THE ROLE OF EDGE GEOMETRY IN GRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

While much work on graph neural networks has focused on designing node features, some structural information, such as curvature, is naturally defined on edges. Such information can help in two ways: by supplying structural information that standard message passing does not recover, or by improving message propagation. We separate the two and ask when each helps, and for which architectures and tasks. We introduce *curvature profiles* that combine degree-based curvature and counts of cycles of different lengths. These profiles can be used as edge features in standard graph networks or to guide message propagation in our curvature-aware RICCIMP layer. Exact constructions show how profiles can improve filtering and how neighboring context reveals curvature information absent from one edge's profile. At the same time, longer cycles yield a strict expressivity hierarchy: for each fixed $k \geq 3$ curvature-profile refinement is incomparable with the coordinate-wise k-Weisfeiler–Leman test once the cycle order reaches an explicit k-dependent threshold, each distinguishing graph pairs the other misses. Because profiles are computed before any message passing, this reach is obtained by changing the labels refinement reads, not the refinement procedure. Empirically, we show that whether curvature is most useful as a feature or for propagation depends on the task. In particular, curvature profiles substantially improve molecular prediction with GINE, achieving scores close to matched subgraph-counting models, and benefit baseline classification benchmarks and community-recovery tasks. RICCIMP also improves image-region labeling over GINE with the same structural features and unweighted multi-hop propagation. On long-range tasks, augmentation effects are mixed, and RICCIMP trails the strongest tested baselines without curvature profiles. Together, these results distinguish when structural augmentation helps prediction and what the tested propagation choices add.

1 INTRODUCTION

Graph neural networks (GNNs) learn from graph-structured data in molecular property prediction and social and biological networks (Gilmer et al., 2017; Kipf & Welling, 2017; Hamilton et al., 2017). Many build vertex representations by repeatedly exchanging and aggregating neighbors' messages (Veličković et al., 2018; Xu et al., 2019). Standard message passing has limits on the structural patterns it can distinguish (Xu et al., 2019; Morris et al., 2019; Bouritsas et al., 2023), while over-smoothing and over-squashing restrict information flow (Li et al., 2018; Alon & Yahav, 2021; Topping et al., 2022). These limits motivate enriching structural inputs (Bouritsas et al., 2023; Fesser & Weber, 2024b) and changing propagation routes (Topping et al., 2022; Fesser & Weber, 2024a). Which approach helps depends on the architecture and task. A molecular predictor may need to identify rings containing a bond, whereas community prediction may require combining evidence from many vertices without mixing communities. The predictor therefore depends both on the information supplied and on how the architecture propagates it.

Edge curvature offers a way to study both choices because it summarizes how the neighborhoods on either side of an edge are connected. Existing methods supply curvature as a feature, use it to weight messages, or add and remove edges to change the routes along which information flows (Ye et al., 2020; Topping et al., 2022; Fesser & Weber, 2024b). To separate curvature's contribution as structural information from its role in propagation, we supply the same structural object to mechanisms whose inputs and propagation choices we can vary independently.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We make this comparison using a *curvature profile*, a vector for each edge containing unweighted Forman–Ricci curvature $F_{ij} = 4 - d_i - d_j$, where d_i, d_j are the endpoint degrees (Forman, 2003), and the number of cycles of each selected length that contain that edge. By recording triangles and longer rings in separate coordinates, the profile lets us control which structural patterns we supply through the maximum cycle length. We add this same vector to the edge-aware graph isomorphism network GINE (Hu et al., 2020) and gated graph convolutions (GatedGCN) (Bresson & Laurent, 2017), yielding curvature-profile versions of standard backbones. We also introduce RICCIMP, a layer in which the profile guides positive edge weights and a learned combination of repeated propagation steps controls how far messages travel. We can then vary the profile, edge weighting, and propagation range separately.

Main contributions. Our contributions are as follows:

- **Two ways to use edge curvature.** We combine degree-based curvature with edge cycle counts for standard graph networks and introduce a RICCIMP layer with separate controls for edge weights and propagation range, so that feature and propagation effects can be measured separately. Exact constructions show how curvature profiles can improve signal filtering at a fixed propagation budget, and how neighboring context can reveal curvature information absent from a single edge's profile.
- **Which graph structures the models can distinguish.** For fixed edge features, we characterize which graph pairs RicciMP can distinguish and state the architectural conditions to realize these distinctions on finite graph collections. Including longer cycles strictly increases expressivity. For the coordinate-wise k-Weisfeiler–Leman (WL) test (Weisfeiler & Leman, 1968) in any fixed dimension $k \geq 3$, we give an explicit cycle-length threshold beyond which the family and the test each distinguish graph pairs that the other misses. The family obtains these distinctions through precomputed labels and unmodified refinement.
- **When structural input and propagation help prediction.** Matched comparisons and ablations evaluate our curvature-profile augmentations and the RICCIMP layer separately. Curvature-profile inputs improve prediction on molecular graphs with ring-sensitive targets (ZINC) and graphs with planted communities (synthetic community-recovery benchmarks). On tasks involving dependencies beyond immediate neighbors (PascalVOC-SP, Peptides-func), longer walks improve prediction and offer a cheaper alternative to deeper RicciMP networks under the tested settings. Learned edge weights help recover community structure and signals shared within communities (selected planted-community benchmarks, rook-graph denoising), although weights fixed across layers capture much of the gain and topology changes can reverse the denoising advantage.

1.1 RELATED WORK

Curvature and structural features. Local curvature profiles summarize incident edge curvature as node features (Fesser & Weber, 2024b). CurvGN (Ye et al., 2020) uses edge curvature to weight messages, and curvature-based rewiring changes communication routes (Topping et al., 2022; Nguyen et al., 2023; Fesser & Weber, 2024a). Edge-based graph substructure networks (GSN-e) supply substructure counts to messages, and the GSN study compares structural features with and without message passing (Bouritsas et al., 2023). Our cycle coordinates follow this edge-count approach, and we study how they interact with propagation.

Message passing beyond immediate neighbors. APPNP (Gasteiger et al., 2019), GPR-GNN (Chien et al., 2021), and BernNet (He et al., 2021) use fixed or learned propagation polynomials. GPS combines local message passing with global attention (Rampášek et al., 2022). We combine supplied curvature profiles, positive edge weights, and learned combinations of propagation steps.

Distinguishing non-isomorphic graphs. The Weisfeiler–Leman perspective characterizes message-passing expressivity (Xu et al., 2019; Morris et al., 2019). Barceló et al. (2021) supply rooted homomorphism counts, whose cycle hierarchy remains bounded by higher-dimensional WL. Loopy WL instead aggregates colored paths during refinement and establishes incomparability with fixed-dimensional WL at sufficiently large path cutoffs (Paolino et al., 2024). For coordinate-wise d-WL with $d \geq 3$, we establish incomparability for static simple-cycle counts on edges followed by message passing. Graph-transformer analyses likewise isolate supplied structural encodings (Zhang et al., 2023; Zhu et al., 2023; Stoll et al., 2025). More details can be found in Appendix C.

2 WARM-UP: CURVATURE PROFILES AND PROPAGATION

A curvature profile and a propagation rule can capture different aspects of geometry. Two exact finite constructions make the distinction concrete: neighboring context can reveal curvature missing from one edge's profile, and a profile can select a better filtering walk.

From cycles to an edge curvature profile. Let $G = (V, E)$ be a finite simple undirected graph, $N(i)$ the neighbors of vertex i, and $d_i = |N(i)|$ its degree. Let $C_r(ij)$ count simple cycles of length r containing edge ij. A simple cycle visits no vertex twice before closing. For maximum cycle order $R \geq 2$, define

$$\mathfrak{F}_R(ij) = \big(F_{ij}, C_3(ij), \ldots, C_R(ij)\big), \qquad F_{ij} = 4 - d_i - d_j. \tag{1}$$

The first coordinate is unweighted Forman curvature (Forman, 2003), and the others count alternative routes by length. At $R = 2$ the cycle list is empty. Keeping lengths separate preserves motif scales, but does not record how the routes overlap.

From a curvature profile to a walk. A symmetric positive edge weight, or *conductance*, $c_{ij} = c_{ji} > 0$ defines a neighbor average of a scalar signal $x \in \mathbb{R}^V$:

$$(P_c x)_i = \frac{\sum_{j \in N(i)} c_{ij} x_j}{\sum_{j \in N(i)} c_{ij}}. \tag{2}$$

For the connected graphs below, the row $P_c(i, \cdot)$ is a probability distribution, and P_c^q describes q walk steps for a positive integer q. The idleness-zero Ollivier–Ricci curvature $\kappa_c(i, j)$ is one minus the minimum transport cost between $P_c(i, \cdot)$ and $P_c(j, \cdot)$, divided by the endpoints' distance (Ollivier, 2009). Costs use unweighted graph shortest paths; idleness zero means the walk always moves to a neighbor. Positive curvature means that the distributions are closer than the endpoints. Conductances change the walk, while the unweighted shortest-path distance stays fixed. We write κ_1 for the curvature of the unit-conductance walk. Appendix D defines the transport problem and gives exact certificates.

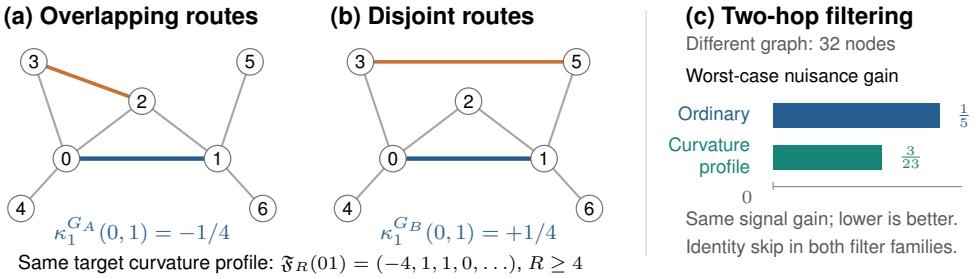

Figure 1: Two complementary roles of curvature profiles and propagation. (a,b) The blue edges 01 have identical curvature profiles at every cycle order, but different unit-walk Ollivier curvatures because their routes overlap differently. (c) On a separate 32-node graph, curvature profile-conditioned propagation lowers the optimal nuisance gain at the same two-hop budget. Both filter families allow an identity term. These are exact constructions.

Proposition 1 (Context missing from an edge curvature profile)**.** *For the graphs G_A, G_B in Figure 1, the target curvature profiles agree for every $R \geq 4$: $\mathfrak{F}_R(01) = (-4, 1, 1, 0, \ldots)$. With unit conductances, $\kappa_1^{G_A}(0, 1) = -1/4$ and $\kappa_1^{G_B}(0, 1) = 1/4$.*

No function of the isolated profile recovers curvature on both graphs, but their neighboring profiles differ. A shared layer can therefore expose the missing context through curvature profile-labeled message passing (Appendix D.2).

Curvature profiles can also improve propagation. To see how a profile can improve a walk, take two 4×4 rook's graphs and match corresponding vertices across the two copies. A rook's graph connects distinct grid cells sharing a row or column. Each internal edge lies in two triangles and each matching edge in none. All vertices have degree seven. A triangle-based conductance doubles the internal weights while leaving the matching weights at one. Let P_1 and P_2 be the unit and reweighted

walks, respectively, and I the identity. We want to retain the difference between community means while suppressing fluctuations within each community. We measure nuisance gain by the largest ratio of output to input Euclidean norm over fluctuations that sum to zero within each community. To compare reweighting at a fixed two-hop budget, consider the following families of signed linear filters:

$$\mathcal{T}_{\mathrm{ord}} = \mathrm{span}\{I, P_1, P_1^2\}, \qquad \mathcal{T}_{\mathrm{prof}} = \mathrm{span}\{I, P_1, P_2P_1\}.$$

Proposition 2 (Curvature profile-assisted filtering; informal)**.** *On this 32-node graph, among filters in these families that preserve the centered community signal, curvature profile reweighting reduces the smallest worst-case within-community nuisance gain from* $1/5$ *to* $3/23$*. Internal-edge curvature increases from* $2/7$ *to* $4/13$*, whereas matching-edge curvature stays zero.*

Appendix D.3 formulates the operator families and minimax problem. The full residual layer realizes these filters locally through its Jacobian (Appendix D.5). The comparison requires centering and a fixed budget because the optimizers amplify constants and higher-degree ordinary filters remove all nuisance in this example. Whether gradient-based optimization finds these weights remains open. Motif-based clustering and diffusion curvature already link local structure to propagation (Benson et al., 2016; Gosztolai & Arnaudon, 2021). Here the exact filtering mechanism motivates a layer with separate structural and propagation controls.

3 RICCIMP

The warm-up motivates using the curvature profile both as a feature and to choose the weights of a multistep walk. Curvature-profile GINE and curvature-profile GatedGCN augment edge features, while RICCIMP uses a positive conductance state and a signed polynomial filter (Figure 2). We give its computation here and detail the expanded maps, all variants, and numerical conventions in Appendix A, which also contains motivations for specific design choices of the layer.

Structural input. For a network with L layers and hidden width d, let $h_i^\ell \in \mathbb{R}^d$ be the state of node i at layer ℓ. The input $\phi_{ij} \in \mathbb{R}^{R-1}$ applies the signed logarithm $\mathrm{slog}\, x = \mathrm{sign}(x)\log(1+|x|)$ to F_{ij} and $\log(1+x)$ to each cycle count in Eq. (1). We cache exact counts up to $R = 6$, with node runs allowed to lower the order at a counting cutoff (Appendix G.1.1) and graph collections enforcing a fixed exact order. The cross-graph theory requires a fixed R and shared injective transform, which excludes node comparisons that use per-graph standardization or different realized orders. We can also append symmetric task edge attributes. ZINC, Peptides, and PascalVOC-SP add eight encoded native edge coordinates to the five structural coordinates at $R = 6$. We continue to write ϕ_{ij} for this augmented vector (Appendix G.1.2).

Positive weights, signed messages. Each edge carries Q positive conductance channels $c_{ij}^\ell \in \mathbb{R}^Q_{>0}$. We take d to be a multiple of Q, share conductances between both edge orientations, and repeat each conductance over its group of d/Q features. A shared affine map initializes $\log c^0$ from ϕ. A learned symmetric edge score $s_{ij}^\ell \in \mathbb{R}^Q$ combines $h_i^\ell + h_j^\ell$, $|h_i^\ell - h_j^\ell|$, and ϕ_{ij}. With learned signed step $\epsilon_\ell \in \mathbb{R}^Q$ and clipping bound $B = 4$, the optional update is

$$\log c_{ij}^{\ell+1} = \mathrm{clip}\big(\log c_{ij}^\ell + \epsilon_\ell \odot \tanh s_{ij}^\ell, -B, B\big).$$

The symbol $\odot$ denotes componentwise multiplication. Let $v_{j\to i}^\ell \in \mathbb{R}^d$ be a message value computed from the sender state and $g_{j\to i}^\ell \in (-1,1)^d$ a signed gate computed from both endpoints. A scalar weight n_{ij}^ℓ downweights cycle-rich edges, using learned scalars $\theta_{\ell r}$ and $\mathrm{softplus}(x) = \log(1+e^x)$. The first aggregate and this weight are

$$a_i^{\ell,1} = \sum_{j\in N(i)} c_{ij}^{\ell+1} \odot n_{ij}^\ell g_{j\to i}^\ell \odot v_{j\to i}^\ell, \qquad n_{ij}^\ell = \left[1 + \frac{C_3(ij)}{2} + \sum_{r=4}^{R} \mathrm{softplus}(\theta_{\ell r}) \frac{C_r(ij)}{r-1}\right]^{-1}.$$

We set $n_{ij}^\ell = 1$ at $R = 2$. An ablation additionally supplies ϕ directly to the message value and gate, rather than using it only for edge weights and the residual step.

Propagation order. Choose a positive integer K as the propagation order and use the walk P_c of Eq. (2) in each conductance channel. Learned signed coefficients $\gamma_{\ell q} \in \mathbb{R}$ combine K propagation

terms, and a nodewise sigmoid gate $\tau_i^\ell \in (0,1)^d$ scales the residual update. Writing $[u; v]$ for vector concatenation, we obtain

$$\bar{a}_i^\ell = \sum_{q=0}^{K-1} \gamma_{\ell q} (P_{c^{\ell+1}}^q a^{\ell,1})_i, \qquad h_i^{\ell+1} = h_i^\ell + \tau_i^\ell \odot \mathrm{Drop}\big(\mathrm{MLP}_\ell(\mathrm{LN}[h_i^\ell; \bar{a}_i^\ell])\big). \tag{3}$$

LayerNorm (LN) (Ba et al., 2016), dropout (Drop) (Srivastava et al., 2014), and the multilayer perceptron (MLP) act nodewise. τ uses h_i and incident mean, minimum, and maximum edge scores. The first aggregate crosses one edge, so K terms reach at most K hops. At $K = 1$ the implementation fixes $\gamma_{\ell 0} = 1$, which larger K recovers by setting the remaining coefficients to zero. Isolates have zero walk rows and zero incident summaries. Appendix A specifies the numerical degree floor and when the resulting walk rows remain probability distributions. For $R = 3, K = 2$, the derivative of this residual layer at a specified input realizes the warm-up filters under constructed weights (Appendix D.5).

