this. Indeed, other explanations found very large q-values ($\geq 10^3$) everywhere, not necessarily correlated with the green nodes. But large q-values everywhere just means the layer reduces to an MLP - and an MLP solves the task perfectly!

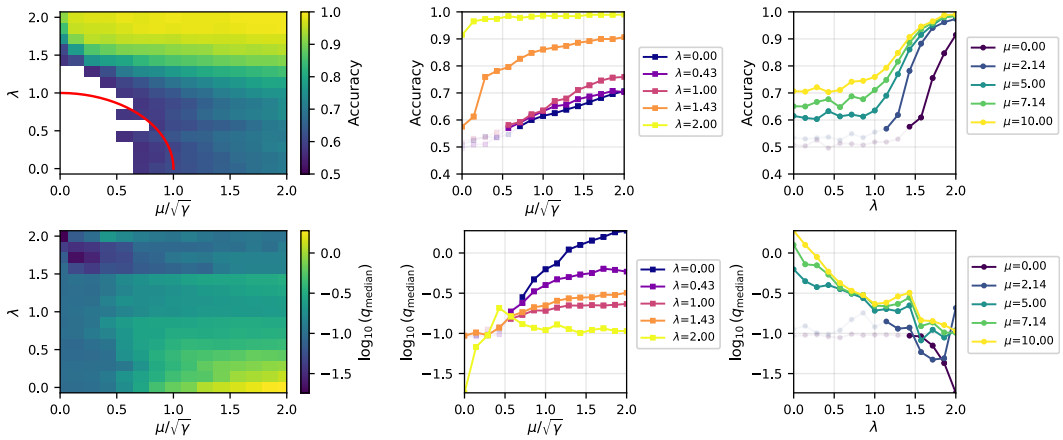


Figure 6: **CSBM dataset. (Top) test accuracy**. Left: accuracy on the parameter-space $(\lambda, \mu/\sqrt{\gamma})$. The red quarter-circle is the theoretical threshold at large n. When the accuracy < 0.56, it is left blank. Two right-top plots: same for some choices of λ and $\mu/\sqrt{\gamma}$. **(Bottom) median of q-values** for all nodes of all graphs (from both classes). Left: median of q-values on the parameter-space $(\lambda, \mu/\sqrt{\gamma})$. Two right-top plots: same for some choices of λ and $\mu/\sqrt{\gamma}$. For the four right-most plots: when accuracy is below 0.56, transparency is used. All plots are averaged over 6 runs. Learned polynomials are not plotted: they all converge close to $\lambda/2$ throughout the parameter space.

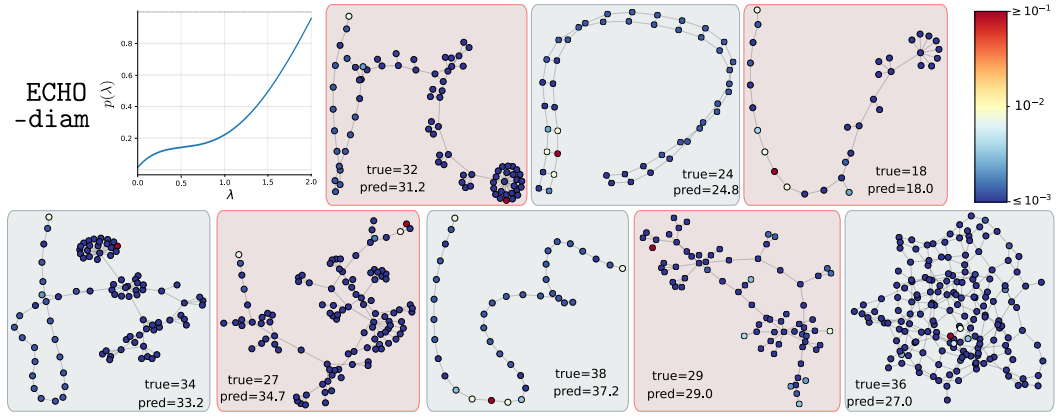


Figure 7: **ECHO-diam benchmark.** The learned polynomial along with the learned q-values plotted on 8 example graphs (log scale; blue: $q_i \leq 10^{-3}$, red: $q_i \geq 10^{-1}$). The true diameter is written, along with the prediction of the Tikhonov layer. Best (in terms of validation loss) out of 5 runs. The median test MAE is 1.20 (the run shown has a test MAE of 1.17). The MAE is the average error of prediction of the diameter.

As shown in Figure 6, the model correctly tracks the theoretical threshold[4]: accuracy is high above the curve $\lambda^2 + \mu^2/\gamma = 1$ and drops to chance below it. The region at small μ and just over the threshold $\lambda > 1$ is difficult to recover. In fact, this region is well-known to be challenging, see [22, 9]. Of particular interest for us here, the median of the learned q values (bottom row) transitions smoothly from large values (feature-dominated regime, large $\mu/\sqrt{\gamma}$) to small values (topology-dominated regime, large λ), with an intermediate range in the mixed regime. The explanation faithfully detects the underlying signal structure, with no direct supervision on which source to use.

[4]The theoretical threshold is sharp (it is a phase transition) for n and p (the node features' dimension) growing to infinity with $\gamma = n/p$ constant. For finite n and p, slightly correlated detections inside the red quarter-circle are to be expected.

4.2 Resistance to oversquashing

We evaluate on the ECHO-diam task from the ECHO benchmark [23]: it is a graph regression task and consists in predicting the diameter of a synthetic graph. This requires capturing the longest shortest path, which is inherently a global property. Results are shown in Figure 7. The learned filter is low-pass, and the Q-network assigns large values to peripheral nodes (leaf-like or degree-one endpoints) and small values to interior nodes. Interior nodes with small q undergo heavy spectral smoothing and carry a nearly uniform signal; peripheral nodes with large q effectively anchor this smooth field with their own features. Global pooling aggregates this contrast into a prediction: the more spread-out the periphery, the higher the diameter. Intuitively, the model identifies the endpoints of the longest path and uses their amplitude contrast with the interior as a proxy for the number of nodes lying between them. The test MAE is 1.17, between the best deep GNN (~ 1.6) and the best transformer (1.05), as reported in [23], with the added benefit of a built-in explanation of how the regression is performed.

5 Concluding remarks

We propose an original take on the problem of interpretability in GNNs. By severely constraining the form of the layer, we force the network to learn by-products (the node importance q-values and the graph filter $p(\lambda)$) that give us insight into how the network is solving the task. Interpretation is a difficult endeavour. Our method encounters two common drawbacks of the literature: i) some explanations can still be hard to understand for us humans (see, for instance, the second explanation for ECHO-diam discussed in Appendix D), ii) some tasks have many different explanations (see, for instance, the second explanation for Triangles discussed in Appendix D). Further research could be to add an auxiliary loss to favor some explanations rather than others. A limitation of the Tikhonov layer is its computational cost in cases where the task requires PCG to approximate long-distance interactions encoded in R (as one needs to increase the number of PCG iterations). Promising solutions come from random numerical linear algebra that are able to obtain Monte-Carlo approximations of $\mathsf{R}\boldsymbol{x}$ (for now only in the case where $p(\mathsf{L}) = \mathsf{L}$), without computing R, via carefully designed random walks on the graph [4], for a total cost linear in n.

Acknowledgments and Disclosure of Funding

This work was partially funded by the European Union (Marie Curie project Rand4TrustPool, 101148828), the Région Auvergne-Rhône-Alpes project TrustGNN, the ANR project GRANOLA (ANR-21-CE48-0009), and the Norwegian Research Council project 345017: *RELAY: Relational Deep Learning for Energy Analytics*. We gratefully thank NVIDIA Corporation for donating the GPUs used in this project.

References

[1] U. Alon and E. Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021.

[2] S. Azzolin, A. Longa, S. Teso, and A. Passerini. Reconsidering faithfulness in regular, self-explainable and domain invariant gnns. In *International Conference on Learning Representations*, volume 2025, pages 48644–48677, 2025.

[3] S. Azzolin, S. Teso, B. Lepri, A. Passerini, and S. Malhotra. GNN explanations that do not explain and how to find them. In *The Fourteenth International Conference on Learning Representations*, 2026.

[4] S. Barthelmé, F. Castell, A. Gaudillière, C. Mélot, M. Quattropani, and N. Tremblay. Spectrum estimation through Kirchhoff random forests. *arXiv preprint arXiv:2507.19164*, 2025.

[5] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi. Graph neural networks with convolutional arma filters. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3496–3507, 2021.

[6] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *International Conference on Learning Representations*, 2014.

[7] S. Chen, A. Sandryhaila, G. Lederman, Z. Wang, J. M. F. Moura, P. Rizzo, J. Bielak, J. H. Garrett, and J. Kovačevic. Signal inpainting on graphs via total variation minimization. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[8] E. Dai and S. Wang. Towards self-explainable graph neural network. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 302–311, 2021.

[9] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Phys. Rev. E*, 84(6):066106, Dec. 2011.

[10] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, 2016.

[11] S. Demko, W. F. Moss, and P. W. Smith. Decay rates for inverses of band matrices. *Mathematics of computation*, 43(168):491–499, 1984.

[12] Y. Deshpande, S. Sen, A. Montanari, and E. Mossel. Contextual Stochastic Block Models. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[13] V. P. Dwivedi, C. K. Joshi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 2023.

[14] J. Gasteiger, A. Bojchevski, and S. Günnemann. Predict then propagate: Graph neural networks meet personalized PageRank. In *International Conference on Learning Representations*, 2019.

[15] J. Gasteiger, S. Weißenberger, and S. Günnemann. Diffusion improves graph learning. *Advances in neural information processing systems*, 32, 2019.

[16] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems*, 2020.

[17] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[18] B. Knyazev, G. W. Taylor, and M. Amer. Understanding attention and generalization in graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[19] Q. Li, Z. Han, and X.-M. Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[20] A. Longa, S. Azzolin, G. Santin, G. Cencetti, P. Lio, B. Lepri, and A. Passerini. Explaining the explainers in graph neural networks: a comparative study. *ACM Computing Surveys*, 57(5):120:1–120:37, 2025.

[21] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang. Parameterized explainer for graph neural network. *Advances in neural information processing systems*, 33:19620–19631, 2020.

[22] L. Massoulié. Community detection thresholds and the weak Ramanujan property. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 694–703. ACM, 2014.

[23] L. Miglior, M. Tolloso, A. Gravina, and D. Bacciu. Can you hear me now? a benchmark for long-range graph propagation. In *International Conference on Learning Representations*, 2026.

[24] S. K. Narang, A. Gadde, and A. Ortega. Signal processing techniques for interpolation in graph structured data. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

[25] A. Ortega. *Introduction to graph signal processing*. Cambridge University Press, 2022.

[26] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, and H. Hoffmann. Explainability methods for graph convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10772–10781, 2019.

[27] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

[28] A. J. Smola and R. Kondor. Kernels and regularization on graphs. In *Learning theory and kernel machines: 16th annual conference on learning theory and 7th kernel workshop, COLT/kernel 2003, Washington, DC, USA, august 24-27, 2003. Proceedings*, pages 144–158. Springer, 2003.

[29] A. Tolmachev, A. Sakai, M. Todoriki, and K. Maruhashi. Bermuda triangles: GNNs fail to detect simple topological structures. In *ICLR 2021 Workshop on Geometrical and Topological Representation Learning*, 2021.

[30] N. Tremblay, P. Gonçalves, and P. Borgnat. Design of graph filters and filterbanks. In *Cooperative and graph signal processing*, pages 299–324. Elsevier, 2018.

[31] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[32] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. GNNexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[33] Z. Zhang, Q. Liu, H. Wang, C. Lu, and C. Lee. ProtGNN: Towards self-explaining graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 9127–9135, 2022.

[34] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16, 2003.

[35] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *International conference on Machine learning*, pages 912–919, 2003.

A Proofs

Lemma A.1. *For a fixed graph $\mathcal{G}$, $\mathsf{P} = p(\mathsf{L})$ has complete K-hop support for generic polynomial coefficients. Equivalently, the set of coefficients for which the property fails has Lebesgue measure zero.*

Proof. Write $\mathsf{L} = \mathsf{I} - \tilde{\mathsf{A}}$ with $\tilde{\mathsf{A}} = \mathsf{D}^{-1/2}\mathsf{A}\mathsf{D}^{-1/2}$. *Step 1:* fix two nodes $i \neq j$ and denote by $0 < d_{\mathcal{G}}(i,j) = m \leq K$ their hop distance. Since no walk of length $\ell < m$ connects i to j, $(\tilde{\mathsf{A}}^{\ell})_{ij} = 0$ for all $\ell < m$. The binomial theorem then boils down to

$$(\mathsf{L}^m)_{ij} = \sum_{\ell=0}^{m} \binom{m}{l} (-1)^{\ell} (\tilde{\mathsf{A}}^{\ell})_{ij} = (-1)^m (\tilde{\mathsf{A}}^m)_{ij} \neq 0$$

as $\tilde{\mathsf{A}}^m$ counts weighted walks of length m, of which at least one exists between i and j. Thus $\mathsf{P}_{ij}(\boldsymbol{\alpha}) = \sum_{k=0}^{K} \alpha_k (\mathsf{L}^k)_{ij}$ is a non-trivial linear form in $\boldsymbol{\alpha}$, so its zero set is a hyperplane in $\mathbb{R}^{K+1}$, hence of measure zero. *Step 2:* If $i = j$, hence $m = 0$, one has: $(\mathsf{L}^m)_{ii} = 1 \neq 0$ and $\mathsf{P}_{ii}(\boldsymbol{\alpha}) = \sum_{k=0}^{K} \alpha_k (\mathsf{L}^k)_{ii}$ is also a non-trivial linear form in $\boldsymbol{\alpha}$: its zero set is also of measure zero. *Step 3:* The conclusion follows by taking a finite union over all pairs (i,j) with $d_{\mathcal{G}}(i,j) \leq K$. □

Proof of Proposition 3.1. *(i)* Recall that the spectrum of L is included in $[0,2]$ such that $p(\mathsf{L}) \succ 0$. As $\mathsf{Q} \succ 0$, their sum is also positive-definite: the inverse exists.