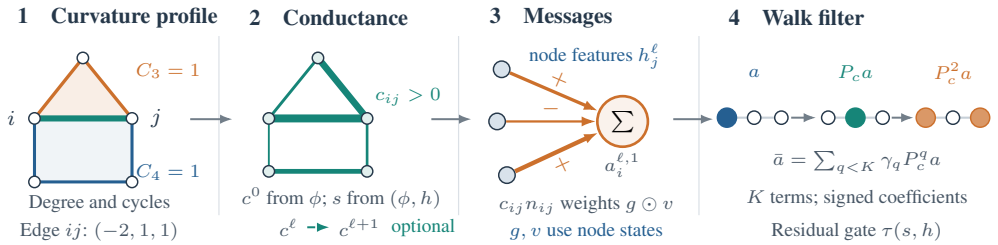

Figure 2: One RICCIMP layer, illustrated for one conductance channel. Cycle curvature profiles initialize positive conductances (edge width), and optional updates use a learned edge score. Node features supply signed messages, weighted by conductance and curvature profile-based redundancy. Repeated conductance-walk steps form a signed polynomial filter before the gated residual update. The rightmost paths illustrate impulse propagation.

Symmetry and propagation range. With symmetric edge labels and deterministic evaluation, relabeling vertices relabels the outputs while conductances remain positive and symmetric. After cycle preprocessing, information travels at most K labeled hops per layer and LK hops across L layers. For degree-normalized variants, the input labels include vertex degrees. Proposition 8 in Appendix E gives the precise conditions and walk properties.

Cost and interventions. With supplied edge inputs of dimension p, a layer whose message values and gates depend only on node states costs $O(|V|d^2 + |E|(dQ + pQ + Kd))$ time and carries $O(|V|d + |E|(Q + p))$ state, excluding activations and preprocessing. We use $Q = 16$ in the implementation. Section 5 tests structural input, conductance, and propagation order separately. Removing the complete structural input also switches to sum aggregation (Appendix A). The learned score need not equal an exact curvature, the conductance update does not discretize Ricci flow, and n_{ij}^ℓ heuristically downweights cycle-rich edges. The geometric terminology motivates these choices, while our theory concerns the supplied profile and propagation range.

4 WHAT THE ARCHITECTURE CAN DISTINGUISH

To distinguish the roles of cycle order R and propagation order K, we first fix the supplied edge labels and then enrich them with longer cycles. Comparison with edge-labeled neighborhood refinement shows which choices change the graph distinctions available to the model family. This focus on supplied encodings also guides graph-transformer analyses (Stoll et al., 2025).

Expressivity with fixed structural input. Color refinement, or 1-WL, repeatedly describes each vertex through its current label and the multiset of its neighbors' labels. Reading the edge labels as well gives the comparison appropriate to RICCIMP. We compare graphs through their histograms of stable colors.

Theorem 3 (RICCIMP and labeled 1-WL; informal)**.** *At deterministic evaluation,* RICCIMP *cannot distinguish graphs that 1-*WL *with the supplied edge labels leaves indistinguishable. On any fixed*

*finite graph collection, a sufficiently wide and deep subfamily with a shared injective edge encoding, sum aggregation, and an injective graph readout can attain all stable labeled-*WL *distinctions.*

Appendix E states the precise assumptions and gives the proof. Although larger K gathers more labeled hops at fixed depth, new stable distinctions within this family require richer labels because propagation cannot split a stable labeled-WL class.

Longer cycles supply new distinctions. Consider two disjoint triangles and one six-cycle, with constant node labels. Ordinary 1-WL gives both graphs the same signature at any depth because every vertex has degree two, but their edge triangle counts differ. We extend this construction to every cycle order. Write $\mathrm{WL}_{\mathfrak{F}_R}$ for stable curvature profile-labeled 1-WL, $\equiv$ for equal separation power, and $\prec$ for strictly weaker power. Let C_m denote an m-cycle graph and $2C_m$ two disjoint copies, distinct from the edge count $C_r(ij)$.

Theorem 4 (Strict curvature profile hierarchy)**.** *On finite unlabeled simple graphs with exact counts, let* $\mathfrak{F}_R(ij) = (4 - d_i - d_j, C_3(ij), \ldots, C_R(ij))$*. Then*

$$\mathrm{WL}_{\mathfrak{F}_2} \equiv \text{1-WL}, \qquad \mathrm{WL}_{\mathfrak{F}_R} \prec \mathrm{WL}_{\mathfrak{F}_{R+1}} \quad \textit{for every } R \geq 2.$$

We may take $2C_{R+1}$ *versus* $C_{2(R+1)}$ *as the strict witness.*

Appendix F proves the hierarchy, and Figure 4 illustrates its first strict separation.

Cycle information and tuple refinement are complementary. We next compare the cycle hierarchy with higher-dimensional WL, which tracks tuples of vertices. Coordinate-wise d-WL updates tuple coordinates separately, whereas the correlated folklore Weisfeiler–Leman test (k-FWL) aggregates coordinate replacements jointly and has the separation power of coordinate-wise $(k+1)$-WL (Feng et al., 2023). The following sufficient cycle-length threshold establishes that neither approach subsumes the other.

Theorem 5 (Incomparability with fixed-dimensional WL)**.** *For each integer* $k \geq 2$*, let* $\rho_k = (k+2)^2/2$ *if* k *is even and* $\rho_k = \binom{k+2}{2}$ *if* k *is odd. For every* $R \geq \rho_k$*, exact* $\mathfrak{F}_R$*-labeled 1-*WL *and coordinate-wise* $(k+1)$-WL *are incomparable on finite unlabeled simple graphs. Each distinguishes a graph pair that the other does not. Equivalently, the incomparability holds with correlated* k-FWL*.*

Comparison with cycle-based refinement. The rooted homomorphism counts of Barceló et al. (2021) allow repeated vertices, and for cycle patterns their hierarchy stays bounded by correlated 2-FWL (coordinate-wise 3-WL). For simple cycles, even global totals distinguish pairs that fixed-dimensional WL misses once the length is large enough (Neuen, 2024). Our edge-rooted counts additionally retain where cycles occur, making that information available for propagation.

Loopy WL also exhibits incomparability with fixed-dimensional WL as its path cutoff grows (Paolino et al., 2024). Our forward direction adapts its Euler-tour argument to labels fixed before refinement, without aggregating paths during refinement. For the reverse direction, tuple refinement detects component structure that edge-labeled message passing can miss. Together with Theorem 3, these comparisons characterize the architecture's distinguishing power under the stated attainment conditions (Appendices C and F). The reverse separation holds at every order, including the implemented $R = 6$, because tuple refinement distinguishes $2C_{R+1}$ from $C_{2(R+1)}$ while order-R profiles cannot. The forward argument requires $R \geq \rho_k$, so its first threshold $\rho_2 = 8$ exceeds $R = 6$. Thus the strict hierarchy of Theorem 4 applies at the implemented orders, whereas this incomparability guarantee does not.

Scope of the theory. Counting precedes the network, and a K-hop layer need not simulate K nonlinear refinement rounds. Finite-domain attainment guarantees neither uniform simulation over unbounded graph sizes, universal approximation, learnability, nor useful predictions. We therefore vary structural input and propagation separately and compare learned conductances with unit and learned-static controls. The warm-up motivates our denoising and topology-transfer experiments.

5 Experiments

To test when the structural information identified by the theory helps prediction, we first supply curvature profiles to established networks, then compare propagation choices within the new layer.

Comparison protocol. The ZINC and Long Range Graph Benchmark (LRGB) comparisons use their standard splits (Dwivedi et al., 2023; 2022b). Each of these comparisons gives models the same validation-search budget and a 500,000-parameter cap. GPS and GatedGCN use adapted reference architectures and training recipes (Rampášek et al., 2022; Tönshoff et al., 2024). The corresponding tables report test means and sample standard deviations over ten seeds (five for Peptides-struct). Appendix H.1 gives selected settings, actual sizes, and paired intervals (Table 12). Node, TU, community, and cycle-order studies use separate protocols (Appendix G).

5.1 CURVATURE PROFILES HELP IN SPECIFIC REGIMES

Table 1: ZINC test mean absolute error (MAE; lower is better), mean ± sample SD over ten seeds. All arms have a common 500k cap and equal validation-search budget. Curvature-profile augmentations and RICCIMP appear below the rule. Bold and underline mark the best and second-best means.

Model	Parameters	MAE ↓
GINE	486,619	0.15671 ± 0.01308
GINE + cycle counts	487,211	0.10183 ± 0.00464
General edge MPNN	467,477	0.13108 ± 0.00742
GSN-e + cycle counts	470,069	$\underline{0.08899} \pm 0.00365$
GINE + curvature profile	487,359	0.09035 ± 0.00340
GSN-e + curvature profile	470,717	$\mathbf{0.08833} \pm 0.00429$
RICCIMP	459,365	0.09490 ± 0.00544

When is additional structure useful? ZINC provides a natural test of cycle information because its molecular score combines lipophilicity, synthetic accessibility, and a penalty for large rings. We compare GINE, a Graph Isomorphism Network with edge features (Xu et al., 2019; Hu et al., 2020), with cycle-only and full curvature-profile inputs. GSN-e, the edge-count version of Graph Substructure Networks, provides a second message architecture consuming the same edge-rooted cycle counts (Bouritsas et al., 2023), allowing us to test whether useful structural input requires our particular propagation rule. The full curvature profile reduces mean absolute error (MAE) by 42.3% relative to plain GINE and by 11.3% relative to cycle-only GINE, improving all ten paired seeds in both comparisons (Table 1). GSN-e is competitive with curvature-profile GINE, and both have better means than RICCIMP. For GSN-e, the paired interval for adding Forman curvature includes zero.

The strict hierarchy (Theorem 4) motivates a fixed-recipe cycle-order sweep. Error drops sharply at six-cycles (Figure 5), as previously observed for GSN (Bouritsas et al., 2023, Section 5.5.1).

Long-range prediction. We also evaluate curvature-profile augmentation on three benchmarks designed to probe long-range prediction. PascalVOC-SP labels image regions represented as graph nodes, Peptides-func predicts peptide functions, and Peptides-struct predicts properties of their three-dimensional shape. The GPS baseline combines message passing and global attention, while GatedGCN learns edge gates for aggregation. Among the tested models, curvature-profile GPS has the best Pascal mean, curvature-profile GatedGCN has the best Peptides-struct mean, and plain GatedGCN has the best Peptides-func mean. The RICCIMP layer trails the strongest baseline without curvature profiles on all three tasks (Table 2). After separate validation selection, curvature-profile GPS exceeds plain GPS's Pascal F1 by 0.54 percentage points, and curvature-profile GatedGCN lowers Peptides-struct MAE by 0.00092 relative to its plain counterpart. Both paired intervals include zero, while the remaining augmentations have worse means. Thus the molecular result does not extend consistently to these strong long-range predictors.

Other graph and node tasks. The same dependence on the backbone appears elsewhere. Curvature-profile GatedGCN has higher mean node-classification accuracy on Roman-empire, while GPR-GNN (Chien et al., 2021) has the best tested means on Cora and CiteSeer. Curvature-profile GINE improves the mean over GINE on all four TU graph-classification datasets in Table 3. The FW–GAT–LCP baseline augments graph attention with local summaries of Ollivier–Ricci curvature (Fesser & Weber, 2024b). Appendices G.2 and G.3 give the settings.

Table 2: LRGB comparison using adapted authors' architectures and training recipes, reporting mean ± sample SD over ten seeds for Pascal and Peptides-func, five for Peptides-struct. F1 and average precision (AP) are percentages. MAE is mean absolute error. Each arm selects its own recipe under a common 500k cap. Appendix H.1 reports actual sizes and selections. Bold and underline mark the best and second-best means. Proposed augmentations and RICCIMP are below the rule.

Model	Pascal F1 ↑	Peptides-func AP ↑	Peptides-struct MAE ↓
GPS	<u>44.48</u>±1.97	66.11±0.84	0.25064±0.00157
GatedGCN	38.58±0.51	**68.35**±0.65	<u>0.24730</u>±0.00204
GPS + curvature profile	**45.02**±0.83	66.04±0.55	0.25255±0.00187
GatedGCN + curvature profile	37.72±0.51	<u>67.45</u>±0.27	**0.24638**±0.00157
RICCIMP $K = 3$	38.33±0.46	66.07±0.78	0.25413±0.00193

Table 3: TU accuracy (percent), mean ± sample SD over ten repeats of stratified 10-fold cross-validation (100 fits per cell). Each repeat averages its ten test folds. Bold and underline mark the best and second-best means.

Method	MUTAG	ENZYMES	PROTEINS	IMDB-BINARY
GatedGCN	74.86±2.04	43.38±2.66	72.35±0.70	50.49±0.80
GPS-Performer	72.51±2.15	37.52±2.10	72.07±0.62	68.54±1.01
GPR-GNN	69.33±2.36	24.87±2.01	70.88±0.66	54.32±1.02
FW–GAT–LCP	82.13±1.43	43.53±3.07	<u>73.50</u>±0.78	65.82±1.60
GINE	77.94±2.12	39.50±2.82	72.03±0.63	70.82±0.68
GINE + curvature profile	**86.30**±1.29	44.63±2.78	**73.75**±1.14	71.60±0.54
GatedGCN + curvature profile	79.30±2.00	47.92±1.85	71.99±0.83	69.33±0.80
RICCIMP $K = 1$	<u>85.60</u>±2.56	**48.68**±1.62	72.64±0.92	<u>71.75</u>±0.84
RICCIMP $K = 3$	84.48±1.98	<u>48.43</u>±2.51	72.42±0.52	**71.90**±0.74

5.2 WHAT THE LAYER CONTRIBUTES

Can longer walks substitute for more layers? The reach bound in Proposition 8 motivates varying propagation order K, the maximum graph steps per layer, at fixed structural input. At fixed width and settings, Pascal F1 rises from 31.67 to 37.74 to 40.26 for $K = 1, 3, 8$. Peptides-func average precision (AP) rises from 63.57 at $K = 1$ to 66.63 at $K = 3$, whereas Peptides-struct changes little (Table 13; Figure 3a). To separate longer walks from nonlinear depth, we compare $K = 3$ with three times as many $K = 1$ layers at equal maximum reach. Pascal F1 is similar (37.74 versus 37.60), but runtime is 25.7 versus 55.4 minutes and model size is 484k versus 1.42M parameters. Under a common 500k cap, $K = 3$ has better means on all three datasets, although actual sizes differ (Appendix H.3). Under these training settings, increasing propagation order offers a cheaper alternative to adding nonlinear layers within RICCIMP.

Comparison with simpler propagation. We add unweighted polynomial propagation to curvature-profile GINE, which also benefits from higher order. At $K = 3$ under the common cap, RICCIMP leads this control on Pascal by 0.90 F1 points, with paired interval $[0.39, 1.50]$, winning eight of ten seeds. It loses on both Peptides tasks and costs more (Tables 15 and 16).

Denoising on the warm-up graph and testing transfer. On the rook graph, triangles identify within-community edges. Proposition 2 proves that curvature reweighting can lower worst-case noise gain within its signal-preserving two-hop filter families. Under constructed parameters, the layer realizes these filters locally through its Jacobian. We test whether training exploits this information to recover a community-constant signal from noise. At $\sigma = 1$, adaptive conductance lowers RICCIMP's mean squared error (MSE) from 1.934×10^{-2} with unit conductance to 5.83×10^{-5}, while static conductance reaches 6.59×10^{-5}. Both fit well, as does curvature-profile GINE, while curvature-profile GatedGCN achieves still lower error. We test transfer by swapping edges within each community while preserving every vertex's degree and evaluating without retraining. After four swaps per community, adaptive RICCIMP has MSE 4.965×10^{-2} versus 1.559×10^{-2} for plain GINE, and curvature-profile GINE rises to 6.726×10^{-2} (Figure 3b; Appendix H.4). After

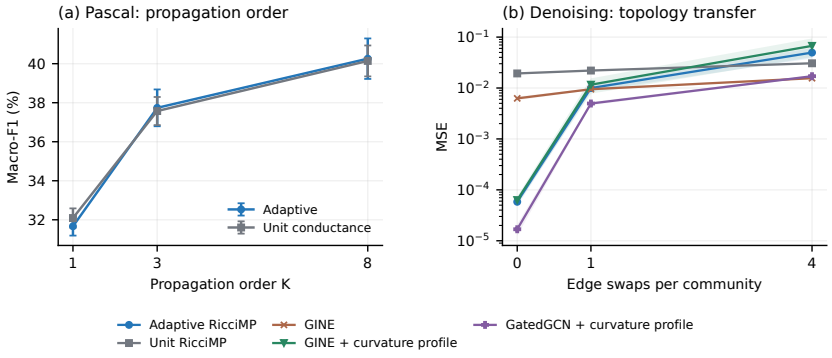

Figure 3: Layer benefits and their boundary. (a) Pascal propagation-order comparison: mean $\pm$ sample SD over ten seeds. Unit conductance retains most of the gain. (b) Rook-graph denoising at noise level $\sigma = 1$: mean and shaded sample SD over ten seeds on the original and changed graphs. Parameter counts differ across backbones.

four swaps, both learned-conductance variants have higher error than unit conductance despite their strong fit on the original graph. In planted-community recovery, too, the curvature profile helps GatedGCN while spectral clustering (Shi & Malik, 2000) and Louvain (Blondel et al., 2008) remain much stronger than the learned layer (Appendix G.4).

A complementary within-layer ablation finds gains from learned conductance in selected planted-community settings and modest mean gains on Roman-empire. (Appendix H.5).

Runtime. On H200, ZINC curvature-profile GINE takes 6.43 minutes per run versus 6.31 for GINE and 15.15 for RICCIMP, corresponding to about 2% overhead for the input augmentation and a full-layer runtime $2.35\times$ that of curvature-profile GINE. These times include startup and evaluation with cached features. Fresh counting is separate (Appendix H.6).

Scope of these comparisons. Long-range augmentation comparisons include independently validation-selected settings alongside input changes, while the propagation-order comparison couples reach, parameterization, and computation. The filtering theorem of Section 2 bounds a filter family, leaving optimization and topology transfer to empirical evaluation.