(ii) Set $\mathsf{M} = \mathsf{Q}^{-1/2} p(\mathsf{L}) \mathsf{Q}^{-1/2} \succ 0$. One has $\mathsf{R} = (p(\mathsf{L}) + \mathsf{Q})^{-1}\mathsf{Q} = \mathsf{Q}^{-1/2}(\mathsf{M} + \mathsf{I})^{-1}\mathsf{Q}^{1/2}$, so R is similar to the symmetric matrix $(\mathsf{M} + \mathsf{I})^{-1}$ and therefore shares its eigenvalues. Since $\mathsf{M} \succ 0$, every eigenvalue of $\mathsf{M} + \mathsf{I}$ is strictly greater than 1, hence every eigenvalue of $(\mathsf{M} + \mathsf{I})^{-1}$—and of R—lies strictly in $(0,1)$.

(iii) Note that $\mathsf{R}^{\top} = \mathsf{Q}(\mathsf{P} + \mathsf{Q})^{-1}$. $\mathsf{R} = \mathsf{R}^{\top} \iff (\mathsf{P} + \mathsf{Q})^{-1}\mathsf{Q} = \mathsf{Q}(\mathsf{P} + \mathsf{Q})^{-1} \iff \mathsf{P}\mathsf{Q} = \mathsf{Q}\mathsf{P} \iff \forall (i,j)\ (q_i - q_j)P_{ij} = 0$ as Q is diagonal. ($\Longrightarrow$) Let R be symmetric. For any adjacent nodes i, j, $d_{\mathcal{G}}(i,j) = 1 \leq K$ (as $K > 0$). Complete K-hop support implies $P_{ij} \neq 0$, forcing $q_i = q_j$. Propagating this along paths implies $q_i = q_j$ whenever i and j belong to the same connected component. ($\Longleftarrow$) Conversely, suppose q is constant on each connected component. If i, j are in the same component, $q_i - q_j = 0$. If they are in different components, $d_{\mathcal{G}}(i,j) = \infty > K$, so $P_{ij} = 0$ by localization. In both cases, $(q_i - q_j)P_{ij} = 0$, so R is symmetric. □

Lemma A.2. *Consider the matrix $\mathsf{M} = p(\mathsf{L})\mathsf{R}$ and let $\boldsymbol{x}$ be a vector. Let $p_{max} = \max_k p(\lambda_k)$. The following upper bound is valid: $|[\mathsf{M}\boldsymbol{x}]_i| \leq p_{max}\|x\|_2$.*

Proof of Lemma. One has $\forall i$, $|[\mathsf{M}\boldsymbol{x}]_i| \leq \|\mathsf{M}\|_2\|x\|_2$. Now, $p(\mathsf{L})$ being invertible, M can be re-written as $\mathsf{M} = \left(\mathsf{Q}^{-1} + p(\mathsf{L})^{-1}\right)^{-1}$: it is symmetric and positive-definite. As such, one has $\|\mathsf{M}\|_2 = \lambda_{\max}(\mathsf{M}) = 1/\lambda_{\min}(\mathsf{Q}^{-1} + p(\mathsf{L})^{-1})$. To upper-bound this, we need to lower-bound the denominator. Noting that $\lambda_{\min}(\mathsf{Q}^{-1} + p(\mathsf{L})^{-1}) \geq \lambda_{\min}(p(\mathsf{L})^{-1}) = 1/\lambda_{\max}(p(\mathsf{L})) = 1/p_{\max}$ finishes the proof. □

Proof of Proposition 3.2. Feature preservation. Let $\mathsf{M} = p(\mathsf{L})\mathsf{R}$. Left-multiply R by $\mathsf{Q}^{-1}(p(\mathsf{L}) + \mathsf{Q})$ and obtain $\mathsf{R} - \mathsf{I} = -\mathsf{Q}^{-1}\mathsf{M}$. Thus $\forall i$:

$$|(\mathsf{R}\boldsymbol{x})_i - x_i| = |[\mathsf{Q}^{-1}\mathsf{M}\boldsymbol{x}]_i| = q_i^{-1}|[\mathsf{M}\boldsymbol{x}]_i| \leq q_i^{-1}p_{\max}\|\boldsymbol{x}\|_2$$

by Lemma A.2.

Graph harmonicity. Left-multiplying $\mathsf{R}\boldsymbol{x}$ by $p(\mathsf{L}) + \mathsf{Q}$, one obtains: $p(\mathsf{L})\boldsymbol{z} = \mathsf{Q}(\boldsymbol{x} - \boldsymbol{z})$, such that $|[p(\mathsf{L})\boldsymbol{z}]_i| = q_i|x_i - z_i|$. Now, one has:

$$q_i|x_i - z_i| = \sqrt{q_i}\sqrt{q_i(x_i - z_i)^2}.$$

Denote by F the functional $F(\boldsymbol{u}) = \sum_j q_j(x_j - u_j)^2 + \boldsymbol{u}^\top p(\mathsf{L})\boldsymbol{u}$. One has:

$$q_i(x_i - z_i)^2 \leq \sum_j q_j(x_j - z_j)^2 \leq F(\boldsymbol{z})$$

as $\boldsymbol{z}^\top p(\mathsf{L})\boldsymbol{z} \geq 0$ (as $p(\mathsf{L}) \succ 0$). By definition, $\boldsymbol{z}$ is the minimizer of the functional $F(\boldsymbol{u})$ such that: $F(\boldsymbol{z}) \leq F(\boldsymbol{x}) = \boldsymbol{x}^\top p(\mathsf{L})\boldsymbol{x} \leq p_{\max}\|\boldsymbol{x}\|_2^2$. Putting everything back together, we finish the proof:

$$|[p(\mathsf{L})\boldsymbol{z}]_i| = q_i|x_i - z_i| \leq \sqrt{q_i}\sqrt{p_{\max}}\|\boldsymbol{x}\|_2.$$

□

Proof of Proposition 3.3. Density of the homogeneous family. Fix a continuous target $h : [0, 2] \to (0, 1)$ and $\varepsilon > 0$. Note that both constants

$$c := \min_\lambda \frac{h(\lambda)}{1 - h(\lambda)} > 0 \quad \text{and} \quad \mu := \min_\lambda \frac{1 - h(\lambda)}{h(\lambda)} > 0$$

are strictly positive (note that $c\mu \leq 1$). For a density argument it suffices to handle $\varepsilon < \mu$, so assume this without loss of generality. Choose any q satisfying

$$0 < q < \frac{c}{1 + c\varepsilon};$$

this is possible since $c > 0$. Set $f(\lambda) := q(1 - h(\lambda))/h(\lambda)$, so that $h(\lambda) = q/(f(\lambda) + q)$. Then $\min_\lambda f = q\mu$ and $\max_\lambda f = q/c$, giving

$$\min_\lambda f - q\varepsilon = q(\mu - \varepsilon) > 0 \quad \text{and} \quad 1 - \max_\lambda f - q\varepsilon = 1 - \frac{q}{c} - q\varepsilon = 1 - q\left(\tfrac{1}{c} + \varepsilon\right) > 0,$$

where the second inequality uses $q < c/(1 + c\varepsilon)$. Hence $f(\lambda) \in (0, 1)$ for all λ. By the Weierstrass approximation theorem there exists a polynomial p with $\|p - f\|_\infty < q\varepsilon$. The two inequalities above guarantee $0 < p(\lambda) < 1$ on $[0, 2]$. The homogeneous filter $g_q(\lambda) = q/(p(\lambda) + q)$ then satisfies

$$|g_q(\lambda) - h(\lambda)| = \frac{q|f(\lambda) - p(\lambda)|}{(p(\lambda) + q)(f(\lambda) + q)} \leq \frac{q \cdot q\varepsilon}{q^2} = \varepsilon$$

uniformly on $[0, 2]$, establishing density.

Strict generalization via non-commutativity. Let $\mathsf{P} = p(\mathsf{L})$. Since

$$\mathsf{R}^{-1} = \mathsf{Q}^{-1}(\mathsf{P} + \mathsf{Q}) = \mathsf{Q}^{-1}\mathsf{P} + \mathsf{I},$$

and since P commutes with L and is invertible, we have

$$[\mathsf{R}, \mathsf{L}] = 0 \iff [\mathsf{R}^{-1}, \mathsf{L}] = 0 \iff [\mathsf{Q}^{-1}, \mathsf{L}]P = 0 \iff [\mathsf{Q}, \mathsf{L}] = 0.$$

where $[\mathsf{R}, \mathsf{L}] = \mathsf{RL} - \mathsf{LR}$ denotes the commutator. For diagonal Q,

$$([\mathsf{Q}, \mathsf{L}])_{ij} = (q_i - q_j)L_{ij},$$

so Q commutes with L iff $q_i = q_j$ on every edge, i.e., iff Q is constant on each connected component. Thus, on a connected graph, any nonconstant Q gives an operator R that does not commute with L.

Finally, suppose that $\mathsf{R} = g(\mathsf{L})$ for some scalar function g. Then R commutes with L, so by the previous paragraph $\mathsf{Q} = q_c\mathsf{I}$ on each connected component G_c. On the zero-eigenvector supported on G_c, R acts by the scalar

$$\frac{q_c}{p(0) + q_c}.$$

But $g(\mathsf{L})$ acts as the same scalar $g(0)$ on the whole nullspace of L. Since $p(0) > 0$, the map

$$q \mapsto \frac{q}{p(0) + q}$$

is strictly increasing, so all q_c must be equal. Hence $\mathsf{Q} = q\mathsf{I}$. Therefore, R can be written as a scalar spectral filter $g(\mathsf{L})$ only in the homogeneous case. □

Proof of Proposition 3.4. (i) Define $\mathsf{M} = p(\mathsf{L}) + \mathsf{Q}$. Since $\mathsf{R}_{ij} = (\mathsf{M}^{-1})_{ij}\, q_j$ with $q_j > 0$, the zero pattern of R is that of M^{-1}. Both $p(\mathsf{L})$ and Q are block-diagonal with respect to connected components, hence so are M and M^{-1}, giving $\mathsf{R}_{ij} = 0$ between components.

(ii) Increasing degree-one case. Write the increasing degree-one polynomial as $p(\lambda) = a + b\lambda$ with $b > 0$ (and necessarily $a > 0$ as $0 < p(\lambda) < 1$ must hold at $\lambda = 0$), so that $\mathsf{M} = p(\mathsf{L}) + \mathsf{Q} = a\mathsf{I} + b\mathsf{L} + \mathsf{Q} = b(\mathsf{L} + \mathsf{Q}')$ with $\mathsf{Q}' = \frac{1}{b}(\mathsf{Q} + a\mathsf{I}) \succ 0$. On each connected component, $\mathsf{L} + \mathsf{Q}'$ is a symmetric positive definite matrix with non-positive off-diagonal entries, and it is irreducible when the component has more than one node. Hence, by the standard inverse-positivity of irreducible nonsingular M-matrices,

$$\left((\mathsf{L} + \mathsf{Q}')^{-1}\right)_{ij} > 0$$

for all i, j in the same connected component. The singleton case is immediate. Since:

$$\mathsf{R} = \mathsf{M}^{-1}\mathsf{Q} = \frac{1}{b}(\mathsf{L} + \mathsf{Q}')^{-1}\mathsf{Q},$$

we get, for every pair i, j within the same connected component:

$$\mathsf{R}_{ij} = \frac{q_j}{b}\left((\mathsf{L} + \mathsf{Q}')^{-1}\right)_{ij} > 0.$$

(iii) Other cases. Let us first treat the simple case $i = j$. One has: $\mathsf{R}_{ii} = \left(\mathsf{M}^{-1}\right)_{ii} q_i$ with $\mathsf{M} = p(\mathsf{L}) + \mathsf{Q}$. As M is positive definite, so is M^{-1}. A well-known fact of positive definite matrices is that their diagonal entries are always positive. As q_i is also positive, we have that $\forall i, \mathsf{R}_{ii} > 0$.

Now, let $\mathcal{C}$ be the set of all ordered pairs (i, j) of distinct vertices belonging to the same connected component of $\mathcal{G}$. Fix a pair $(i, j) \in \mathcal{C}$. Let the entries $q_1, \ldots, q_n$ of the diagonal matrix Q be treated as independent formal variables. By Cramer's rule, the Tikhonov entry is given by

$$\mathsf{R}_{ij} = \left(\mathsf{M}^{-1}\right)_{ij} q_j = \frac{\mathsf{C}_{ji}(\mathsf{M})\, q_j}{\det(\mathsf{M})},$$

where $\mathsf{C}_{ji}(\mathsf{M})$ is the (j, i)-th cofactor of M. By the Leibniz formula for determinants, both $\det(\mathsf{M})$ and the cofactors $\mathsf{C}_{ji}(\mathsf{M})$ are multivariate polynomials in $\mathbb{R}[q_1, \ldots, q_n]$. Since $\mathsf{P} \succ 0$ and $\mathsf{Q} \succ 0$, M is positive definite, ensuring $\det(\mathsf{M}) > 0$ for all positive diagonal Q.

The cofactor is given by $\mathsf{C}_{ji}(\mathsf{M}) = (-1)^{i+j} \det(\mathsf{M}_{\setminus j, \setminus i})$, where $\mathsf{M}_{\setminus j, \setminus i}$ denotes the submatrix of M obtained by deleting row j and column i. This submatrix is indexed by rows $\mathcal{V} \setminus \{j\}$ and columns $\mathcal{V} \setminus \{i\}$, and each variable q_k ($k \neq i, j$) appears exactly once, in the entry corresponding to the original diagonal position (k, k) of M, as part of the linear term $\mathsf{P}_{kk} + q_k$.

Since i and j lie in the same connected component, there exists a shortest path in the *K-hop graph* (*i.e.*, the graph whose edges connect pairs of vertices at graph distance at most K) from i to j. Let $i = v_0, v_1, \ldots, v_m = j$ be such a shortest path, so that $d_{\mathcal{G}}(v_r, v_{r+1}) \leq K$ for all $0 \leq r < m$, and m is minimal. By the complete K-hop support hypothesis, each off-diagonal entry along this path satisfies $\mathsf{P}_{v_r, v_{r+1}} \neq 0$.