6 DISCUSSION

We studied curvature profiles as structural features and as inputs to a layer that separately controls edge weights and propagation range. Our theory connects the profiles to filtering, characterizes the layer through labeled refinement, and establishes a strict cycle-order hierarchy and incomparability with coordinate-wise d-WL, $d \geq 3$, above explicit cycle-length thresholds.

Curvature profiles improve GINE on ZINC across every paired seed with little training overhead once counts are cached, and help selected classification and community tasks. Within RICCIMP, longer walks cost less than additional layers in the tested settings, and the full layer improves on polynomial GINE on Pascal. Learned conductance helps denoising and selected community tasks.

Long-range augmentation effects are mixed, with paired intervals for favorable mean changes including zero, while RICCIMP trails the strongest tested plain backbones. Unit weights retain most gains from longer walks, static weights capture much of the conductance benefit, and topology changes reverse the denoising advantage. Future scalar-rescaling and topology-transfer controls could clarify whether relative edge weighting helps beyond aggregate rescaling. Comparisons with higher-order models could also test the complementarity suggested by incomparability.

REPRODUCIBILITY STATEMENT.

Proofs are in Appendices D–F, models in Appendices A–B, and experimental protocols, uncertainty, and incomplete-run coverage in Appendices G–H. Code is supplied as anonymous supplementary material.

AI USE STATEMENT

In this work, we used generative AI to help develop theoretical models and conceptual frameworks, formulate mathematical claims, develop key arguments and constructions used in proofs, write and check proofs, propose and refine hypotheses, design research methodology and experiments, implement methods, develop generators for synthetic graph datasets and noisy signals, and interpret experimental results. We did not use generative AI for translation or dataset cleaning and reformatting. Qualitative and thematic data analysis is not applicable to this work.

We also used these tools for literature search and analysis, brainstorming, suggesting experimental parameters, creating and editing software, preparing figures, tables, and other research artifacts, drafting and editing the manuscript, suggesting its structure, and formatting references.

We have reviewed all AI-assisted work, including the mathematical arguments, code and experimental results, literature attribution, and manuscript text. We take responsibility for the final content of this work, including text, claims, code, and artifacts produced with the assistance of generative AI.

REFERENCES

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

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. URL https://arxiv.org/abs/1607.06450.

Pablo Barceló, Floris Geerts, Juan Reutter, and Maksimilian Ryschkov. Graph neural networks with local graph parameters. In *Advances in Neural Information Processing Systems*, volume 34, pp. 25280–25293, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/d4d8d1ac7e00e9105775a6b660dd3cbb-Abstract.html.

Austin R. Benson, David F. Gleich, and Jure Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016. doi: 10.1126/science.aad9029.

Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008. doi: 10.1088/1742-5468/2008/10/P10008.

Cristian Bodnar, Francesco Di Giovanni, Benjamin P. Chamberlain, Pietro Liò, and Michael M. Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in GNNs. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/75c45fca2aa416ada062b26cc4fb7641-Abstract-Conference.html.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):657–668, 2023. doi: 10.1109/TPAMI.2022.3154319.

Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017. URL https://arxiv.org/abs/1711.07553.

Benjamin P. Chamberlain, James Rowbottom, Davide Eynard, Francesco Di Giovanni, Xiaowen Dong, and Michael M. Bronstein. Beltrami flow and neural diffusion on graphs. In *Advances in Neural Information Processing Systems*, volume 34, 2021a. URL https://proceedings.neurips.cc/paper/2021/hash/0cbed40c0d920b94126eaf5e707be1f5-Abstract.html.

Benjamin Paul Chamberlain, James Rowbottom, Maria I. Gorinova, Stefan Webb, Emanuele Rossi, and Michael M. Bronstein. GRAND: Graph Neural Diffusion. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1407–1418, 2021b. URL https://proceedings.mlr.press/v139/chamberlain21a.html.

Jialong Chen, Bowen Deng, Zhen Wang, Chuan Chen, and Zibin Zheng. Graph neural ricci flow: Evolving feature from a curvature perspective. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/4d3ac0eee841e6df6e08e51932943266-Abstract-Conference.html.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive Universal Generalized PageRank Graph Neural Network. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=n6jl7fLxrP.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2009.14794.

Francesco Di Giovanni, James Rowbottom, Benjamin Paul Chamberlain, Thomas Markovich, and Michael M. Bronstein. Understanding convolution on graphs via energies. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=v5ew3FPTgb.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=wTTjnvGphYj.

Vijay Prakash Dwivedi, Ladislav Rampášek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. In *Advances in Neural Information Processing Systems*, volume 35, pp. 22326–22340, 2022b. doi: 10.52202/068431-1622. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023. URL https://jmlr.org/papers/v24/22-0567.html.

Moshe Eliasof, Eldad Haber, and Eran Treister. Feature transportation improves graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 11874–11882, 2024. doi: 10.1609/aaai.v38i11.29073.

Jiarui Feng, Lecheng Kong, Hao Liu, Dacheng Tao, Fuhai Li, Muhan Zhang, and Yixin Chen. Extending the design space of graph neural networks by rethinking folklore Weisfeiler–Lehman. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1cac8326ce3fbe79171db9754211530c-Abstract-Conference.html.

Lukas Fesser and Melanie Weber. Mitigating over-smoothing and over-squashing using augmentations of Forman–Ricci curvature. In *Proceedings of the Second Learning on Graphs Conference*, volume 231 of *Proceedings of Machine Learning Research*, pp. 19:1–19:28, 2024a. URL https://proceedings.mlr.press/v231/fesser24a.html.

Lukas Fesser and Melanie Weber. Effective structural encodings via local curvature profiles. In *International Conference on Learning Representations*, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/6188c02ccc16a7587716de2efd754033-Abstract-Conference.html.

Robin Forman. Bochner's method for cell complexes and combinatorial Ricci curvature. *Discrete & Computational Geometry*, 29:323–374, 2003. doi: 10.1007/s00454-002-0743-x.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then Propagate: Graph Neural Networks Meet Personalized PageRank. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1gL-2A9Ym.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272, 2017. URL https://proceedings.mlr.press/v70/gilmer17a.html.

Adam Gosztolai and Alexis Arnaudon. Unfolding the multiscale structure of networks with dynamical Ollivier–Ricci curvature. *Nature Communications*, 12:4561, 2021. doi: 10.1038/s41467-021-24884-1.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30, pp. 1025–1035, 2017. URL https://papers.nips.cc/paper_files/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html.

Mingguo He, Zhewei Wei, Zengfeng Huang, and Hongteng Xu. BernNet: Learning arbitrary graph spectral filters via bernstein approximation. In *Advances in Neural Information Processing Systems*, volume 34, pp. 14239–14251, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/76f1cfd7754a6e4fc3281bcccb3d0902-Abstract.html.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016. URL https://arxiv.org/abs/1606.08415.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJlWWJSFDH.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456, 2015. URL https://proceedings.mlr.press/v37/ioffe15.html.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1609.02907.

Haifeng Li, Jun Cao, Jiawei Zhu, Yu Liu, Qing Zhu, and Guohua Wu. Curvature graph neural network. *Information Sciences*, 592:50–66, 2022. doi: 10.1016/j.ins.2021.12.077.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11604. URL https://ojs.aaai.org/index.php/AAAI/article/view/11604.

Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. doi: 10.1109/TIT.1982.1056489.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019. doi: 10.1609/aaai.v33i01.3301 4602.

Daniel Neuen. Homomorphism-distinguishing closedness for graphs of bounded tree-width. In *41st International Symposium on Theoretical Aspects of Computer Science*, volume 289 of *Leibniz International Proceedings in Informatics*, pp. 53:1–53:12, 2024. doi: 10.4230/LIPIcs.STACS.2024.53. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.STACS.2024.53.

Khang Nguyen, Nong Minh Hieu, Vinh Duc Nguyen, Nhat Ho, Stanley Osher, and Tan Minh Nguyen. Revisiting over-smoothing and over-squashing using ollivier-ricci curvature. In *International conference on machine learning*, pp. 25956–25979. PMLR, 2023.

Yann Ollivier. Ricci curvature of Markov chains on metric spaces. *Journal of Functional Analysis*, 256(3): 810–864, 2009. doi: 10.1016/j.jfa.2008.11.001.

Marco Pacini, Gabriele Santin, Bruno Lepri, and Shubhendu Trivedi. On universality classes of equivariant networks. In *Advances in Neural Information Processing Systems*, volume 38, pp. 61689–61719, 2025. doi: 10.52202/085713-2063. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/59013e03266e6dd3e5ab0336f6531e7e-Abstract-Conference.html.

Marco Pacini, Mircea Petrache, Bruno Lepri, Shubhendu Trivedi, and Robin Walters. On universality of deep equivariant networks. In *International Conference on Learning Representations*, pp. 40184–40205, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/4371f23e601415ed6f86310cab06bd4a-Abstract-Conference.html.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, 1999.

Raffaele Paolino, Sohir Maskey, Pascal Welke, and Gitta Kutyniok. Weisfeiler and leman go loopy: A new hierarchy for graph representational learning. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-3838. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/dad28e90cd2c8caedf362d49c4d99e70-Abstract-Conference.html.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tJbbQfw-5wv.

Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *Advances in Neural Information Processing Systems*, volume 35, pp. 14501–14515, 2022. doi: 10.52202/068431-1054. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/5d4834a159f1547b267a05a4e2b7cf5e-Abstract-Conference.html.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. doi: 10.1109/34.868688.

Joshua Southern, Jeremy Wayland, Michael M. Bronstein, and Bastian Rieck. On the expressive power of Ollivier–Ricci curvature on graphs. In *Topology, Algebra, and Geometry in Machine Learning Workshop at ICML*, 2023. URL https://openreview.net/forum?id=F1fuuUYui1.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56): 1929–1958, 2014. URL https://jmlr.org/papers/v15/srivastava14a.html.

Timo Stoll, Luis Müller, and Christopher Morris. Generalizable insights for graph transformers in theory and practice. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/f79df6cbc6e5f708440004fad7ef64cc-Abstract-Conference.html.

Li Sun, Zhenhao Huang, Hua Wu, Junda Ye, Hao Peng, Zhengtao Yu, and Philip S. Yu. DeepRicci: Self-supervised graph structure-feature co-refinement for alleviating over-squashing. In *IEEE International Conference on Data Mining*, pp. 558–567, 2023. doi: 10.1109/ICDM58522.2023.00065. URL https://arxiv.org/abs/2401.12780.

Jan Tönshoff, Martin Ritzert, Eran Rosenbluth, and Martin Grohe. Where did the gap go? reassessing the long-range graph benchmark. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=Nm0WX86sKv.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=7UmjRGzp-A.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

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

Boris Yu. Weisfeiler and Andrei A. Leman. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsiya, Series 2*, (9):12–16, 1968. URL https://www.iti.zcu.cz/wl2018/pdf/wl_paper_translation.pdf. In Russian.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Ze Ye, Kin Sum Liu, Tengfei Ma, Jie Gao, and Chao Chen. Curvature graph network. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BylEqnVFDB.

Bohang Zhang, Shengjie Luo, Liwei Wang, and Di He. Rethinking the expressive power of GNNs via graph biconnectivity. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=r9hNv76KoT3.

Wenhao Zhu, Tianyu Wen, Guojie Song, Liang Wang, and Bo Zheng. On structural expressive power of graph transformers. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3628–3637, 2023. doi: 10.1145/3580305.3599451. URL https://doi.org/10.1145/3580305.3599451.

A Complete Layer Specification

Section 3 gave the layer's computation. We now specify the maps and normalization conventions that determine its variants and computational cost, following the construction in Figure 2.

A.1 Structural input

We store both edge orientations for the graph and curvature profile of Section 2. Let d be the hidden width and L the number of neural layers. Node i has state $h_i^\ell \in \mathbb{R}^d$ at layer ℓ. Write $\operatorname{slog} x = \operatorname{sign}(x)\log(1+|x|)$ for the signed logarithm. The transformed curvature profile $\phi_{ij} \in \mathbb{R}^{R-1}$ compresses the count scale, while redundancy coordinates $\rho_{ij,r}$ retain the raw counts. We compute

$$\phi_{ij} = \big(\operatorname{slog} F_{ij}, \log(1 + C_3(ij)), \ldots, \log(1 + C_R(ij))\big), \tag{4}$$

$$\rho_{ij,r} = \frac{C_r(ij)}{r-1}, \qquad r = 3, \ldots, R, \tag{5}$$

Both orientations receive identical values. In the node experiments, we standardize each coordinate of ϕ across a transductive graph and leave the redundancy coordinates $\rho_{ij,r}$ unstandardized. This graph-dependent preprocessing falls outside the cross-graph theorem, which uses the exact label $\lambda_{ij} = \mathfrak{F}_R(ij)$ or a shared injective transform. The signed logarithm and coordinatewise logarithms in Eq. (4) preserve label equality because they are injective. A fixed affine scaling with nonzero coordinate scales preserves this equality as well, and therefore leaves the labeled-WL partition unchanged.

We compute exact counts once and cache them. The implementation supports $R \le 6$ and handles its counting budget differently for nodes and graph collections: node preprocessing lowers the realized order when the work estimate exceeds the budget, while graph-collection validation rejects a cache that lacks its fixed requested order. The theory allows arbitrary exact R. Appendix G.1.1 reports the realized R_{eff} and validation rule. Unavailable orders are never zero-filled. For the fixed-order theorem, compared models must use the same exact cycle order.

In molecular and superpixel experiments, we concatenate encoded symmetric task edge attributes with the transformed curvature profile and continue to write ϕ_{ij} for the resulting vector (Appendix G.1.2). For PascalVOC-SP, eight learned native-edge coordinates join the five curvature-profile coordinates at $R = 6$. All 13 coordinates enter the initial conductance and symmetric edge score, while the default value and message gate use node states only. The labeled-WL upper bound applies to the combined supplied edge label. The strict cycle-order hierarchy concerns the exact curvature-profile coordinates alone.

A.2 Positive conductance and signed messages

Symmetric edge score and conductance state. The model carries Q positive conductance channels $c_{ij}^\ell \in \mathbb{R}^Q_{>0}$, with hidden width d divisible by Q. Each conductance acts on a group of d/Q hidden coordinates. The initial state and learned symmetric score $s_{ij}^\ell \in \mathbb{R}^Q$ use shared learned maps $A_0, W_h, W_\Delta, W_\phi$ and biases b_0, b_s:

$$\log c_{ij}^0 = A_0\phi_{ij} + b_0, \tag{6}$$

$$s_{ij}^\ell = W_h^\ell h_i^\ell + W_h^\ell h_j^\ell + W_\Delta^\ell |h_i^\ell - h_j^\ell| + W_\phi^\ell \phi_{ij} + b_s^\ell. \tag{7}$$

Let $\epsilon_\ell \in \mathbb{R}^Q$ be a learned signed step vector and let $B > 0$ be a log-conductance bound ($B = 4$ in the experiments). We denote componentwise multiplication by $\odot$ and let $\mathrm{clip}(x, -B, B)$ project each coordinate of x onto $[-B, B]$. The adaptive variant updates conductance as

$$\log c_{ij}^{\ell+1} = \mathrm{clip}\big(\log c_{ij}^{\ell} + \epsilon_\ell \odot \tanh s_{ij}^{\ell}, -B, B\big), \qquad c_{ij}^{\ell+1} = \exp(\log c_{ij}^{\ell+1}). \tag{8}$$

The static variant learns c^0 but keeps it fixed across neural layers. The bounded static control additionally projects $\log c^0$ onto $[-B, B]$; the unrestricted static variant uses Eq. (6) without this projection. The unit-conductance control sets every c_{ij} to one. Equation (7) is symmetric in i, j, so symmetric curvature profiles preserve $c_{ij} = c_{ji} > 0$. We call s_{ij} a learned edge score because training uses curvature and motif inputs without requiring the score to equal an exact curvature. The carried c controls conductance, and Eq. (8) does not discretize Ricci flow. The name "RicciMP" refers to the curvature-derived motivation and input.

Message and redundancy weight. By default, the value and gate for a directed message $j \to i$ depend only on the endpoint node states:

$$v_{j\to i}^{\ell} = W_v^{\ell} h_j^{\ell}, \tag{9}$$

$$g_{j\to i}^{\ell} = \tanh\big(W_i^{\ell} h_i^{\ell} + W_j^{\ell} h_j^{\ell} + b_g^{\ell}\big). \tag{10}$$

The ablation that supplies curvature profiles directly to messages adds separate maps of ϕ_{ij} to the value and gate. A scalar redundancy weight downweights edges with more counted cycles:

$$n_{ij}^{\ell} = \left[1 + \frac{C_3(ij)}{2} + \sum_{r=4}^{R} \mathrm{softplus}(\theta_{\ell r}) \frac{C_r(ij)}{r-1}\right]^{-1}. \tag{11}$$

We learn the scalars $\theta_{\ell r} \in \mathbb{R}$ and use $\mathrm{softplus}(x) = \log(1 + e^x)$ to keep their coefficients positive. For $R = 2$, there are no cycle coordinates and we set $n_{ij}^{\ell} = 1$. This count-based weight is a heuristic and does not equal effective resistance in general. The first aggregate, with conductance channels broadcast to the corresponding feature groups, is

$$a_i^{\ell,1} = \sum_{j \in N(i)} c_{ij}^{\ell+1} \odot n_{ij}^{\ell}\, g_{j\to i}^{\ell} \odot v_{j\to i}^{\ell}. \tag{12}$$

The signed gate can reverse a feature contribution, a possibility that neighbor-softmax attention excludes.

The optional aggregation conventions replace n_{ij}^{ℓ} in Eq. (12) by 1 (sum), $1/\max(d_i, 1)$ (mean), or $1/\sqrt{\max(d_i, 1)\max(d_j, 1)}$ (symmetric-degree normalization). Conductance and message gates remain in place under each convention. The reported RicciMP families use the resistance-inspired choice above by default. At $R = 2$, its redundancy weight is one, giving the sum convention. Changing this first aggregate leaves the conductance-degree normalization of subsequent walk steps unchanged.

A.3 A LEARNED MULTISTEP CONDUCTANCE FILTER

To extend the first aggregate beyond one edge, each channel uses the conductance walk P_c from Eq. (2), with weighted degree $D_{c,i} = \sum_{j \in N(i)} c_{ij}$. For an isolated node, the implementation defines this row of P_c to be zero and fills each empty incident mean, minimum, and maximum below with the zero vector. On every component containing an edge, symmetric positive conductances make the exact P_c a reversible random walk. The implementation divides by $\max(D_{c,i}, 10^{-6})$ for numerical safety. This agrees with P_c on nonisolated vertices when the log-conductances used for propagation lie in $[-4, 4]$, as in the adaptive and bounded static variants. Unrestricted static conductances can activate the floor, in which case the implemented operator need not be stochastic.

Choose a positive integer K for the number of propagation terms. Starting from (12), let $\gamma_{\ell q} \in \mathbb{R}$ be learned filter coefficients for $q = 0, \ldots, K-1$. The filtered aggregate is

$$\bar{a}_i^{\ell} = \sum_{q=0}^{K-1} \gamma_{\ell q} \big(P_{c^{\ell+1}}^{q} a^{\ell,1}\big)_i. \tag{13}$$

The first aggregate already travels one edge, so the largest term reaches K graph hops in one neural layer. We learn the signed coefficients $\gamma_{\ell q}$ for $K > 1$. At $K = 1$, the implementation uses $\bar{a}^{\ell} = a^{\ell,1}$ directly, fixing $\gamma_{\ell 0} = 1$. Setting this coefficient to one and $\gamma_{\ell q} = 0$ for $q > 0$ embeds that case in any larger K.

The filtered aggregate enters a nodewise residual step, where a sigmoid gate $\tau_i^{\ell} \in (0,1)^d$ scales the learned increment. Write σ for the logistic sigmoid, LN for LayerNorm, and Drop for dropout. Square brackets with semicolons denote concatenation. The gate summarizes edge scores over neighbors $j \in N(i)$, taking each statistic coordinatewise:

$$\tau_i^{\ell} = \sigma\left(W_{\tau}^{\ell}[h_i^{\ell}; \operatorname{mean}_j s_{ij}^{\ell}; \min_j s_{ij}^{\ell}; \max_j s_{ij}^{\ell}] + b_{\tau}^{\ell}\right), \tag{14}$$

$$h_i^{\ell+1} = h_i^{\ell} + \tau_i^{\ell} \odot \operatorname{Drop}\big(\operatorname{MLP}_{\ell}(\operatorname{LN}[h_i^{\ell}; \bar{a}_i^{\ell}])\big). \tag{15}$$

The warm-up's operators arise from the first aggregate and one additional walk when $R = 3$ and $K = 2$. Appendix D.5 proves that, under stated fixed-parameter choices, the local Jacobian of this full residual layer realizes both scalar filter families, including the identity term.

The implementation uses $Q = 16$ conductance channels. With supplied curvature profiles and $p = \dim \phi$, one layer with curvature-profile input restricted to conductance initialization and edge scores costs $O(|V|d^2 + |E|(dQ + pQ + Kd))$ time. Exposing the curvature profile directly to d-wide message values and gates adds $O(|E|pd)$. Carried and cached state uses $O(|V|d + |E|(Q + p))$ memory, excluding activations and the one-time cycle preprocessing. These costs include the dense node maps and the $K - 1$ extra sparse propagations.

Ablation controls. Three controls isolate different parts of the residual backbone. (i) *No structure* removes the structural input ϕ and redundancy coordinates ρ and uses sum aggregation; (ii) *unit conductance* sets $c_{ij} \equiv 1$ but retains the edge score, its nodewise step gate, and the redundancy normalizer; and (iii) K changes only the number of polynomial terms. The first control therefore tests the complete supplied structural pathway. The second tests learned conductance while retaining the other uses of curvature-profile information.

To separate the two structural pathways, the factorial control masks curvature-profile input and redundancy normalization independently while retaining parameter dimensions (Appendix H.2.1). This differs from complete structural deletion: the masked coordinates have already been computed, even though the control suppresses their use.

A.4 Neural parameterization and initialization

The maps above use the following affine conventions and initializations. The input encoder is a two-layer MLP with GELU (Hendrycks & Gimpel, 2016). For the displayed linear maps, b_0, b_s, b_g, and b_{τ} in Eqs. (6), (7), (10), and (15) collect the corresponding trainable biases. The difference, curvature profile-to-score, sender-value, and sender-gate maps are bias-free. The curvature profile-to-gate map is also bias-free, whereas the optional curvature profile-to-value map is affine. The update block is $\operatorname{Linear}(2d, 2d)$–GELU–$\operatorname{Linear}(2d, d)$ after LayerNorm of $[h; \bar{a}]$, with a residual connection and dropout. The output of the final RicciMP layer receives one additional LayerNorm before any task head. The node head is $\operatorname{Linear}(d, d)$–GELU–dropout– $\operatorname{Linear}(d, C_{\text{out}})$, where C_{out} is the number of task outputs. We initialize the step-gate weights to zero and its bias b_{τ} to 2, so each gate starts at $\sigma(2) \approx 0.88$. The conductance initializer starts at $\log c^0 = 0$, so all initial conductances equal one, and the update rates start at 0.05. We initialize the higher-cycle normalizer parameters at $\theta_{\ell r} = -4$, giving $\operatorname{softplus}(-4) \approx 0.01815$.

For $K > 1$, we initialize $\gamma_{\ell 0} = 1$ and $\gamma_{\ell q} = 0.5$ for $q > 0$. At $K = 1$ we fix the coefficient to one.

B Baseline Definitions

Ordinary and structural baselines. GCN, GAT, and GraphSAGE follow Kipf & Welling (2017), Veličković et al. (2018), and Hamilton et al. (2017), respectively. GINE is the edge-aware GIN operator (Xu et al., 2019; Hu et al., 2020). Curvature-profile GINE linearly embeds the same standardized ϕ used by RicciMP into its edge inputs. Curvature-profile GatedGCN is the residual gated graph convolution of Bresson & Laurent (2017), again with the same curvature profile. The

comparison suite also includes GPS (Rampášek et al., 2022), with positional encodings absent or validation-selected according to the study, and GPR-GNN (Chien et al., 2021) with a task-specific readout (Appendix G.1). Curvature-profile GPS augments the GPS edge input with our curvature profile. We specify the adapted native GPS/GatedGCN stacks and matched GSN-e controls in the H200 study separately in Appendix H.1.

Matched ORC-LCP comparator. For every undirected edge, we solve the exact optimal-transport problem for idleness-zero Ollivier–Ricci curvature. For each node, we append the minimum, maximum, mean, population standard deviation, and median of its incident edge curvatures to the original node features. Our implementation of Fesser–Weber local curvature profiles with GAT (FW–GAT–LCP) uses width 512 and four attention heads on Roman-empire, with the same depth, dropout, and learning-rate grid as the other models in Table 6. The complementary node and TU experiments use their task-specific widths and training settings (Appendices G.2 and G.3). We evaluate this comparator under our shared protocol rather than copying scores from Fesser & Weber (2024b).

C Further Comparisons with Related Work

The comparisons in Section 1.1 depend on what structural information each model receives and how it uses that information. We make these distinctions precise through the curvature and counting conventions, then compare the scope of the expressivity results.

Curvature as an input or operator. Curvature enters graph learning through several approaches, including those of Li et al. (2022); Fesser & Weber (2024a); Southern et al. (2023). Fesser–Weber local curvature profiles summarize incident Ollivier–Ricci curvatures by five node statistics (Fesser & Weber, 2024b). Their curvature uses uniform probability measures on the two endpoint neighborhoods, with each measure assigning zero mass to its own center vertex. Our profiles change both the curvature definition and its placement in the network, retaining degree-based Forman curvature and each cycle length as separate edge coordinates.

Long-range graph propagation. Propagation provides another way to use geometry. Diffusion and graph-PDE networks change how information moves through a graph (Chamberlain et al., 2021b; Di Giovanni et al., 2023; Eliasof et al., 2024). Beltrami flow and neural sheaf diffusion learn diffusion geometries (Chamberlain et al., 2021a; Bodnar et al., 2022). DeepRicci jointly refines graph structure and features (Sun et al., 2023), while Graph Neural Ricci Flow evolves features using a learned curvature (Chen et al., 2025). These approaches motivate our choice to make geometry an input to propagation. Within the resulting layer, ablations separately vary the supplied curvature profile, positive conductance updates, and polynomial order, which specifies the maximum propagation range.

Structural expressivity and transformers. GSN-e supplies edge-rooted substructure counts to each neighbor's message before aggregation into node representations, whereas GSN-v supplies vertex-rooted counts (Bouritsas et al., 2023, Equations (4)–(5)). Our cycle coordinates are an instance of edge-count augmentation. The GSN paper's Proposition 3.4 gives graph pairs that suitable fixed-size substructures distinguish but 2-FWL does not. Theorem 5 treats every fixed k-FWL dimension $k \geq 2$ and proves both directions of incomparability for a specified cycle family, without separating RicciMP from the general GSN-e family.

Transformer analyses also distinguish the supplied structure from the architecture that consumes it. Generalized-distance refinement (GD-WL) and structural encoding enhanced global refinement (SEG-WL) express bounds in terms of the supplied pair encodings (Zhang et al., 2023; Zhu et al., 2023). Stoll et al. (2025) give simulation and upper-bound results for generalized-distance transformers under stated encoding and architectural conditions. We ask the analogous question for supplied edge labels and the residual architecture of Eq. (3).

Counting conventions and cycle hierarchies. Barceló et al. (2021, Section 2) initialize node features with vertex-rooted homomorphism counts. Homomorphisms preserve edges but may identify distinct pattern vertices. For a cycle C_r, their graph-level count is $\hom(C_r, G) = \operatorname{tr}(A_G^r)$, the number of closed walks of length r, where A_G is the adjacency matrix. Their Proposition 3 bounds a finite pattern family by the maximum pattern treewidth, and Proposition 5 gives a strict cycle-length hierarchy. Because cycles have treewidth two, that hierarchy remains bounded by their 2-WL. Their

Appendix B.1 uses correlated tuple replacements, so this is 2-FWL, or coordinate-wise 3-WL, in our notation.

In contrast, $C_r(e)$ counts simple cycles through edge e. Let $\#C_r(G)$ count undirected cycle subgraphs, whether induced or not, and let $\mathrm{inj}(C_r, G)$ count injective homomorphisms. Then

$$\mathrm{inj}(C_r, G) = 2r\,\#C_r(G), \qquad \sum_{e\in E} C_r(e) = r\,\#C_r(G).$$

The distinction between simple cycles and closed walks changes the relevant treewidth criterion. For subgraph counts, Neuen uses the maximum treewidth among homomorphic images of the pattern, rather than just the pattern's own treewidth (Neuen, 2024, Theorem 3). For cycles this maximum grows with cycle length. Through the identities above, edge-labeled refinement can recover global totals that a fixed-dimensional test misses. Those totals already supply the separation, so placing the counts on edges is not essential to it.

Loopy WL **and the scope of incomparability.** Paolino et al. (2024, Definitions 6–7) refine each vertex's color using colored simple paths between its neighbors that avoid the vertex itself. Their Theorem 1 recovers cycle counts through length $r + 2$, where r is the path-length cutoff. Corollary 1 selects this cutoff according to a fixed WL dimension. Our forward separation applies their Appendix F Neuen/Euler-tour argument to static edge counts instead of colored paths. Component structure supplies the reverse separation in Appendix F, which also proves our explicit threshold.

The same witnesses show that global simple-cycle totals through a sufficiently large finite cutoff are already incomparable with each fixed coordinate-wise d-WL, $d \geq 3$, with the cutoff depending on the comparator's dimension. Our result retains the reverse separation after edge localization and labeled refinement. Theorem 3 then connects that refinement to the architecture under its attainment conditions. These results give no ordering between our refinement and Loopy WL.

Graph separation leaves open which functions a network can approximate, the question addressed by equivariant-network universality studies (Pacini et al., 2025; 2026). Our finite-domain attainment theorem concerns separating parameters on a fixed collection, without establishing this approximation property or guaranteeing that training finds those parameters.

D Proofs for the curvature profile–propagation warm-up

We prove the two warm-up propositions by certifying transport costs and solving a finite-dimensional filtering problem. We then connect the filtering calculation to the local derivative of the residual layer.

D.1 Transport convention

To compare the endpoint distributions, let $\delta(u, v)$ be the unweighted shortest-path distance. For probability vectors μ, ν on V, let $\Pi(\mu, \nu)$ contain the nonnegative matrices with row sums μ and column sums ν. The least transport cost and the idleness-zero Ollivier–Ricci curvature (ORC) (Ollivier, 2009), for distinct vertices i, j, are

$$W_{1,\delta}(\mu, \nu) = \min_{\pi\in\Pi(\mu,\nu)} \sum_{u,v} \delta(u, v)\pi_{uv}, \qquad \kappa_c(i, j) = 1 - \frac{W_{1,\delta}(P_c(i, \cdot), P_c(j, \cdot))}{\delta(i, j)}. \tag{16}$$

Positive curvature means that the two distributions are closer than their starting vertices. Keeping δ fixed when changing c lets us compare walks using the same distance to measure their curvature.

D.2 A complete edge curvature profile can miss curvature-relevant context

Proof of Proposition 1. We enumerate the routes containing the target edge, certify the two transport costs, and show how a shared layer can retain the distinguishing endpoint context.

Equal target curvature profiles. Let G_A and G_B have vertex set $\{0, \ldots, 6\}$ and common edge set

$$E_* = \{01, 02, 03, 04, 12, 15, 16\}.$$

Their remaining edges are 23 in G_A and 35 in G_B. The notation ij denotes the unordered edge $\{i, j\}$. Both endpoints of the target edge 01 have degree four. Removing that edge leaves exactly two simple paths between its endpoints in each graph:

	path of length two	path of length three
G_A	$0{-}2{-}1$	$0{-}3{-}2{-}1$
G_B	$0{-}2{-}1$	$0{-}3{-}5{-}1.$

Adding 01 closes a triangle and a square. Every containing simple cycle must arise from one of these paths after removing 01, so the list is exhaustive and the complete target curvature profiles agree:

$$F_{01} = -4, \qquad C_3(01) = C_4(01) = 1, \qquad C_r(01) = 0 \quad (r \geq 5).$$

The alternate paths share an internal vertex in G_A and are internally disjoint in G_B.

Matching transport bounds. With unit conductances, the endpoint transition measures are uniform on $N(0) = \{1, 2, 3, 4\}$ and $N(1) = \{0, 2, 5, 6\}$. Each atom has mass $1/4$. For an upper bound on transport cost, assign these four masses as follows:

	source–target pairs	four distances	cost
G_A	$(1,5), (2,2), (3,0), (4,6)$	$1, 0, 1, 3$	$5/4$
G_B	$(1,6), (2,2), (3,5), (4,0)$	$1, 0, 1, 1$	$3/4.$

To prove that these couplings are optimal, define vertex potentials $\psi_A, \psi_B : \{0, \ldots, 6\} \to \mathbb{R}$ by listing their values in vertex order:

$$\psi_A = (0, -1, 0, 1, 1, -2, -2), \qquad \psi_B = (0, -1, -1, -1, 1, -2, -2).$$

Each potential changes by at most one on every edge of its graph, hence is 1-Lipschitz for δ. Write μ_i for the transition measure at vertex i in the graph under consideration. Their expectation differences are

$$\sum_z \psi_A(z)\big(\mu_0(z) - \mu_1(z)\big) = \frac{5}{4}, \qquad \sum_z \psi_B(z)\big(\mu_0(z) - \mu_1(z)\big) = \frac{3}{4}.$$

For every coupling π of μ_0, μ_1, the Lipschitz property gives $\sum_z \psi(z)(\mu_0(z) - \mu_1(z)) \leq \sum_{x,y} \delta(x, y)\pi(x, y)$. These lower bounds match the displayed couplings, so

$$W_1^{G_A}(\mu_0, \mu_1) = \frac{5}{4}, \quad W_1^{G_B}(\mu_0, \mu_1) = \frac{3}{4}, \qquad \kappa_1^{G_A}(01) = -\frac{1}{4}, \quad \kappa_1^{G_B}(01) = \frac{1}{4}.$$

Here the subscript 1 denotes unit conductance, and the superscript specifies the graph and its shortest-path metric.