Moreover, by minimality of the path, for any r, s with $|r - s| > 1$ we have $d_{\mathcal{G}}(v_r, v_s) > K$: otherwise, replacing the sub-path $v_r, \ldots, v_s$ by the single hop $v_r \to v_s$ would yield a strictly shorter

path, contradicting minimality. By K-localization, it follows that $\mathsf{P}_{v_r,v_s} = 0$ whenever $|r-s| > 1$. We now exhibit a monomial in the Leibniz expansion of $\det(\mathsf{M}_{\setminus j,\setminus i})$ that is *not canceled* by any other term. Consider the permutation

$$\sigma^* : \quad v_r \longmapsto v_{r+1} \quad (0 \le r \le m-1), \qquad k \longmapsto k \quad (k \in \mathcal{V} \setminus \{v_0, \dots, v_m\}),$$

which is a valid bijection from $\mathcal{V} \setminus \{j\} = \mathcal{V} \setminus \{v_m\}$ to $\mathcal{V} \setminus \{i\} = \mathcal{V} \setminus \{v_0\}$. This permutation selects: the off-diagonal constants $\mathsf{P}_{v_r,v_{r+1}}$ for the path vertices; and the linear diagonal terms $(\mathsf{P}_{kk} + q_k)$ for all $k \in \mathcal{V} \setminus \{v_0, \dots, v_m\}$. Choosing the variable q_k from each such diagonal factor yields the monomial

$$\gamma_{ij} \cdot \prod_{k \notin \{v_0,\dots,v_m\}} q_k, \qquad \text{where} \quad \gamma_{ij} = \operatorname{sgn}(\sigma^*)(-1)^{i+j} \prod_{r=0}^{m-1} \mathsf{P}_{v_r,v_{r+1}} \neq 0.$$

This monomial involves precisely the variable set $\mathcal{S} = \{q_k \mid k \notin \{v_0, \dots, v_m\}\}$ and has degree $|\mathcal{S}| = n - (m+1)$.

We claim no other permutation $\tau \neq \sigma^*$ contributes a term containing $\prod_{k \in \mathcal{S}} q_k$. Any such τ must also fix all $k \in \mathcal{V} \setminus \{v_0, \dots, v_m\}$ (to select the diagonal factor $\mathsf{P}_{kk} + q_k$ and thus include q_k), meaning τ must act as a bijection from $\{v_0, \dots, v_{m-1}\}$ to $\{v_1, \dots, v_m\}$ on the path vertices. But since $P_{v_r,v_s} = 0$ for $|r-s| > 1$, the only bijection from $\{v_0, \dots, v_{m-1}\}$ to $\{v_1, \dots, v_m\}$ that yields a nonzero product $\prod_r \mathsf{M}_{v_r,\tau(v_r)}$ is the forward shift σ^* itself. In other words, σ^* is the unique permutation contributing the monomial $\gamma_{ij} \prod_{k \in \mathcal{S}} q_k$: this term cannot be canceled and $C_{ji}(\mathsf{M})$ is not identically zero as a polynomial in $\mathbb{R}[q_1, \dots, q_n]$.
For each pair $(i,j) \in \mathcal{C}$, define the exceptional set

$$\mathfrak{E}_{ij} = \left\{ \mathsf{Q} \in \mathbb{R}^n_{>0} \;\middle|\; \mathsf{C}_{ji}(\mathsf{M}) = 0 \right\},$$

which is the zero-set of a nonzero polynomial, hence an algebraic variety of Lebesgue measure zero in $\mathbb{R}^n$. Define the global exceptional set

$$\mathfrak{E} = \bigcup_{(i,j) \in \mathcal{C}} \mathfrak{E}_{ij}.$$

Since $\mathcal{G}$ is finite, $|\mathcal{C}| \le n(n-1)$, so $\mathfrak{E}$ is a finite union of measure-zero sets and thus has Lebesgue measure zero.

For any positive diagonal matrix $\mathsf{Q} \notin \mathfrak{E}$, we have $\mathsf{C}_{ji}(\mathsf{M}) \neq 0$ for all $(i,j) \in \mathcal{C}$ simultaneously. Since $q_j > 0$ and $\det(\mathsf{M}) > 0$, it follows that

$$\mathsf{R}_{ij} = \frac{\mathsf{C}_{ji}(\mathsf{M})\, q_j}{\det(\mathsf{M})} \neq 0$$

for every pair (i,j) in the same connected component, completing the proof. □

Proof of Proposition 3.5. Set $\mathsf{S} = \mathsf{I} + \mathsf{Q}^{-1/2} p(\mathsf{L})\, \mathsf{Q}^{-1/2}$ such that $\mathsf{Q}^{1/2} \mathsf{S}\, \mathsf{Q}^{1/2} = p(\mathsf{L}) + \mathsf{Q}$, giving $\mathsf{R} = \mathsf{Q}^{-1/2} \mathsf{S}^{-1} \mathsf{Q}^{1/2}$, hence $|\mathsf{R}_{ij}| = \sqrt{q_j/q_i}\; |(\mathsf{S}^{-1})_{ij}|$. The matrix S is SPD and is K-localized: $\mathsf{S}_{ij} = 0$ whenever $d_{\mathcal{G}}(i,j) > K$. Write $d = d_{\mathcal{G}}(i,j)$. By the Demko–Moss–Smith bound [11],

$$|(\mathsf{S}^{-1})_{ij}| \le 2 \|\mathsf{S}^{-1}\|_2\, \rho_0^{d/K}$$

where $\rho_0 = (\sqrt{\kappa} - 1)/(\sqrt{\kappa} + 1)$ and $\kappa = \operatorname{cond}(\mathsf{S})$. Using $\operatorname{arctanh}(x) \ge x$ for $x \in (0,1)$, $\rho_0 \le e^{-2/\sqrt{\kappa}}$. Finally, since $\mathsf{S} \succeq \mathsf{I}$, $\|\mathsf{S}^{-1}\|_2 \le 1$, yielding $|(\mathsf{S}^{-1})_{ij}| \le 2e^{-\frac{2}{K\sqrt{\kappa}} d}$ and the result. □

Proof of Proposition 3.6. Write $\mathsf{P} = p(\mathsf{L})$. Since $p(\lambda) > 0$ on $[0,2]$, we have $P \succ 0$, hence P is invertible. Moreover $\mathsf{R}^{-1} = \mathsf{Q}^{-1}(P + \mathsf{Q}) = \mathsf{Q}^{-1} P + \mathsf{I}$. If two positive diagonal matrices $\mathsf{Q}_1, \mathsf{Q}_2$ produce the same operator R, then $\mathsf{Q}_1^{-1} P + \mathsf{I} = \mathsf{Q}_2^{-1} P + \mathsf{I}$, so $(\mathsf{Q}_1^{-1} - \mathsf{Q}_2^{-1}) P = 0$. Since P is invertible, $\mathsf{Q}_1^{-1} = \mathsf{Q}_2^{-1}$, hence $\mathsf{Q}_1 = \mathsf{Q}_2$. □

Proof of Proposition 3.7. Let $\mathsf{P}_j = p_j(\mathsf{L})$ and R be the common operator. Inverting $\mathsf{R} = (\mathsf{P}_j + \mathsf{Q}_j)^{-1} \mathsf{Q}_j$ we obtain $\mathsf{R}^{-1} = \mathsf{Q}_j^{-1} \mathsf{P}_j + \mathsf{I}$, hence $\mathsf{Q}_1^{-1} \mathsf{P}_1 = \mathsf{Q}_2^{-1} \mathsf{P}_2$. Let

$$\Delta = \mathsf{Q}_2 \mathsf{Q}_1^{-1} = \operatorname{diag}(\delta_1, \dots, \delta_n), \quad \delta_i > 0.$$

Then $\mathsf{P}_2 = \Delta\,\mathsf{P}_1$.

Since P_1 is invertible (as $p_1(\lambda) > 0$ for $\lambda \in [0, 2]$), $\Delta = \mathsf{P}_2\mathsf{P}_1^{-1}$. Both P_2 and P_1^{-1} commute with L, hence Δ commutes with L. Since Δ is diagonal, $(\delta_i - \delta_j)L_{ij} = 0$ for all i, j, so δ_i is constant on each connected component. Write this constant as α_c.