A distinguishing aggregate. We next use neighboring profiles to expose the context missing from the target edge. In the displayed layer, choose exact order $R = 4$, unit conductance, and $\theta_4 = \log(e - 1)$ in Eq. (11). Then $\mathrm{softplus}(\theta_4) = 1$, so the redundancy weight is

$$n_{ij} = \left(1 + \frac{C_3(ij)}{2} + \frac{C_4(ij)}{3}\right)^{-1}.$$

With a constant message gate $1/2$ and value 2 in one channel, the scalar first aggregate at vertex i becomes $a_i = \sum_{j \in N(i)} n_{ij}$. The incident triangle and square counts give the weights needed for the endpoint sums, so we can omit the remaining edges from the table:

	01	02	03	04	12	15	16
G_A	$6/11$	$1/2$	$6/11$	1	$6/11$	1	1
G_B	$6/11$	$2/3$	$3/4$	1	$2/3$	$3/4$	$1.$

Summing these weights at each endpoint gives

	a_0	a_1	$a_0 + a_1$
G_A	$57/22$	$34/11$	$125/22$
G_B	$391/132$	$391/132$	$391/66,$

$$\frac{391}{66} - \frac{125}{22} = \frac{8}{33}.$$

Retaining the statistic through the node update. To pass this distinguishing aggregate through LayerNorm and the residual update to a subsequent edge score, choose a width d that is a multiple

of 16 and initialize every hidden state to the first coordinate vector $e_1 \in \mathbb{R}^d$. Let e_3 denote the third coordinate vector. The linear value map $W_v = 2e_3e_1^T$ and constant gate $1/2$ produce the vector aggregate $a_i e_3$. Set $K = 1$, $\gamma_0 = 1$, disable dropout, and fix the step gate to $1/2$. Set both conductance initialization and update to maintain $c = 1$.

The LayerNorm input $x_i = [e_1; a_i e_3] \in \mathbb{R}^{2d}$ has fixed coordinates one and zero. Take the LayerNorm affine scale one and bias zero. If two such inputs have the same normalized value, one is a positive scalar multiple of the other plus a constant vector. Their fixed coordinates force the scalar to be one and the shift to be zero, so their aggregate values agree. LayerNorm is therefore injective on these update cases.

Because there are at most 14 distinct update cases, Lemma 10 lets the implemented MLP interpolate their outputs with its $2d$ hidden units. Prescribing the output $2a_i e_3$ makes the gated residual equal $h_i' = e_1 + a_i e_3$, retaining the aggregate while preserving both anchors.

In the next layer, a symmetric edge score can read the sum of the third coordinates at its endpoints. For example, subtracting the shared bias $383/66$ from that sum gives

$$s_{01}^{G_A} = -\frac{4}{33}, \qquad s_{01}^{G_B} = \frac{4}{33}.$$

These scores give distinct weights after a unit conductance-update step starting from unit conductance:

$$c_{01}^{G_A} = \exp\left(-\tanh\frac{4}{33}\right), \qquad c_{01}^{G_B} = \exp\left(\tanh\frac{4}{33}\right).$$

Neither value activates the implemented log-conductance clamp, and all maps and parameters are shared across the two graphs. The resulting weights show that contextual propagation can distinguish the two target edges despite their identical curvature profiles. The distinguishing information comes from other edges' curvature profiles, and even ordinary degree context already distinguishes the graphs. The construction shows how to retain that context without estimating curvature automatically. □

D.3 The fixed-budget filtering problem

The first construction shows how propagation can expose context missing from one edge's profile. To study how a profile can instead guide propagation, we now compare two walks at a fixed filtering budget. Let H be the 4×4 rook graph, in which two distinct grid vertices are adjacent when they share a row or column. Join two copies, with vertex sets V_+ and V_-, by matching corresponding vertices, so every vertex has degree seven. Each internal edge lies in two triangles and each matching edge in none, allowing the triangle coordinate to distinguish edge types that F_{ij} does not. For $r > 0$, let P_r denote the walk with internal conductance r and matching conductance one. At $R = 3$, the curvature profile rule $c_{ij} = (1 + C_3(ij))^{\log 2/\log 3}$ gives conductances two and one, respectively, and hence realizes P_2.

Let $s \in \mathbb{R}^{32}$ equal $+1$ on V_+ and -1 on V_-, and let $U = \{z \in \mathbb{R}^{32} : \sum_{i\in V_+} z_i = \sum_{i\in V_-} z_i = 0\}$ be the space of within-community fluctuations. A centered input satisfies $\sum_i x_i = 0$ and uniquely decomposes as $x = \eta s + z$, with $\eta \in \mathbb{R}$ and $z \in U$. A linear filter $T : \mathbb{R}^{32} \to \mathbb{R}^{32}$ preserves the community signal if $Ts = s$. Its nuisance gain is $\|T|_U\|_{2\to 2} = \sup_{z\in U\setminus\{0\}} \|Tz\|_2/\|z\|_2$. Compare the three-coefficient, two-hop operator families

$$\mathcal{T}_{\mathrm{ord}} = \mathrm{span}\{I, P_1, P_1^2\}, \qquad \mathcal{T}_{\mathrm{prof}} = \mathrm{span}\{I, P_1, P_2P_1\}.$$

Here I is the identity on $\mathbb{R}^{32}$, whereas the other terms use one or two neighbor-averaging steps. Only the second walk differs between the families. **Exact statement of Proposition 2.** For the graph, signal, and operator families above,

$$\min_{T\in\mathcal{T}_{\mathrm{ord}},\, Ts=s} \|T|_U\|_{2\to 2} = \frac{1}{5}, \qquad \min_{T\in\mathcal{T}_{\mathrm{prof}},\, Ts=s} \|T|_U\|_{2\to 2} = \frac{3}{23}. \tag{17}$$

The internal-edge curvature increases from $2/7$ for P_1 to $4/13$ for P_2, while matching-edge curvature remains zero.

D.4 CURVATURE PROFILE-CONDITIONED FILTERING ON THE ROOK CONSTRUCTION

Proof of Proposition 2. We realize the two conductance types in the layer and certify their curvatures before comparing their filtering behavior. A common eigenbasis then reduces the nuisance gain to finitely many responses, yielding a lower bound for every admissible choice of filter coefficients.

Graph, curvature profile, and first aggregate. Let B be the adjacency matrix of the rook graph H defined in Appendix D.3. The two-copy graph is the Cartesian product $H \square K_2$, where K_2 is a single edge. Its two edge types have

	internal edge	matching edge
F_{ij}	-10	-10
$C_3(ij)$	2	$0.$

With exact order $R = 3$, the coefficient $\log 2/\log 3$ on $\log(1 + C_3)$ in Eq. (6), with all other coefficients and the bias zero, gives conductance two internally and one on the matching. Set the subsequent conductance-update step to zero. The log-conductances 0 and $\log 2$ lie strictly inside $[-4, 4]$, and the resulting weighted degrees exceed the numerical denominator floor. An affine initializer also realizes the same values after a fixed affine standardization of this nonconstant triangle coordinate.

For internal-to-matching conductance ratio $r > 0$, write the corresponding walk as

$$P_r = \frac{1}{6r+1}\begin{pmatrix} r\mathsf{B} & I_{16} \\ I_{16} & r\mathsf{B} \end{pmatrix}.$$

At the curvature profile choice $r = 2$, the actual $R = 3$ redundancy weights are $1/2$ internally and one on the matching, making conductance times redundancy one on every edge. A constant positive gate and a shared value rescaling then give the first aggregate exactly $P_1 f$ for a scalar input f, followed by the conductance walk P_2. The ordinary construction obtains the same first aggregate from unit conductance and sum aggregation, using the same gate and value rescaling, and follows it with P_1. With $K = 2$, their pre-update stages span, respectively, $\{P_1, P_2P_1\}$ and $\{P_1, P_1^2\}$.

Exact curvature. For an internal edge, the two transition measures share two common neighbors, each of mass $r/(6r + 1)$. Swapping the differing rook coordinate between its two endpoint values takes the first transition measure to the second, fixes the common neighbors, and moves every remaining atom to an adjacent vertex. Its transport cost equals total variation, which also bounds every transport cost from below because distinct vertices have distance at least one. For a matching edge, swapping the K_2 coordinate transports the two measures at cost one. Their supports are disjoint, so the same lower bound proves optimality. Hence

$$\kappa_r(\text{internal}) = \frac{2r}{6r+1}, \qquad \kappa_r(\text{matching}) = 0,$$

and the comparison is

	$r = 1$	$r = 2$
κ_r(internal)	$2/7$	$4/13$
κ_r(matching)	0	$0.$

The internal-versus-matching gap increases, while the zero curvature of the matching edges prevents a positive lower curvature bound on all edges.

Signal and noise modes. Recall the community contrast s and within-community fluctuation space U from Appendix D.3. The globally centered input space is the orthogonal sum $\mathrm{span}\{s\} \oplus U$. The adjacency eigenvalues of K_4 are three on constants and minus one on their orthogonal complement. Since $H = K_4 \square K_4$, its adjacency eigenvalues and multiplicities are

eigenvalue of B	6	2	-2
multiplicity	1	6	$9.$

For a B-eigenvector w with eigenvalue ϑ and a sign $\sigma \in \{1, -1\}$, the vector $(w, \sigma w)$ has P_r-eigenvalue $(r\vartheta + \sigma)/(6r + 1)$. The resulting common eigenbasis gives

mode	P_1	P_2	P_2P_1
s	$5/7$	$11/13$	$55/91$
$\vartheta = 2,\ \sigma = 1$	$3/7$	$5/13$	$15/91$
$\vartheta = 2,\ \sigma = -1$	$1/7$	$3/13$	$3/91$
$\vartheta = -2,\ \sigma = 1$	$-1/7$	$-3/13$	$3/91$
$\vartheta = -2,\ \sigma = -1$	$-3/7$	$-5/13$	$15/91.$

The last four rows exhaust U. Both walks are symmetric, so an operator formed from them has Euclidean operator norm on U equal to its largest absolute response on these rows.

Sharp lower bounds with a matched identity term. First take $T = \alpha I + aP_1 + bP_2P_1 \in \mathcal{T}_{\mathrm{prof}}$ with $Ts = s$, where $\alpha, a, b \in \mathbb{R}$ are arbitrary scalar coefficients. Let $\varepsilon = \|T|_U\|_{2\to 2}$ denote its nuisance amplification, and abbreviate the centers of the outer and inner paired responses by

$$A = \alpha + \frac{15b}{91}, \qquad C = \alpha + \frac{3b}{91}.$$

The two opposite-sign pairs in the table imply

$$|A| + \frac{3|a|}{7} \le \varepsilon, \qquad |C| + \frac{|a|}{7} \le \varepsilon.$$

Signal preservation and these two bounds give

$$\begin{aligned} 1 &= \alpha + \frac{5}{7}a + \frac{55}{91}b = \frac{13}{3}A - \frac{10}{3}C + \frac{5}{7}a \\ &\le \frac{23}{3}\varepsilon - \frac{7}{3}|a| + \frac{5}{7}a \le \frac{23}{3}\varepsilon. \end{aligned}$$

The signal-preservation constraint therefore forces every such curvature profile-conditioned operator to have nuisance norm at least $3/23$.

For $T = \alpha I + aP_1 + bP_1^2 \in \mathcal{T}_{\mathrm{ord}}$, instead set $A = \alpha + 9b/49$ and $C = \alpha + b/49$. The same two nuisance constraints hold, while signal preservation gives

$$1 = 3A - 2C + \frac{5}{7}a \le 5\varepsilon - \frac{11}{7}|a| + \frac{5}{7}a \le 5\varepsilon.$$

This gives the ordinary class a lower bound of $1/5$. Both bounds are attained by the operators

$$T^*_{\mathrm{prof}} = \frac{91P_2P_1 - 9I}{46}, \qquad T^*_{\mathrm{ord}} = \frac{49P_1^2 - 5I}{20}.$$

Their responses on the outer and inner noise pairs are respectively $\pm 3/23$ and $\pm 1/5$, and both send s to itself, attaining the minimax values in Eq. (17). Optimizing all three scalar coefficients in each space compares the two families at a fixed propagation budget and signal subspace, without fixing the total number of parameters in a neural network. □

Both the centering assumption and the restriction on polynomial degree matter for this comparison. On the constant mode, the two displayed optimizers have responses

$$T^*_{\mathrm{ord}}\mathbf{1}_{32} = \frac{11}{5}\mathbf{1}_{32}, \qquad T^*_{\mathrm{prof}}\mathbf{1}_{32} = \frac{41}{23}\mathbf{1}_{32}.$$

Both responses exceed one, so the optimizers need not contract arbitrary uncentered inputs. Increasing the ordinary polynomial degree to four also allows it to vanish on all four nuisance eigenvalues while retaining its value on the community eigenvalue, confining the strict comparison to the specified degree-two operator budget.

D.5 Realizing the filters through LayerNorm and the residual MLP

To connect the two optimal filters to the nonlinear layer, we realize their linear responses at a fixed input. Anchor channels make the derivative of LayerNorm explicit, and the nonzero derivative of GELU lets the residual MLP select the required response.

Lemma 6 (Attainable local filtering response)**.** *Use the graph, walks, and operator spaces of Proposition 2, disable dropout, and let the hidden width $d \ge 16$ be a multiple of 16. Fix either $T = \alpha I + aP_1 + bP_2P_1$ or $T = \alpha I + aP_1 + bP_1^2$. There are shared parameters of the corresponding $R = 3$ curvature profile-conditioned or unit-conductance, no-structure layer, respectively, and fixed vectors $u, v \in \mathbb{R}^d$ such that, for every scalar signal $f \in \mathbb{R}^{32}$, the input $h_i(t) = u + tf_iv$ satisfies*

$$h_i^{\mathrm{next}}(0) = u, \qquad \frac{d}{dt}h_i^{\mathrm{next}}(t)\Big|_{t=0} = (Tf)_iv.$$

The same parameters work for every f. Thus the minimizing operators in Proposition 2 are attainable scalar Jacobians of the implemented residual update.

Proof. We route the scalar perturbation through the two graph operators, then choose a one-unit MLP derivative that combines it with the local residual. Pad the orthogonal, zero-sum channel vectors

$$u = (1, -1, 0, 0), \qquad v = (0, 0, 1, -1)$$

with zeros to dimension d. Fix a constant message gate $g_0 \in (0, 1)$, a constant step gate $\tau_0 \in (0, 1)$, and zero conductance update. We attain these constants with zero gate weights and finite biases. Choose the value map

$$W_v = \frac{vv^T}{14g_0}.$$

Since $v^T u = 0$ and $v^T v = 2$, it annihilates the anchor and sends the input perturbation to $tf_i v/(7g_0)$. In the curvature profile construction, conductance times redundancy is one on every edge, and this is also true in the ordinary construction with no structure. Their first aggregate is therefore $t(P_1 f)_i v$.

To carry both constructions through the remaining calculation, let $\nu = 2$ in the curvature profile construction and $\nu = 1$ in the ordinary construction, and define the propagated operator $A_\nu = aP_1 + bP_\nu P_1$. Set the polynomial coefficients to $\gamma_0 = a$, $\gamma_1 = b$. The concatenated input to the update is then

$$x_i(t) = [u; 0] + t[f_i v; (A_\nu f)_i v].$$

All conductance channels can use the same weights, which is compatible with their implemented broadcast over groups of hidden coordinates.

Take LayerNorm's affine scale one and bias zero. Let $\epsilon_{\mathrm{LN}} > 0$ be its fixed variance stabilizer and set $s_0 = (1/d + \epsilon_{\mathrm{LN}})^{1/2}$. The base vector $x_i(0) = [u; 0]$ has mean zero and variance $1/d$ across its $2d$ coordinates. Each displayed perturbation is orthogonal to both the all-ones vector and $[u; 0]$. On this orthogonal subspace, the derivative of LayerNorm is

$$\frac{d}{dt}\,\mathrm{LN}(x_i(t))\bigg|_{t=0} = \frac{1}{s_0}[f_i v; (A_\nu f)_i v].$$

One hidden unit of the implemented affine–GELU–affine update MLP suffices to combine the two parts of this derivative. For an input $y = [y^{(1)}; y^{(2)}] \in \mathbb{R}^{2d}$, with blocks in $\mathbb{R}^d$, choose its preactivation to be the linear functional

$$L(y) = \frac{\alpha - 1}{2}\langle v, y^{(1)}\rangle + \frac{1}{2}\langle v, y^{(2)}\rangle.$$

Its bias can be zero because $L(\mathrm{LN}([u; 0])) = 0$. Set its output column to $(2s_0/\tau_0)v$, set all other output columns to zero, and set the output bias to zero. The identities $\mathrm{GELU}(0) = 0$ and $\mathrm{GELU}'(0) = 1/2$ give zero base update and derivative

$$\frac{1}{\tau_0}\big((\alpha - 1)f_i + (A_\nu f)_i\big)v.$$

After multiplication by the constant step gate and addition of the residual $f_i v$, the layer has the required derivative $(Tf)_i v$. □

The construction realizes the filters as exact first-order responses at a specified operating point. This local realization leaves open how the nonlinear layer behaves at other input magnitudes and gives no ordering of unrestricted trained networks, additional layers, or task heads.

E Proofs for the Architecture and Labeled-WL Bound

Let λ_{ij} be a supplied symmetric edge label that is preserved by graph isomorphisms and unchanged when we add a disjoint component. Exact curvature profiles satisfy these conditions. Starting from node labels x_i, set $z_i^0 = x_i$ and define the color after $t + 1$ rounds by

$$z_i^{t+1} = \mathrm{HASH}\big(z_i^t, \{\{(\lambda_{ij}, z_j^t) : j \in N(i)\}\}\big). \tag{18}$$

Here HASH assigns different colors to different arguments, and we compute colors jointly across the compared graphs so that their stable-color histograms form comparable graph signatures. For symmetric-degree aggregation, we include degree in the initial labels when stating finite-round bounds, without changing stable separation.

Theorem 7 (Formal version of Theorem 3)**.** *Fix $K \geq 1$.*

1. *For every shared deterministic-inference* RICCIMP *parameterization, h_i^L is determined by the color of i after at most LK rounds of λ-labeled 1-*WL*. Hence stably λ-*WL*-equivalent graphs receive identical outputs under any invariant function of the node-state multiset.*
2. *Let $\mathcal{C}$ be any fixed finite collection of finitely labeled graphs, and let $T_{\mathcal{C}}$ be the stabilization round of λ-labeled 1-*WL *on their disjoint union. For every $L \geq T_{\mathcal{C}}$, some finite-width L-layer* RICCIMP *subfamily with sum aggregation, dropout disabled, and an injective finite-domain graph readout attains every stable distinction on $\mathcal{C}$, provided the architecture receives one shared injective encoding of the supplied edge labels. The attaining parameters may depend on $\mathcal{C}$.*

*Thus the architecture family and λ-labeled 1-*WL *have the same stable graph-separation power on a fixed finite domain.*

For the upper bound, we show that labeled neighbor multisets determine each layer's output. Attainment requires encoding these multisets injectively through sum aggregation and LayerNorm, then using an injective readout. The construction establishes existence on a fixed finite domain and leaves mean or symmetric normalization, generic trained parameters, and the implemented non-injective graph readout outside its scope.

Proposition 8 (Equivariance, reach, and positive geometry)**.** *Assume symmetric edge labels, both stored edge orientations, and evaluation with dropout disabled. A* RICCIMP *layer is permutation equivariant, and all conductances are positive and symmetric in exact arithmetic. Every post-update learned-flow conductance $c^{\ell+1}$ lies in $[e^{-B}, e^{B}]$, whereas the unclipped initialization c^0, also used by the unclipped static variant, has no such uniform bound without an additional assumption on $A_0\phi + b_0$. The bounded-static variant projects $\log c^0$ into $[-B, B]$ and keeps it fixed. For the exact weighted-degree operator, on each component containing an edge P_c is reversible with stationary mass proportional to weighted degree. Isolates follow the zero-row convention above. One layer depends on at most K labeled graph hops, and L layers on at most LK hops.*

Permuting dropout masks along with the vertices gives the same equivariance identity for each realization. The reversibility statement requires exact weighted-degree division and applies to the positive-degree blocks of P_c. It does not extend to the signed polynomial in Eq. (13). We can retain directed task attributes provided we keep them outside the conductance inputs.

Proof of Proposition 8. At deterministic evaluation, a vertex permutation acts on node tensors by rows and on directed edge tensors by the induced edge permutation. Shared linear maps, endpoint sums, and absolute differences commute with this action, while scatter sum, mean, minimum, and maximum are invariant to edge order. The score, conductance, message aggregate, polynomial, and node update therefore commute with the vertex permutation.

If $\lambda_{ij} = \lambda_{ji}$ and the previous conductance is symmetric, Eqs. (6)–(8) give the same score and update in both orientations. Exponentiation makes the initialization positive, with no uniform upper bound because Eq. (6) is unclipped. Learned flow then clips each post-update log-conductance in Eq. (8), giving $e^{-B} \leq c_{ij}^{\ell+1} \leq e^{B}$. Static flow retains the positive, symmetric c^0, with the same bound when we apply the static projection. For the exact operator, let C denote the weighted adjacency matrix of one conductance channel and let D_c be its diagonal matrix of weighted degrees. On a component containing an edge, C is symmetric and positive on E, so $P_c = D_c^{-1}C$ obeys detailed balance with $\pi_i = D_{c,i} / \sum_v D_{c,v}$, where the sum is over that component. Isolates follow the stated zero-row and zero-summary conventions. Equation (12) travels one edge, and each of the $K - 1$ subsequent applications of P_c travels at most one more. Induction over layers gives the LK support bound. □

To attain labeled refinement, a sum must retain multiplicities of the joint edge-label and neighbor-color multiset. The next lemma constructs weights that prevent collisions on the finitely many multisets under consideration.

Lemma 9 (Finite exponential-moment encoding)**.** *Let $\Lambda \subset \mathbb{R}^p$ be finite, let $n : \Lambda \to \mathbb{R}_{>0}$ be fixed, and let $\mathcal{M} \subset \mathbb{N}_0^{\Lambda}$ be a finite set of multiplicity vectors, where $\mathbb{N}_0$ denotes the nonnegative integers.*

For almost every $w \in \mathbb{R}^p$, the map

$$m \longmapsto \sum_{\lambda \in \Lambda} m_\lambda n(\lambda) e^{w^\top \lambda}$$

is injective on $\mathcal{M}$. Moreover, an admissible w exists in every prescribed nonempty open ball in $\mathbb{R}^p$.

Proof. For distinct $m, m' \in \mathcal{M}$, their difference defines the real-analytic function $f_{m,m'}(w) = \sum_\lambda (m_\lambda - m'_\lambda) n(\lambda) e^{w^\top \lambda}$. To verify linear independence of the distinct exponentials, choose a direction u for which the numbers $u^\top \lambda$ are distinct and send the scalar t in $w = tu$ to infinity. The term with largest exponent and nonzero coefficient then cannot cancel. Thus $f_{m,m'}$ is not identically zero, and its zero set has Lebesgue measure zero. A finite union over pairs in $\mathcal{M}$ still has measure zero and therefore cannot contain a nonempty open ball. □

The moment encoding recovers the refinement multiset. To implement the resulting finite update table with the stated encoder and residual block, we also need a width bound that does not grow with the ambient input dimension.

Lemma 10 (Exact finite-set GELU interpolation)**.** *Let $x_1, \dots, x_M \in \mathbb{R}^a$ be distinct, with $M \geq 1$, and let $y_1, \dots, y_M \in \mathbb{R}^b$ be arbitrary targets. An affine–GELU–linear network with M hidden units maps each x_i exactly to y_i. The hidden-unit bound is independent of a and b.*

Proof. We first construct an invertible table of hidden-unit values, and the output map then solves a finite linear system. Choose a vector $v \in \mathbb{R}^a$ whose projections $t_i = v^\top x_i$ are distinct, and reorder them increasingly. Choose thresholds $s_1 < t_1$ and $s_j \in (t_{j-1}, t_j)$ for $2 \leq j \leq M$. The matrix $A_{ij} = (t_i - s_j)_+$ is lower triangular with positive diagonal, so it is invertible. Write Φ for the standard normal distribution function, so $\mathrm{GELU}(t) = t\Phi(t)$. For a scale $c > 0$, define

$$A^{(c)}_{ij} = c^{-1} \,\mathrm{GELU}(c(t_i - s_j)) \,.$$

As $c \to \infty$, this matrix converges entrywise to A. Continuity of the determinant gives a finite c for which $A^{(c)}$ is invertible. Solve $A^{(c)} W = Y$, where row i of Y is $y_i^\top$. The hidden affine maps $x \mapsto c(v^\top x - s_j)$ followed by GELU and output weights $c^{-1} W$ then interpolate all targets exactly. □

Proof of Theorem 7. Upper bound. The displayed redundancy rule, sum aggregation, and mean aggregation use supplied labels and incident multiplicities. The optional symmetric-degree normalizer additionally uses neighboring degrees, so for that variant the finite-round and radius bounds start from $z_i^0 = (x_i, d_i)$. Degree is already determined after one ordinary refinement round, so this convention does not change stable graph separation. The attaining subfamily below uses sum aggregation and needs no such augmentation. Run labeled refinement on the disjoint union of the compared graphs. At initialization, the encoder output is a shared function of x_i and hence of z_i^0. Suppose the incoming node state is determined by a labeled-WL color at round t, and the carried edge state by the two endpoint colors and λ_{ij}. Equations (7)–(12) are shared functions of the self color and the multiset of labeled neighbor colors, so the first aggregate is determined after one further refinement round. Each multiplication by P_c is another labeled neighbor aggregation and costs at most one further round. The incident score statistics and residual update are shared functions of the same data. One neural layer is therefore determined by at most K more refinement rounds, giving the LK bound by induction. Since an invariant readout sees the same node-state multiset on graphs with equal stable color histograms, it cannot separate them.

Attainment on a fixed finite collection. Only finitely many edge labels, refinement colors, degrees, and neighbor multiplicity vectors occur through $T_{\mathcal{C}}$. Let $S = \sum_{G \in \mathcal{C}} |V(G)|$ be the total number of vertices. When $S = 0$ there is nothing to prove. Otherwise, each round has at most S colors and at most S distinct node-update cases. Choose $d \geq S + 2$ to be a multiple of Q, and encode color b at each round by $u_b = (0, 1, e_b, 0, \dots) \in \mathbb{R}^d$, reusing the S-coordinate color block across rounds. We keep the first two coordinates fixed as anchors and use Lemma 10 to implement the initial-label encoding with at most $S \leq d$ hidden units, setting unused output weights to zero. The same bound fits every later update into the implemented hidden width $2d$.

Use $K = 1$ behavior inside the chosen K family by setting $\gamma_0 = 1$ and $\gamma_q = 0$ for $q > 0$. Disable dropout, set $\epsilon = 0$, use sum aggregation, and set the message and step gates to constants $g_0 \neq 0$ and $\alpha \in (0, 1)$. Let $\widetilde{\lambda} = \phi(\lambda)$ under a single shared injective transform. Its image $\widetilde{\Lambda}$ is finite. For sum aggregation the redundancy multiplier is $n(\widetilde{\lambda}) \equiv 1$. Apply Lemma 9 to the union of all multiplicity sets needed across rounds and colors with this n. Choose its w in a sufficiently small open ball that $|w^\top \widetilde{\lambda}| < B$ for every $\widetilde{\lambda} \in \widetilde{\Lambda}$. Repeat w across conductance channels and set $b_0 = 0$. This choice keeps the clip inactive, so $c_{ij} = \exp(w^\top \widetilde{\lambda}_{ij})$ throughout.

At each round, we choose the finitely many current color codes to be linearly independent. Allocate one value coordinate for each of them, and choose $W_v u_b = g_0^{-1} e_b$. Node i then receives

$$a_i^{(b)} = \sum_{\widetilde{\lambda} \in \widetilde{\Lambda}} m_i(\widetilde{\lambda}, b) n(\widetilde{\lambda}) e^{w^\top \widetilde{\lambda}},$$

where $m_i(\widetilde{\lambda}, b)$ counts neighbors of color b joined by the corresponding original label. The vector over b therefore determines the joint multiset $\{\{(\lambda_{ij}, z_j)\}\}$.

It remains to account for the displayed normalization and residual. Set the LayerNorm affine scale and shift to one and zero. If $\mathrm{LN}(x) = \mathrm{LN}(y)$, then $y = ax + \beta \mathbf{1}$ for some $a > 0$. For $x = [u_b; a_i]$ and $y = [u_{b'}; a_{i'}]$, the two anchored coordinates force $\beta = 0$ and $a = 1$. Hence LayerNorm is injective on the finite set of update cases. For a case whose incoming and next colors are b and b', use Lemma 10 to prescribe the MLP output $\alpha^{-1}(u_{b'} - u_b)$. The residual update then equals $u_{b'}$ exactly. The first two target coordinates vanish, so we can set their output weights to zero. Induction preserves the anchored, linearly independent color codes. After $T_{\mathcal{C}}$ layers, any extra layers use zero increments. The final LayerNorm is injective on these codes by the same argument. Another finite-domain map sends each resulting color code to a basis vector, and a sum readout records its multiplicity. This separates every pair of distinct stable graph histograms on $\mathcal{C}$. □

Remark 11 (Why the lower bound is restricted). The proof relies on preserving multiplicities, which mean or symmetric normalization can discard. A fixed positive label-only multiplier can be absorbed into $n(\lambda)$ in Lemma 9, whereas arbitrary reweighting may destroy injectivity. Whether gradient training recovers these constructed parameters remains a separate question.

F Proof of the Curvature-Profile Hierarchy and Incomparability

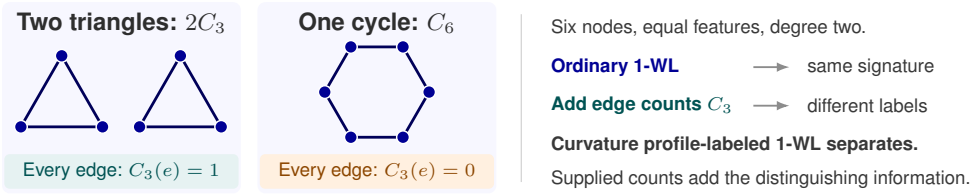

Figure 4: A worked example of the strict curvature profile hierarchy, $R = 2 \to 3$. Both graphs have six identically featured vertices of degree two, so ordinary 1-WL and the degree-Forman curvature profile give the same stable signature. Every edge has triangle count one in $2C_3$ and zero in C_6, so adding C_3 to the supplied curvature profile separates the graphs in one labeled refinement step.

Lemma 12 (Degree Forman labels and stable refinement)**.** *Let β be any fixed symmetric edge labeling of finite simple graphs, let x be their initial vertex labels, and let $F(ij) = 4 - d_i - d_j$. Stable β-labeled 1-*WL *and stable (β, F)-labeled 1-*WL*, both starting from x, have the same graph-separation power. In particular, for every $R \geq 2$, adding F to the edge cycle counts $(C_3, \ldots, C_R)$ leaves stable separation unchanged. At $R = 2$, the cycle tuple is empty and both rules have ordinary 1-*WL *power.*

Proof. Projection onto β shows that augmented refinement is at least as strong. For the converse, compute a common stable β-coloring q on the disjoint union of the compared graphs. Each color q_i determines d_i and the multiset of pairs (β_{ij}, q_j). The endpoint colors therefore determine $F(ij)$, so q_i also determines the multiset of $((\beta_{ij}, F(ij)), q_j)$. Thus q is stable under the augmented rule and

refines the initial labels. Inductively, every augmented color is a function of q. Equal stable β-color histograms consequently give equal augmented histograms, proving the converse. □

Although the stable graph-separation power is unchanged, supplying F can affect finite-round refinement, finite-depth networks, and optimization.

Proof of Theorem 4. Lemma 12 gives the base case. Projection from $\mathfrak{F}_{R+1}$ to $\mathfrak{F}_R$ gives nesting. For strictness set $m = R + 1$, let $G = 2C_m$, and let $H = C_{2m}$. Both have $2m$ vertices, degree two, and no cycles of orders $3, \ldots, R$, so all $\mathfrak{F}_R$ edge labels are identical and both stable graph signatures are uniform. Every edge of G has $C_m = 1$, whereas every edge of H has $C_m = 0$. The first $\mathfrak{F}_{R+1}$-labeled update separates them. □

We call the joint-replacement tuple rule correlated k-FWL and the separate-coordinate rule coordinate-wise k-WL. For $k \geq 2$, correlated k-FWL has the graph-separation power of coordinate-wise $(k+1)$-WL (Feng et al., 2023). Neuen calls the joint-replacement rule k-WL. In our notation, his characterization says that the subgraph count of F is invariant under correlated k-FWL exactly when the maximum treewidth among edge-surjective homomorphic images of F is at most k (Neuen, 2024).

To prove Theorem 5, we use the cycle-order threshold ρ_k from the main text to find a pair that cycle counts separate and correlated k-FWL misses. For the reverse direction, component structure separates a pair missed by the curvature profile. This argument supplies a sufficient threshold and leaves the smallest possible cycle order open.

Proof of Theorem 5. Fix $k \geq 2$ and put $n = k + 2$. If n is odd, an Euler tour of K_n has length $\binom{n}{2}$. If n is even, duplicate a perfect matching before taking the Euler tour, producing length $n^2/2$. In both cases the resulting closed walk gives a vertex- and edge-surjective homomorphism from C_{ρ_k} onto K_n. Hence the hereditary treewidth of C_{ρ_k} is at least $\mathrm{tw}(K_n) = k + 1$.

Since $\mathrm{tw}(K_n) = k+1 > k$, Neuen's characterization supplies graphs G_k, H_k that correlated k-FWL cannot distinguish but that have different numbers of C_{ρ_k} subgraphs. If $R \geq \rho_k$, the first curvature profile-labeled update records the multiset of incident $C_{\rho_k}(e)$ values. Summing over node colors determines

$$\sum_{v \in V} \sum_{e \ni v} C_{\rho_k}(e) = 2\rho_k \, \#C_{\rho_k}(G),$$

so curvature profile-labeled 1-WL distinguishes G_k, H_k.

For the reverse direction, set $m = R + 1$ and again compare $2C_m$ with C_{2m}. Their $\mathfrak{F}_R$ labels and vertex histograms agree. Correlated 2-FWL detects component structure. Let A be the adjacency matrix and I the identity matrix, and set $B = I + A$. After t pair-refinement rounds the pair color determines $(B^{2^t})_{uv}$ by induction through matrix multiplication. For $2^t \geq 2m - 1$, positivity of this entry is equivalent to u, v lying in the same component. The two graphs have respectively $2m^2$ and $4m^2$ ordered within-component pairs. Thus 2-FWL, and hence k-FWL, separates them. Translating k-FWL to coordinate-wise $(k+1)$-WL gives the stated $d = k + 1$ form. □

G Complementary Benchmark Protocols

The node, TU, cycle-order, and planted-community comparisons in Section 5 use the protocols below. We report them separately from the ZINC/LRGB comparison and layer controls in Appendix H, without pooling studies.

G.1 Shared implementation

Preprocessing and training. Triangle and four-cycle counts use sparse matrix algebra, with exact local adjacency routines above the sparse-work threshold, whereas five- and six-cycle counts use simple-path enumeration. We compute counts once per undirected edge, cache them, and share them across both orientations. We use $R = 6$ subject to the work limit below, or $R = 4$ for community recovery. We standardize curvature-profile columns using training edges on inductive tasks and

all edges on transductive graphs, without labels. Training uses AdamW (Loshchilov & Hutter, 2019) and gradient clipping at norm 5. Unless we specify a cosine schedule or the community rule below, a plateau scheduler halves the learning rate after one quarter of the early-stopping patience without validation-loss improvement. Except for community recovery below, we select checkpoints to maximize the validation metric (minimize MAE), break ties by validation loss, and then evaluate on the test set. Runs use PyTorch 2.5.1, CUDA 12.4, and PyTorch Geometric 2.7.0.