Now let $\boldsymbol{u}_c \neq \mathbf{0}$ be an eigenvector of L associated to eigenvalue 0, supported on component $\mathcal{G}_c$. Since $\mathsf{P}_j = p_j(\mathsf{L})$,

$$\mathsf{P}_2\boldsymbol{u}_c = p_2(0)\boldsymbol{u}_c, \qquad \mathsf{P}_1\boldsymbol{u}_c = p_1(0)\boldsymbol{u}_c.$$

Using $\mathsf{P}_2 = \Delta\mathsf{P}_1$ on this component gives $p_2(0)\boldsymbol{u}_c = \alpha_c p_1(0)\boldsymbol{u}_c$. Since $p_1(0) > 0$, all $\alpha_c = p_2(0)/p_1(0)$ are equal. Denoting the common value by α, we obtain

$$\mathsf{Q}_2 = \alpha\mathsf{Q}_1, \qquad \mathsf{P}_2 = \alpha\mathsf{P}_1.$$

□

Proof of Proposition 3.8. Equal pre-activation outputs up to a permutation of channel blocks means $\mathsf{R}'_j\mathsf{H} = \mathsf{R}_{\pi(j)}\mathsf{H}$ for some permutation π. Since H has rank n, it has a right inverse. Hence $\mathsf{R}'_j = \mathsf{R}_{\pi(j)}$ for each j. Applying Proposition 3.7 to each channel gives $\mathsf{Q}'_j = \alpha_j\,\mathsf{Q}_{\pi(j)}$ with $p'_j(\mathsf{L}) = \alpha_j\,p_{\pi(j)}(\mathsf{L})$. □

B Choice of hyperparameters

Common settings. We use the Adam optimizer. We use one channel with a degree $K = 5$ for the polynomial p. Skip connections are implemented in the Q-network. The Q-network ends with a 2-layer MLP head. $q_{\text{min}} = 1\text{e}{-10}$, $q_{\text{max}} = 1\text{e}10$. No dropout.

How the other hyperparameters were chosen. For all experiments except ECHO-diam, hyperparameters were found by sweeping (not exhaustively however – due to computation time constraints) (i) for the Q-network: ChebNet/ARMA, hidden dimension $\in [4, 8, 16]$, number of layers $\in [3, 5, 10, 15, 20]$, chebychev order $\in [3, 10]$ for ChebNet, number of inner layers and number of stacks $\in [3, 5]$ for ARMA, the initialization[5] of $q \in [0.01, 0.1, 1.0]$ (ii) for the initial shape of the polynomial: close to $\lambda/2$ or close to flat; (iii) for the rest: hidden dimension $\in [4, 8, 16]$, patience $\in [50, 150, 250]$, learning rate $\in [5\text{e}{-4}, 1\text{e}{-3}, 5\text{e}{-3}]$. Batch sizes were chosen from the original papers (for the CSBM dataset, we tried 256 and 512). For the global pooling: mean, sum (both without normalization), mean+max+sum with layer normalization, and sum+sumsq (the sum of squares, only for CSBM) with batch normalization. For PCG: tolerance and maximum number of iterations were not swept (1e-6 and 30, respectively, are in general very good for many tasks).

For ECHO-diam, we swept (i) for the Q-network: ChebNet/ARMA, hidden dimension $\in [4, 8, 16]$, number of layers $\in [3, 5, 10]$, chebychev order $\in [3, 10]$ for ChebNet, number of inner layers and number of stacks $\in [3, 5]$ for ARMA, the initialization value for $q :\in [0.001, 0.01]$; (ii) for the initial shape of the polynomial: close to $\lambda/2$ or close to flat; (iii) for the rest: hidden dimension $\in [64, 128, 256, 512]$, patience $\in [50, 150]$, learning rate $\in$ [5e-3, 1e-2]. For the global pooling: mean (without normalization), mean+max+sum with layer normalization. PCG tolerance and maximum number of iterations were swept: PCG tolerance $\in$ [1e-6, 1e-10, 1e-14], maximum number of iterations $\in [20, 30, 40, 50]$ (best results were obtained for the smallest tolerance and the largest number of iterations: the task really needs to increase the reach of PCG in order to better approximate long-distance interactions).

Overall, we estimate that 3 full weeks of GPU time is the total time of computation of the experiments reported here.

Colors-3 (Fig. 4). Initialization for the coefficients of the polynomial: close to $\lambda/2$. Hidden dimension: 16. Q-network: a ChebNet of 3 layers with degree-3 polynomials in each, hidden dimension of 8. Initialization for q_i: 1.0. Pooling: mean+sum+max (concatenated) with a layer normalization. Batch size = 64. Patience = 50. PCG: tolerance is set to 1e-6, and the maximum number of iterations is 30. Learning rate = 5e-3.

Clique distance (Fig. 5, top). Initialization for the coefficients of the polynomial: close to the constant function equal to 0.5. Input features: the all-one vector. Hidden dimension: 8. Q-network: a

[5] for this, we initialize the last layer of the MLP head to 0 and set its bias to the wanted value

ChebNet of 5 layers with degree-3 polynomials in each, hidden dimension of 8. Initialization for q_i: 0.1. Pooling: mean+sum+max (concatenated) with a layer normalization. Batch size = 128. Patience = 150. PCG: tolerance is set to 1e-6, and the maximum number of iterations is 30. Learning rate = 5e-3.

Triangles (Fig. 5, bottom). Initialization for the coefficients of the polynomial: close to $\lambda/2$. Input features: the all-one vector. Hidden dimension: 8. Q-network: a deep ARMA network of 20 layers with 3 inner layers and 3 stacks each, hidden dimension of 8. Initialization for q_i: 0.01. Pooling: mean+sum+max (concatenated) with a layer normalization. Batch size = 128. Patience = 250. PCG: tolerance is set to $1e-6$, and the maximum number of iterations is 30. Learning rate = 1e-3.

Triangles: second explanation (Fig. 9). We only list what is different from above. Here, the polynomial is fixed to a function very close to $\lambda/2$ (see caption of figure). Hidden dimension: 16. Pooling: sum with no normalization.

CSBM (Fig. 6). Initialization for the coefficients of the polynomial: close to $\lambda/2$. Hidden dimension: 16. Q-network: a ChebNet of 3 layers with degree-10 polynomials in each, hidden dimension of 8. Initialization for q_i: 0.1. Pooling: sum+sumsq (concatenated) with a batch normalization. Batch size = 128. Patience = 50. PCG: tolerance is set to 1e-6, and the maximum number of iterations is 30. Learning rate = 5e-4.

ECHO-diam (Fig. 7). Hidden dimension: 256 (we do as in [23], where they add a linear embedding layer prior to the network to test). Initialization for the coefficients of the polynomial: close to $\lambda/2$. Q-network: a ChebNet of 2 layers with degree-3 polynomials in each, hidden dimension of 8. Initialization for q_i: 0.001. Pooling: mean with no normalization. Batch size = 512. Patience = 150. PCG: tolerance is set to 1e-14, and the maximum number of iterations is set to 50 (number of iterations is always the stopping criterion in practice). Learning rate = 1e-2.

ECHO-diam: second explanation (Fig. 8). We only list what is different from above. Q-network: a ChebNet of 4 layers. Pooling: mean+sum+max (concatenated) with a layer normalization.

C Background on the CSBM

The Contextual Stochastic Block Model (CSBM) combines two well-studied random models: the stochastic block model (SBM) for graph structure, and a Gaussian covariate model for node features. We consider graphs on n nodes (even) split into two blocks of equal size $n/2$, with ground-truth labels $v_i \in \{-1, +1\}$.

Graph structure: SBM. For each pair of nodes (i, j), the edge is drawn independently as a Bernoulli with parameter p_{in} if i, j belong to the same block and p_{out} otherwise. Setting the average degree $\bar{d}$ and the signal parameter $\lambda \in [0, \sqrt{\bar{d}}]$:

$$p_{\text{in}} = \frac{1}{n}\big(\bar{d} + \lambda\sqrt{\bar{d}}\big), \qquad p_{\text{out}} = \frac{1}{n}\big(\bar{d} - \lambda\sqrt{\bar{d}}\big).$$

The block structure becomes detectable when λ is large. The fundamental limit for recovering the communities (up to a constant fraction of misclassified nodes, with high probability as $n \to \infty$) is $\lambda = 1$: weak recovery is possible if and only if $\lambda > 1$ [9, 22].

Node features: covariate model. With feature dimension $p = n/\gamma$, node i has feature vector $\boldsymbol{f}_i \in \mathbb{R}^p$ drawn as

$$\boldsymbol{f}_i = \sqrt{\tfrac{\mu}{n}}\, v_i\, \boldsymbol{u} + \boldsymbol{z}_i,$$

where $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, \mathsf{I}_p/p)$ is a shared random direction, $\boldsymbol{z}_i \sim \mathcal{N}(\boldsymbol{0}, \mathsf{I}_p/p)$ is noise, and $\mu > 0$ is the feature signal strength. Weak recovery from features alone is possible if and only if $\mu/\sqrt{\gamma} > 1$ [12].

Combined CSBM threshold. Jointly using both graph and features, the information-theoretic threshold for weak recovery is [12]:

$$\lambda^2 + \frac{\mu^2}{\gamma} > 1.$$

This curve in the $(\lambda,\ \mu/\sqrt{\gamma})$ plane separates the recoverable from the unrecoverable regime, and is the red quarter-circle shown in Figure 6.

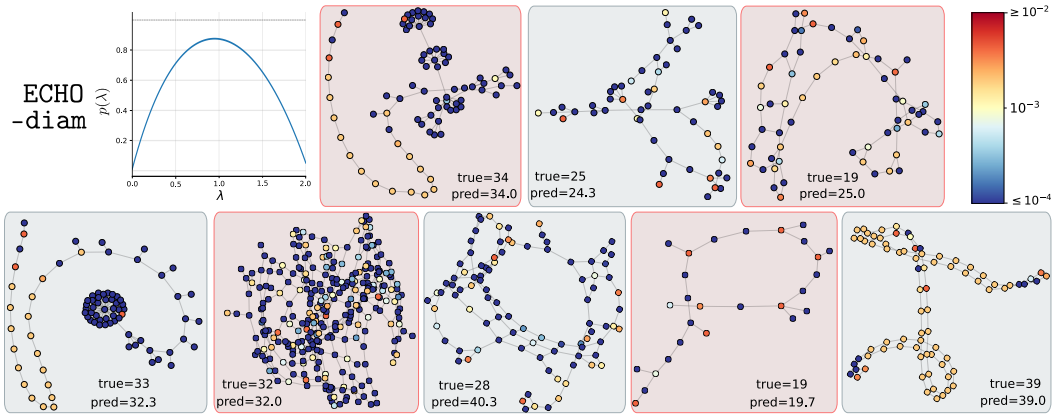


Figure 8: **ECHO-diam benchmark.** Another explanation, obtained with other hyperparameters. The learned polynomial (top-left) along with the learned q-values plotted on 8 example graphs (log scale; blue: $q_i \leq 10^{-4}$, red: $q_i \geq 10^{-2}$). The true diameter is written, along with the prediction of the Tikhonov layer. Best run (in terms of validation loss) out of 5 runs (with different seeds). The median test MAE is 1.12 (the run shown has a test MAE of 1.05). Recall that the MAE here is the average error of prediction of the diameter.

Classification task. The Tikhonov layer is trained to distinguish two distributions of graphs:

- **Class 1 (signal):** graphs drawn from CSBM$(\lambda, \bar{d}, \gamma, \mu)$, with community-aligned features.
- **Class 0 (null):** features drawn with $\mu = 0$ (no community signal); graph drawn from SBM$(\lambda, \bar{d})$ then randomly edge-rewired to preserve the degree distribution (removing any detectable community structure while keeping the degree sequence).

A model succeeding at this task must (i) jointly exploit structure and features, especially in the zone where $\lambda^2 + \frac{\mu^2}{\gamma} > 1$ and $\lambda^2 < 1$ and $\frac{\mu^2}{\gamma} < 1$; (ii) automatically detect when μ is small that the features are useless noise; (ii) automatically detect when λ is small that the topology is useless noise.

D Additional experiments

We plot, in Figures 8 and 9, additional explanations obtained with other choices of hyperparameters (see Appendix B to see the exact changes in hyperparameters), for, respectively, the ECHO-diam and the Triangles datasets. In Figure 8, we see an explanation that is harder to understand than the one shown in the main text (the filter is not only penalizing high frequencies but also low frequencies, in quite a symmetrical way), but boasts a better test MAE and matches the MAE reported in the original paper for the best Transformers architecture.

In Figure 9, we show another explanation that is different in spirit than the one shown in the main text. In this example, the filter was not learned and set to a function very close to $p(\lambda) = \lambda/2$ (see caption), thus mimicking the traditional Tikhonov denoising setting. In the results of the run, we see that the triangle nodes are still different from the others, but this time associated with low values of q compared to large values of the other nodes. The network is not counting the number of nodes in the triangle; it is diffusing the features that are in the triangle. Both explanations are equally valid (and obtain the same almost-perfect test accuracy).

E Runtime experiment

In this appendix, we evaluate the runtime and memory footprint of the Tikhonov layer, against standard GNN baselines on real-world graphs containing up to several thousand nodes.

Experimental setup. We use REDDIT-MULTI-5K, a benchmark of medium-size social-interaction graphs (3,824 training graphs, averaging 508 nodes and 1,189 edges, with maxima of 3,648 nodes and 9,566 edges). Experiments run on a single NVIDIA RTX 6000 Ada GPU (PyTorch 2.4.1,

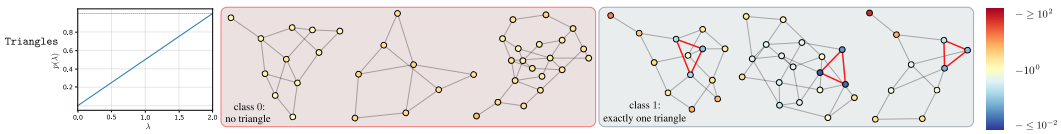


Figure 9: **Triangles benchmark.** Another explanation, obtained with other hyperparameters. The polynomial is here fixed (not learned) to $p(\lambda) = \epsilon + (\frac{1}{2} - \epsilon)\lambda$ with $\epsilon =$ 1e-3 (left). One obtains the learned q-values plotted on 6 example graphs (log scale; blue: $q_i \leq 10^{-2}$, red: $q_i \geq 10^2$): 3 from each class. Best run (in terms of validation loss) out of 5 runs (with different seeds). The median test accuracy is 95% (the run shown has test accuracy of 99.5%).