Random seeds. Cora and CiteSeer use split and initialization seeds $0, \dots, 4$. TU uses repeat seeds $s = 0, \dots, 9$ and fold indices $f = 0, \dots, 9$; stratified fold allocation uses random state s, and initialization uses $1000s + f$. For planted-community recovery, graph seeds are $0, \dots, 9$ for training, $10000, \dots, 10004$ for validation, and $20000, \dots, 20019$ for testing. The feature seed equals the graph seed, with separate deterministic streams for topology, features, and vertex permutations in the supplied generator. Neural initialization seeds are $0, 1, 2$; classical methods use seed 0. Both width and parameter-budget panels share graph draws and initialization seeds.

Additional dependencies. The recorded H200 training environment used NumPy 2.2.6, SciPy 1.15.3, scikit-learn 1.5.2, TorchMetrics 1.6.1, PyTorch Lightning 2.5.0.post0, `torch_scatter` 2.1.2 (the PyTorch 2.5/CUDA 12.4 build), OGB 1.3.6, YACS 0.1.8, Numba 0.67.0, POT 0.9.7, and pandas 2.3.3.

Supplementary code. The source-only archive supplies preprocessing (`ricci.preprocessing`), node and graph experiments (`ricci.experiments`), reference GPS/GatedGCN and matched GSN-e adapters (`ricci.h200_reference`), planted-community experiments (`community.experiment`), denoising (`ricci.h200_warmup`), and aggregation (`analysis.statistics` and `community.analysis`). The caller supplies the settings documented in these appendices; the archive contains no datasets, fitted checkpoints, recorded results, or experiment presets.

Backbone conventions. The complementary node/TU GPS baseline uses GINE, four-head Performer attention (Choromanski et al., 2021), and no positional encoding, differing from the native LRGB adapter in Appendix H.1. GPR-GNN uses its two-layer input network, propagation dropout 0.5, and $K = 10$ with personalized PageRank (PPR) initialization (Page et al., 1999; Chien et al., 2021). The PPR teleport probability is $\alpha = 0.1$ unless selected below. Its graph readout is a task adaptation, and it does not consume edge attributes. The suffix "-sep" denotes an update that reads the receiver state h and its message aggregate m separately, using a learned linear map W, layer normalization LN, and the GELU activation: $h \leftarrow h + \mathrm{GELU}(W\,\mathrm{LN}[h \,\|\, m])$, in place of $h \leftarrow h + \mathrm{GELU}(\mathrm{LN}\, m)$ (Platonov et al., 2023). At equal width and depth, parameter counts still differ across models. We constrain model size explicitly in the ZINC sweep and community capacity comparison only. Tables report mean and sample SD over the units specified for each study.

G.1.1 Realized curvature-profile orders

We limit cycle enumeration using a deterministic work estimate. For candidate order $r \in \{5, 6\}$ and vertex degrees d_i, preprocessing computes

$$W_r(G) = \sum_{ij \in E} \left[\max\left\{ \frac{d_i + d_j}{2} - 1, 1 \right\} \right]^{r-2}.$$

Preprocessing retains all computed coordinates and stops before the first order whose estimate exceeds the budget. The threshold is 3×10^9 for single-graph preprocessing and 5×10^7 per graph in the graph-collection loader. Triangle and four-cycle counts precede this guard. Table 4 reports the largest retained order R_{eff} for each dataset. For graph collections, training requires exact counts through $R = 6$ for every graph and rejects incomplete counts or inconsistent feature dimensions. TU, ZINC, and LRGB therefore use all coordinates through six, and the order sweep uses their prefixes. Unavailable coordinates are never zero-filled.

G.1.2 Native attributes and readouts

Structural coordinates supplement the attributes supplied with a dataset. Table 5 specifies the encoders used by the shared graph-collection models. GPS and GatedGCN in the main LRGB comparison instead use the encoders in Appendix H.1. Categorical encoding sums one learned embedding per

Table 4: Realized curvature-profile orders. Budgets limit the counting work estimate, not elapsed time. Community recovery fixes $R = 4$.

Study	Dataset	R_{eff}	Counting budget
Heterophily	Roman-empire	6	3×10^9
Citation	Cora	5	3×10^9
	CiteSeer	6	3×10^9
Graph collection	TU, ZINC, Peptides, PascalVOC-SP	6	5×10^7 per graph

input column. The node embedding has the model's hidden width. We concatenate the width-eight native edge embedding with the curvature profile. When we select random-walk structural encodings (RWSE) (Dwivedi et al., 2022a), we standardize them with training statistics, linearly map them to hidden width, and add them to the node embedding.

Table 5: Native input encoders for the shared graph-collection models. Standardization uses training statistics. Removing the curvature profile retains native inputs and any selected positional encoding.

Dataset	Node attributes	Native edge attributes
ZINC-12k	Categorical atom types; embedding sum	Categorical bond types; embedding sum
Peptides	Categorical atom descriptors; embedding sum	Categorical bond descriptors; embedding sum
PascalVOC-SP	Standardized real features; linear map	Standardized real features; linear map
TU	Supplied node features or constant one; linear map	Edge labels removed for every arm

Native and structural edge embeddings enter the maps specified in Appendix A. Curvature-profile GINE projects their concatenation into the GINE edge input, whereas curvature-profile GatedGCN uses it as the initial edge state. The corresponding plain backbones retain native attributes. GINE sums neighbors, while GatedGCN normalizes its gates. Readouts use mean pooling on TU and Peptides, sum pooling on ZINC, and a node head on PascalVOC-SP, with no claim that these trained heads realize the injective readout in Theorem 7.

G.2 Node classification

Roman-empire. Run $s \in \{0, \dots, 9\}$ uses official split s and initialization seed s (Platonov et al., 2023). Models use width 512, no weight decay, at most 1,000 epochs, and patience 100. Each arm selects depth $\{5, 10\}$, dropout $\{0.2, 0.5\}$, and learning rate $\{3 \times 10^{-4}, 10^{-3}, 3 \times 10^{-3}\}$ by mean validation accuracy over the ten splits (12 settings). GPR-GNN selects its teleport weight $\alpha \in \{0.1, 0.5, 0.9\}$ instead of depth (18 settings). Table 6 gives test accuracy, selected settings, and parameter counts. The profile helps GatedGCN but not GINE under separate tuning, which couples the input change with the selected settings.

Cora and CiteSeer. Five seeded, unstratified 50/25/25 splits of the largest connected component replace the public Planetoid masks. All arms use depth 4, width 64, learning rate 3×10^{-3}, at most 300 epochs, and patience 40. We select dropout and weight decay from $\{0.3, 0.5\} \times \{10^{-5}, 5 \times 10^{-4}\}$ by mean validation accuracy. Table 7 reports the selected settings and test scores.

Table 6: Roman-empire accuracy (percent), mean ± sample SD over ten official splits. Settings: validation-selected depth / dropout / learning rate in units of 10^{-3} / parameters. GPR-GNN replaces depth with teleport weight α. Width is 512, but model sizes differ. Bold marks the best mean.

Method	Accuracy ↑	Selected settings
GatedGCN	88.38±0.56	10 / 0.5 / 1 / 13.9M
GPS-Performer	87.60±0.42	10 / 0.2 / 0.3 / 29.6M
GPR-GNN	67.80±0.57	α=0.5 / 0.5 / 3 / 163k
FW–GAT–LCP	84.74±0.25	5 / 0.5 / 3 / 6.0M
GINE	85.41±0.57	10 / 0.5 / 1 / 8.6M
GCN-sep	87.44±0.54	10 / 0.5 / 1 / 8.6M
SAGE-sep	86.95±0.53	10 / 0.5 / 1 / 11.2M
GINE-sep	88.61±0.64	10 / 0.5 / 0.3 / 13.8M
GINE + curvature profile	85.16±0.57	10 / 0.5 / 0.3 / 8.6M
GatedGCN + curvature profile	**90.03**±0.55	10 / 0.5 / 1 / 13.9M
RICCIMP $K = 1$	89.97±0.56	10 / 0.5 / 0.3 / 27.4M
RICCIMP $K = 3$	89.44±0.54	10 / 0.5 / 0.3 / 27.4M
GINE-sep + curvature profile	88.72±0.40	10 / 0.5 / 0.3 / 13.8M

Table 7: Cora and CiteSeer accuracy (percent), mean ± sample SD over five seeded 50/25/25 splits of the largest connected component. Bold marks the best mean, and superscripts indicate validation-selected dropout and weight decay.

Method	Cora ↑	CiteSeer ↑
GatedGCN	$84.95\pm1.62^{\mathrm{d}}$	$75.36\pm1.69^{\mathrm{b}}$
GPS-Performer	$81.41\pm1.96^{\mathrm{b}}$	$74.98\pm1.74^{\mathrm{b}}$
GPR-GNN	$\mathbf{87.81}\pm0.93^{\mathrm{b}}$	$\mathbf{78.00}\pm0.81^{\mathrm{d}}$
FW–GAT–LCP	$86.62\pm1.80^{\mathrm{b}}$	$76.11\pm0.88^{\mathrm{b}}$
GINE	$85.37\pm2.66^{\mathrm{a}}$	$76.45\pm1.46^{\mathrm{d}}$
GINE + curvature profile	$85.14\pm1.46^{\mathrm{b}}$	$76.91\pm1.00^{\mathrm{b}}$
GatedGCN + curvature profile	$84.66\pm1.70^{\mathrm{d}}$	$76.87\pm0.49^{\mathrm{b}}$
RICCIMP $K = 1$	$82.70\pm2.60^{\mathrm{b}}$	$74.00\pm1.71^{\mathrm{d}}$
RICCIMP $K = 3$	$85.02\pm1.90^{\mathrm{d}}$	$76.79\pm1.30^{\mathrm{b}}$

Settings (dropout, weight decay): $a = (0.3, 10^{-5})$, $b = (0.3, 5\times10^{-4})$, $c = (0.5, 10^{-5})$, $d = (0.5, 5\times10^{-4})$. Depth is 4, width 64, and learning rate 3×10^{-3}.

G.3 TU CLASSIFICATION AND ZINC CYCLE ORDER

TU. MUTAG, ENZYMES, PROTEINS, and IMDB-BINARY use width 64, depth 4, dropout 0.15, batch size 32, learning rate 2×10^{-3}, weight decay 10^{-4}, at most 180 epochs, patience 30, and mean pooling. We repeat stratified 10-fold cross-validation ten times. Within a repeat, fold f is the test set, fold $f + 1$ (cyclically) supplies validation checkpoints, and the other eight folds train the model. Each repeat averages its ten folds, and Table 3 reports means and SDs over repeats, conditional on these datasets. With our implementation and inputs, plain GatedGCN has no graph-discriminating signal on IMDB-BINARY by construction because constant node inputs, absent edge attributes, normalized aggregation, and mean pooling make all graph representations identical. Nonconstant edge features can break this symmetry, so this particular improvement measures recovery from a degenerate baseline.

ZINC cycle-order sweep. We evaluate the official ZINC-12k split (Dwivedi et al., 2023) with seeds 0–3, a 500k parameter cap, six layers, sum pooling, batch size 128, no dropout or weight decay, and learning rate 10^{-3}. Training lasts 500 epochs with cosine scheduling and ten warm-up epochs. This sweep uses a fixed recipe, whereas the main comparison selects learning rates. $R = 2$ supplies only Forman curvature, whereas larger R adds exact cycle counts through that order. The no-curvature-profile control also removes redundancy normalization while retaining native attributes. Figure 5 shows the improvement on adding six-cycles across all four model families.

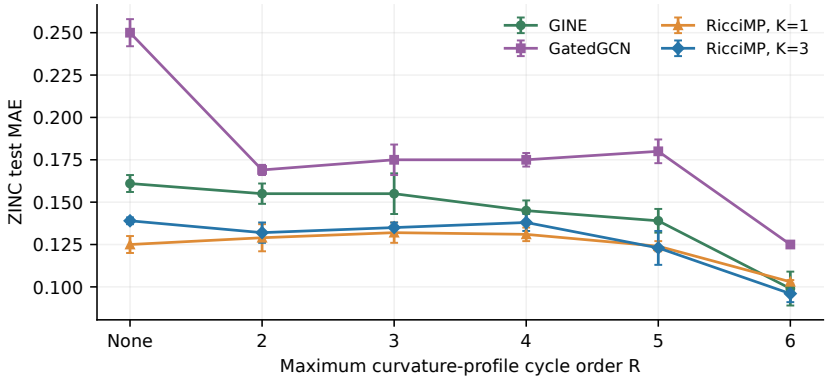

Figure 5: ZINC cycle-order response at a fixed training recipe and a common 500k parameter cap. Means and sample SDs over four seeds. Lower MAE is better. "None" removes the curvature profile and redundancy normalization, whereas $R = 2$ retains Forman curvature only. The largest improvement from increasing R occurs when six-cycle counts enter the input.

G.4 PLANTED-COMMUNITY RECOVERY

We examine the assortative regime discussed in the main text. Each graph has four communities of 60 vertices, with eight neighbors inside and two outside each vertex's community. The feature signal-to-noise ratio (SNR) is 0.3, defined by the construction below. We independently draw ten training, five validation, and 20 test graphs and permute node identifiers. The curvature profile uses $R = 4$.

Graph and feature generation. Within each block we draw a random 8-regular graph. A random 2-regular graph on the four block identifiers specifies which block pairs receive uniformly random perfect matchings, giving every vertex two cross-block neighbors.

Let M_{in} and M_{out} be the numbers of undirected within- and between-block edges. We perform $10(M_{\text{in}} + M_{\text{out}})$ attempted two-edge switches to randomize the graph while preserving its block degrees. Each attempt selects a block pair with probability proportional to its edge count, chooses two edges without replacement, and proposes $ab, cd \mapsto ad, cb$. We randomly orient within-block edges and accept a proposal only with four distinct endpoints and no new edge already present, preserving each vertex's degree into every block.

For each graph, we draw four independent Gaussian vectors in $\mathbb{R}^{16}$ and normalize them to unit vectors $\mu_1, \ldots, \mu_4$. Writing z_i for vertex i's planted label, we set $x_i = s\mu_{z_i} + \xi_i/4$, with independent $\xi_i \sim N(0, I_{16})$ and $s = 0.3$. Since $\|s\mu_{z_i}\|_2 = s$ and $\mathbb{E}\|\xi_i/4\|_2^2 = 1$, SNR here denotes the signal amplitude relative to the root-mean-square noise norm. We redraw centroids per graph, generating topology, features, and vertex permutations with independent random streams, separate from the three optimization seeds.

Unlabeled training objective. For one graph with $N = 240$ vertices, let A be its symmetric binary adjacency matrix, let $d = A\mathbf{1}$ be its degree vector, and let $v = \mathbf{1}^\top d$ be twice its number of edges. The network's four logits per vertex give a row-stochastic assignment matrix $Z \in \mathbb{R}^{N\times 4}$ by softmax. We minimize

$$\mathcal{L}(Z; A) = -\frac{\operatorname{tr}(Z^\top AZ) - \|Z^\top d\|_2^2/v}{v} + \left(\frac{2}{N}\|Z^\top \mathbf{1}\|_2 - 1\right).$$

With balance-penalty coefficient one, this objective rewards assortative partitions. Both the loss and validation selection operate without planted labels.

Training and evaluation. Neural models train inductively without community labels. For each initialization, we select the checkpoint and learning rate by the minimum mean validation value of the complete unlabeled objective, including its balance penalty. We test learning rates 10^{-3} and 3×10^{-3} in that order, retaining the earlier candidate on an exact tie; checkpoint ties also retain the earlier checkpoint. The plateau scheduler monitors this objective and halves the learning rate with patience $\max\{1, \lfloor 40/3 \rfloor\} = 13$. Models use four layers where applicable, dropout 0.2, weight decay 10^{-5}, at most 200 epochs, patience 40, and three initializations. One comparison fixes width 64, while another chooses widths near 200k parameters. The capacity search considers widths $16, 20, \ldots, 16384$, restricted to multiples of 16 for RICCIMP, and chooses the parameter count nearest 200k, breaking ties toward smaller width. Table 8 gives realized sizes and adjusted Rand index (ARI). We average initializations within each test graph, then report mean and sample SD over 20 graphs. These SDs condition on the training graphs. Both parameter-budget comparisons also share graph draws and initialization seeds, making them paired comparisons rather than independent data replications. Spectral clustering, Louvain (resolution one), and feature k-means (Lloyd, 1982) fit each test graph directly. All except Louvain receive the number of clusters. The near-perfect recovery of spectral clustering and Louvain shows that the neural predictors leave much of the available signal unused. The assortative modularity objective limits the scope of this study to assortative recovery.

Table 8: Planted-community recovery in the neutral assortative $8/2$ degree setting at feature SNR 0.3. ARI ↑ is mean (sample SD) over 20 test graphs after averaging three neural initializations. Classical methods fit each test graph, whereas neural models are inductive. Bold marks the best mean.

Method	Width-64 ARI	Parameters	Near-200k ARI	Parameters
Spectral	**1.000** (0.000)	—	**1.000** (0.000)	—
Louvain	0.999 (0.002)	—	0.999 (0.002)	—
Feature k-means	0.113 (0.058)	—	0.113 (0.058)	—
GINE	0.148 (0.051)	60,232	0.088 (0.035)	193,612
GatedGCN	0.186 (0.080)	94,532	0.242 (0.074)	192,560
GINE + curvature profile	0.018 (0.009)	60,488	0.014 (0.007)	194,076
GatedGCN + curvature profile	0.392 (0.064)	94,788	0.498 (0.118)	192,928
RICCIMP $K = 1$	0.280 (0.125)	196,808	0.229 (0.128)	196,808
RICCIMP $K = 3$	0.032 (0.017)	196,820	0.026 (0.016)	196,820

Near 200k parameters, GINE and its curvature-profile variant use width 116, GatedGCN and its curvature-profile variant use width 92, and both RICCIMP orders retain width 64. This target does not match compute. Classical methods have no neural parameter count.