PyG 2.7.0, CUDA 12.4), using deterministic training settings. After discarding 5 warm-up epochs, we report mean $\pm$ standard deviation over 20 epochs, measuring both complete training epochs (forward, backward, optimizer step) and forward-only inference. Peak GPU memory is measured with `torch.cuda.max_memory_allocated`, including autograd buffers.

Fair comparison. All models share the same architecture apart from the graph convolution operator: identical input features, hidden width (16), readout (mean + sum + max), layer-normalized MLP head, optimizer (Adam, learning rate 10^{-3}), random seed, and data splits. Parameter counts are also comparable (3,300 for Tikhonov versus approximately 2,900 for the baselines). Importantly, one Tikhonov layer involves more computation than a single message-passing step: in our test setting, it combines a 3-layer ChebNet Q-network ($K = 3$), and a degree-5 Bernstein filter. We therefore compare it against 3-layer GIN, GCN, and ChebNet ($K = 3$) baselines rather than single-layer models.

Results. At $T = 30$ (with `cg_tol = 0`, ensuring that T iterations are indeed reached), where T is the maximum number of PCG iterations in both the forward and adjoint backward solves, the Tikhonov layer requires 5.1 ± 0.05 s/training epoch and 2.7 ± 0.03 s/inference epoch, with 198 MB of peak memory. The corresponding training times and memory footprints of the 3-layer baselines are 0.9 s / 198 MB for GCN, 0.9 s / 155 MB for GIN, and 1.3 s / 195 MB for ChebNet. Our layer is thus approximately 4–6$\times$ slower, with a comparable memory footprint (1.01–1.27$\times$ that of the baselines).

To assess the effect of T, we choose $T \in \{5, 15, 30\}$, keeping `cg_tol = 0`. Training time increases from 2.5 to 3.4 to 5.1 s/epoch, approximately linearly in T, while peak memory remains constant at 198 MB. This is enabled by implicit differentiation: the backward pass solves an adjoint system rather than storing the forward solver iterations; thus avoiding memory increase with T.

Discussion. ChebNet provides the closest comparison, since it uses the same convolution operator as our Q-network. Relative to this baseline, the Tikhonov layer is approximately $3.9\times$ slower at $T = 30$, but only $1.9\times$ slower at $T = 5$. The additional cost comes from the global approximate inversion, whose computational budget can be adjusted through T. Our implementation thus offers a tunable trade-off between solver accuracy and runtime, while retaining a memory footprint close to standard GNNs.

F Illustration on two real-world datasets

We train the graph Tikhonov layer on two real-world graph datasets: ZINC [13], a graph-regression task evaluated by MAE, and OGBG-MolHIV [16], a graph-classification task evaluated by ROC-AUC. We use the standard splits and evaluation protocols and compare the Tikhonov model against a matched backbone (a GNN with either Cheb, ARMA, or GINE layers) trained with the same molecular encoders, prediction head, optimization protocol, and random seeds. We observe that performance metrics are similar for the Tikhonov layer than with traditional GNN architectures; while providing potential insights on the inference for each graph. For ZINC, the best standard performance is obtained for GINE with 8 layers, 64 hidden dimensions, mean+sum+max global pooling with batch normalization. To be fair, we compare this with a single Tikhonov layer with 64 hidden dimensions, with a Qnet which is itself a GINE network with 8 layers and 64 dimensions; in a mono-channel with a polynomial of degree 5. We obtain for both methods test MAEs (0.154) that are within standard errors of one another. For OGBG-MolHIV, the best baselines are ChebNet and GINE

with 3 layers, 64 hidden dimensions, mean global pooling with batch normalization. We compare this with a single Tikhonov layer with 64 hidden dimensions, with a Qnet which is itself a ChebNet (or a GINE) network with 3 layers and 32 dimensions; in a mono-channel with a polynomial of degree 5. We obtain for both methods test ROC-AUCs (0.770) that are also within standard errors of one another.

On ZINC, we plotted qvalues (not shown) obtained on each node; on a few molecules of the dataset. Some atoms are highlighted with large values of q, hinting that the existence of such atoms is important for the task; whereas some atoms are associated to lower values of q hinting that the structure there is important (such as cycles). A collaboration with chemists would be necessary to delve deeper into these interpretations.

These results show that the Tikhonov layer is not less expressive than standard baselines on real-world datasets, while providing interpretability to the results.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction claim (i) a self-explainable GNN layer with a variational interpretation, (ii) theoretical properties (identifiability, faithfulness, oversmoothing/oversquashing mitigation), and (iii) experimental validation on synthetic benchmarks. All three are delivered in Sections 3–4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: there are 3 limitations to this work, discussed in the concluding remarks. The first two are not particular to this work but rather to the whole field: (i) some explanations found by the network are sometimes hard to understand for humans, (ii) some tasks have many different possible (correlated) explanations. The last limitation that is discussed is the computation time of PCG. We derive the computation time and compare it with the time of standard baselines in the practical imploementation part.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: All propositions state their assumptions explicitly. Full proofs appear in Appendix A, including an auxiliary Lemma used in the proof of Proposition 3.2. All results are numbered and cross-referenced.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The layer architecture is fully defined (Bernstein polynomial parametrization, PCG solver, implicit differentiation scheme, q_i bounding). Hyperparameter choices are described in Appendix B. All datasets used are publicly available benchmarks, except for the dataset CSBM that we make available online (along with the code and tutorial notebooks) in the provided url.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: code and the CSBM benchmark are released in the provided url. All other datasets used are existing public benchmarks (Colors-3, Triangles, Clique distance, ECHO-diam).

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: how hyperparameters were chosen (and which ones were chosen for the figures) is described in the Appendix B, as well as the choice of optimizer and learning rates. Data splits are the ones from the literature for all datasets except the CSBM. For the CSBM, everything is either described in the paragraph in the experimental section of the main text or in appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Each figure caption reports both the average metric over 5 seeds and the metric for the displayed run. For the CSBM experiment, accuracy is shown as a 2D heatmap (averaged over 6 seeds) over the parameter space, which provides statistical context across regimes.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We extensively discuss the theoretical computation time of our method, in the main text. And provide experimental wall-clock time validations in Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The paper presents a methodological contribution (an interpretable GNN layer) evaluated on synthetic benchmarks. No human subjects, sensitive data, or harmful applications are involved.

   Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper is a foundational contribution to interpretable GNN methods. It has no direct path to negative societal applications; the positive impact (improving transparency of graph-based predictions) is quite straightforward and noted in the introduction and related works sections.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper introduces a GNN layer architecture trained on small synthetic benchmarks. There is no risk of misuse associated with releasing this model or data.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets (Colors-3, Triangles, Clique distance, ECHO-diam) are cited by their original papers. The CSBM data is generated synthetically following the parametrization of Deshpande et al. [12].

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Code implementing the Tikhonov layer, as well as the new CSBM graph classification benchmark, are relleased in provided url. They are fully documented.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper involves no crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper involves no human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were not used as part of the core methodology. They were used only for writing assistance, which does not require declaration under NeurIPS policy.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.