H ZINC AND LONG-RANGE PROTOCOLS AND CONTROLS

The ZINC and LRGB comparisons in Section 5 use the following settings and controls.

H.1 Protocol and comparator definitions

Selection and training. Final fits use seeds 1–10, or 1–5 on Peptides-struct, and seed 0 is the only selection seed. On LRGB, we select jointly among depths $\{6, 8, 10\}$, no positional encoding or random-walk structural encodings (RWSE), and prediction-head depths $\{2, 3\}$, giving 12 settings per model. On ZINC, we select the learning rate from $\{0.0005, 0.001, 0.002\}$. We evaluate the test set after choosing the best validation checkpoint. Tables 10 and 11 give the chosen settings, which may differ between plain and augmented backbones. Table 9 gives the fixed recipes for the mechanism controls. Fits use AdamW, zero weight decay, cosine scheduling, and FP32 arithmetic on one H200 (PyTorch 2.5.1, CUDA 12.4, PyG 2.7.0). The 500k cap includes all encoders and heads, but actual counts differ. The cosine-scheduled runs, including the ZINC cycle-order sweep, use the full epoch budget (early-stopping patience equals the epoch budget), with validation checkpoint selection. The adapted GPS/GatedGCN models use widths in multiples of four. The RICCIMP depth controls use multiples of 16.

Structural inputs and adapters. The ZINC and LRGB comparisons use official splits and the attributes supplied with each dataset. The full curvature profile contains signed-log Forman curvature and $\log(1 + C_r)$ for exact C_3–C_6 counts, standardized using training edges only. Cycle-only inputs omit Forman curvature. The LRGB GPS implementation combines a local gated graph convolution, Transformer attention (Vaswani et al., 2017), batch normalization (Ioffe & Szegedy, 2015), and the encoders and training components from the reference implementation. We similarly adapt the reference GatedGCN architecture. Curvature-profile variants add a bias-free projection to the native edge embedding. GPS uses four attention heads and attention dropout 0.5. RWSE contains 20 return probabilities, which these models encode in 28 dimensions. Gradient clipping uses norm 1 for GPS and Peptides GatedGCN, no clipping for Pascal GatedGCN, and norm 5 for the shared-implementation models and GSN-e. The shared input representation, search grid, width selection, and validation-only checkpoint choice make these comparisons adaptations of the reference protocol rather than reproductions of published scores.

GSN-e uses the local-edge general-message equation (Bouritsas et al., 2023). A message MLP consumes both endpoints, edge-rooted counts, and native edge attributes, and an update MLP combines the receiver state and aggregated messages. Counts enter every edge message. Each cycle has one edge orbit, so matching exact C_3–C_6 counts retains all cycle edge-orbit information at those orders. Continuous count transforms, native inputs, optimization, and search budget match the ZINC comparison. We supply no extra degree tag. The "General edge MPNN" row in Table 1 removes cycle identifiers while retaining this message and update architecture. This differs from GINE's additive edge-embedding message and from published GSN experiments using categorical counts or longer cycles.

Uncertainty. Values are means and sample SDs over final seeds. F1 and AP are reported as percentages, while MAE and MSE retain their native units. We compute descriptive paired intervals from 20,000 percentile-bootstrap resamples of seeds, conditional on the data and selected settings. Unless we specify a paired Student-t interval or Holm adjustment below, the intervals have no multiplicity adjustment. Hyperparameter selection is held fixed, so these intervals cover neither its variability nor variation across new datasets. Appendix G specifies the distinct uncertainty units used by the complementary benchmarks.

Table 9: Fixed recipes used by the conductance, component, depth, and polynomial controls. ZINC models share the listed fixed settings. All use AdamW, zero weight decay, cosine scheduling, and a validation-selected checkpoint. LRGB dropout is 0.1, whereas ZINC dropout is zero. The learning rate (LR) is 10^{-3} on LRGB and validation-selected on ZINC. L is the number of layers, PE denotes positional encoding, Head is the prediction-head depth, and Warm-up is the number of warm-up epochs.

Dataset	L	PE	Head	Epochs	Batch	Warm-up
PascalVOC-SP	10	—	2	200	50	10
Peptides-func	6	RWSE	2	250	128	5
Peptides-struct	8	—	3	250	128	5
ZINC	6	—	2	500	128	10

Table 10: Selected LRGB settings. L: layers; d: width; H: head depth; PE: positional encoding. All learning rates are 10^{-3}. Parameters include encoders and heads. Minutes include startup and evaluation with cached inputs.

Model	L	d	PE	H	Parameters	Min.
PascalVOC-SP						
GPS	10	60	RWSE	2	487,429	38.44
GPS + curvature profile	10	60	—	2	487,521	38.10
GatedGCN	10	96	—	2	482,517	21.16
GatedGCN + curvature profile	10	96	—	2	482,997	21.53
RICCIMP ($K = 3$)	10	64	RWSE	2	485,545	25.90
Peptides-func						
GPS	6	76	RWSE	2	477,150	18.67
GPS + curvature profile	6	76	RWSE	2	477,530	18.92
GatedGCN	6	120	RWSE	3	486,926	7.75
GatedGCN + curvature profile	6	120	—	3	491,770	7.39
RICCIMP ($K = 3$)	8	64	—	2	465,498	12.76
Peptides-struct						
GPS	8	64	RWSE	3	454,007	22.04
GPS + curvature profile	10	60	RWSE	2	492,807	26.47
GatedGCN	6	120	RWSE	3	487,047	7.74
GatedGCN + curvature profile	6	120	RWSE	3	487,647	7.93
RICCIMP ($K = 3$)	8	64	—	3	469,723	12.62

Table 11: Selected ZINC settings for Table 1. All models use six layers, a two-layer head, sum pooling, and no positional encoding. Minutes include startup and evaluation with cached inputs.

Model	Width	LR	Min.
GINE	148	0.001	6.31
General edge MPNN	108	0.0005	6.97
GINE + cycle counts	148	0.0005	6.41
GINE + curvature profile	148	0.002	6.43
GSN-e + cycle counts	108	0.0005	7.09
GSN-e + curvature profile	108	0.0005	7.06
RICCIMP ($K = 3$)	80	0.002	15.15

Table 12 gives the paired intervals used in the main text, including all six GPS/GatedGCN augmentation comparisons. The ZINC and GPS/GatedGCN comparisons retain the independently selected settings of each arm.

Table 12: Paired contrasts supporting the main comparisons. Positive values favor the first arm. MAE improvements are reductions, and F1 and AP improvements are percentage points. Intervals use 20,000 percentile-bootstrap resamples of paired seeds, conditional on the data and chosen configurations, without multiplicity adjustment. Wins count seeds favoring the first arm. For ZINC, "full" denotes the curvature profile and "cycles" omits Forman curvature. The LRGB augmentation rows compare each named backbone with and without the curvature profile. Polynomial controls use the common 500k cap and fixed recipes.

Comparison	Improvement	Paired 95% interval	Wins/n
ZINC (MAE reduction)			
GINE: full vs. none	$+0.06636$	$[0.05796, 0.07468]$	10/10
GINE: full vs. cycles	$+0.01148$	$[0.00849, 0.01458]$	10/10
GSN-e: full vs. cycles	$+0.00066$	$[-0.00351, 0.00409]$	7/10
RICCIMP vs. GINE (both full)	-0.00455	$[-0.00757, -0.00135]$	2/10
RICCIMP vs. GSN-e (both full)	-0.00657	$[-0.01070, -0.00275]$	2/10
GINE (full) vs. GSN-e (cycles)	-0.00137	$[-0.00445, 0.00174]$	4/10
LRGB: adding the curvature profile			
Pascal: GPS	$+0.54$	$[-0.48, 1.83]$	6/10
Pascal: GatedGCN	-0.86	$[-1.09, -0.60]$	0/10
Peptides-func: GPS	-0.07	$[-0.74, 0.57]$	6/10
Peptides-func: GatedGCN	-0.90	$[-1.40, -0.41]$	1/10
Peptides-struct: GPS	-0.00191	$[-0.00354, -0.00028]$	1/5
Peptides-struct: GatedGCN	$+0.00092$	$[-0.00003, 0.00213]$	4/5
RICCIMP *vs. curvature-profile GINE polynomial,* $K = 3$			
Pascal	$+0.90$	$[0.39, 1.50]$	8/10
Peptides-func	-1.18	$[-1.60, -0.72]$	0/10
Peptides-struct	-0.00646	$[-0.00799, -0.00520]$	0/5

H.2 CONDUCTANCE AND CURVATURE-PROFILE PATHWAYS

Table 13 fixes width, depth, input, and optimization while changing polynomial order and conductance. We initialize adaptive conductance from the curvature profile and update it across layers, whereas trainable static conductance learns its initialization but holds the weights fixed across layers. By setting those weights to one while retaining curvature-based gates and cycle-redundancy normalization, the unit control isolates the conductance pathway within the curvature-aware layer. For the static control in this table, we clip the initial log-conductance to $[-4, 4]$ and hold it fixed across layers. Adaptive updates use the same bounds. All 70 bounded-static fits complete with finite validation and test scores. Appendix H.6 reports the numerical sensitivity to removing this bound.

Neither learned static weights nor adaptive updates show a clear additional benefit on these tasks. All eight paired 95% Student-t intervals for static versus unit weights include zero, as do all eight for adaptive versus static weights. None of the 24 conductance contrasts passes Holm correction at 0.05, leaving both an additional benefit and equivalence unresolved. Unit weights retain most of the mean order gain on Pascal and Peptides-func, while preserving the other curvature-dependent pathways.

H.2.1 INPUT AND NORMALIZATION

Table 14 masks the curvature-profile input and cycle-redundancy normalization independently. The normalizer uses $C_3/2$ and learned positive weights on $C_r/(r-1)$ for $r = 4, 5, 6$. Setting the normalizer to "Off" zeros its redundancy vector, leaving the unit denominator. Setting the curvature profile to "Off" masks its input coordinates but retains native attributes. Allocated parameter dimensions stay fixed, including inactive weights, and masking one pathway leaves the other open

Table 13: Propagation order and conductance at fixed width 64 and dataset-specific training settings. Mean ± sample SD over ten seeds for Pascal and Peptides-func and five for Peptides-struct. Static and adaptive conductances both use log bounds $[-4, 4]$. All cells are complete. Bold marks the best mean at the same order and dataset, without implying significance. A dash denotes an unrun order.

K	Conductance	Pascal F1 ↑	Peptides-func AP ↑	Peptides-struct MAE ↓
1	Unit	$\mathbf{32.09} \pm 0.49$	$\mathbf{63.62} \pm 0.79$	0.25661 ± 0.00271
1	Bounded static	31.62 ± 0.86	63.51 ± 0.59	0.25712 ± 0.00267
1	Adaptive	31.67 ± 0.48	63.57 ± 0.77	$\mathbf{0.25522} \pm 0.00279$
2	Unit	—	65.68 ± 0.61	—
2	Bounded static	—	$\mathbf{66.01} \pm 1.22$	—
2	Adaptive	—	65.90 ± 0.70	—
3	Unit	37.57 ± 0.72	66.06 ± 0.94	0.25430 ± 0.00198
3	Bounded static	37.48 ± 0.79	66.30 ± 0.80	0.25401 ± 0.00221
3	Adaptive	$\mathbf{37.74} \pm 0.94$	$\mathbf{66.63} \pm 0.72$	$\mathbf{0.25395} \pm 0.00121$
8	Unit	40.14 ± 0.79	—	—
8	Bounded static	40.10 ± 1.24	—	—
8	Adaptive	$\mathbf{40.26} \pm 1.04$	—	—

to structural information. The small, mixed effects leave training's ability to realize the theoretical complementarity unresolved.

Table 14: Curvature-profile input × cycle-redundancy normalization at $K = 3$. We hold all other settings and parameter dimensions fixed. Mean ± sample SD, ten seeds (five on Peptides-struct). The best mean per dataset is bold. Off masks the designated input or denominator while preserving allocated parameter dimensions.

Curvature profile	Normalizer	Pascal F1 ↑	Peptides-func AP ↑	Peptides-struct MAE ↓
Off	Off	37.47 ± 0.48	66.04 ± 0.85	$\mathbf{0.25326} \pm 0.00240$
On	Off	$\mathbf{37.94} \pm 0.58$	66.47 ± 0.75	0.25459 ± 0.00086
Off	On	36.96 ± 0.99	66.25 ± 0.97	0.25424 ± 0.00161
On	On	37.74 ± 0.94	$\mathbf{66.63} \pm 0.72$	0.25395 ± 0.00121

H.3 Order versus depth and simpler propagation

Table 15 compares $K = 3$ with three times as many $K = 1$ nonlinear layers, giving equal maximum hop reach. Fixed-width comparisons use width 64, whereas common-cap comparisons reduce the deeper models' widths, so parameter counts and compute remain unequal. The fixed-width and common-cap $K = 3$ fits are separate repetitions, even when the architectures coincide. For Table 16, each GINE block first performs one message-passing step. We then combine its output with up to $K - 1$ unweighted random-walk steps using learned signed coefficients, before the residual update. The resulting maximum reach is K hops per block. Because the message and update maps change along with conductance, this comparison measures their combined effect.

Table 15: Propagation order versus three times as many nonlinear layers. Each pair has the same maximum KL-hop reach. Fixed-width comparisons use $d = 64$, while common-cap comparisons choose the largest width in multiples of 16 that satisfies the 500k parameter cap. Actual sizes and runtime differ, so the common cap is not exact parameter or compute matching. Mean $\pm$ sample SD over ten seeds (five on Peptides-struct). Bold compares only the two rows within each pair.

Budget	K	L	d	Parameters	Test score	Min.
PascalVOC-SP: F1 ↑						
Fixed width	3	10	64	484,201	$\mathbf{37.74} \pm 0.94$	25.70
Fixed width	1	30	64	1,422,151	37.60 ± 0.69	55.45
500k cap	3	10	64	484,201	$\mathbf{37.94} \pm 0.73$	25.84
500k cap	1	30	32	404,583	35.47 ± 0.76	48.41
Peptides-func: AP ↑						
Fixed width	3	6	64	373,038	$\mathbf{66.63} \pm 0.72$	10.94
Fixed width	1	18	64	935,808	63.75 ± 1.02	20.61
500k cap	3	6	64	373,038	$\mathbf{66.24} \pm 0.59$	11.18
500k cap	1	18	32	284,352	61.92 ± 0.72	20.58
Peptides-struct: MAE ↓						
Fixed width	3	8	64	469,723	$\mathbf{0.25395} \pm 0.00121$	12.59
Fixed width	1	24	64	1,220,083	0.25407 ± 0.00129	24.93
500k cap	3	8	64	469,723	$\mathbf{0.25414} \pm 0.00160$	13.00
500k cap	1	24	32	364,755	0.26135 ± 0.00343	25.43

Table 16: A simpler unweighted polynomial after curvature-profile GINE, compared with common-cap $K = 3$ RICCIMP. All rows obey the 500k cap and the fixed dataset recipe. Mean $\pm$ sample SD over ten seeds (five on Peptides-struct). The best listed mean per dataset is bold. The $K = 3$ comparison has equal per-layer propagation order.

Model	K	d	Parameters	Test score	Min.
PascalVOC-SP: F1 ↑					
GINE + curvature profile	1	120	487,855	30.23 ± 0.58	12.15
GINE + curvature profile	3	120	487,885	37.04 ± 0.83	16.83
RICCIMP	3	64	484,201	$\mathbf{37.94} \pm 0.73$	25.84
Peptides-func: AP ↑					
GINE + curvature profile	1	124	478,752	65.17 ± 0.76	6.03
GINE + curvature profile	3	124	478,770	$\mathbf{67.42} \pm 0.66$	7.08
RICCIMP	3	64	373,038	66.24 ± 0.59	11.18
Peptides-struct: MAE ↓					
GINE + curvature profile	1	112	491,299	0.25130 ± 0.00093	6.03
GINE + curvature profile	3	112	491,323	$\mathbf{0.24769} \pm 0.00067$	7.54
RICCIMP	3	64	469,723	0.25414 ± 0.00160	13.00

H.4 DENOISING AND TOPOLOGY TRANSFER

On the 32-node two-community rook graph, the target is a Gaussian amplitude times the community-sign vector. At noise level $\sigma = 1$, we subtract the within-community means of independent standard Gaussian nuisance. Each example receives an independent vertex permutation, and the models receive neither IDs nor community labels. Each seed uses the same 2,048 training, 256 validation, and 512 test examples across arms. Training uses 200 epochs, AdamW at 10^{-3}, zero weight decay, batch size 64, cosine scheduling, and validation-MSE selection.

All models propagate node features over at most two hops and use width 32. RICCIMP uses one $K = 2$ layer, while GINE/GatedGCN use two ordinary layers. Parameter counts are 15,475 for each RICCIMP arm, 8,739 for GINE, 8,835 for curvature-profile GINE, and 13,313 for curvature-profile GatedGCN. Unit-conductance RICCIMP retains curvature-based gates and triangle normalization, so it is not the curvature-free scalar theorem comparator. Test graphs receive one or four degree-preserving switches within each community, preserving the matching between communities. We recompute structural inputs without retraining. Table 17 gives the values underlying the transfer plot.

Table 17: Rook-graph denoising at noise level $\sigma = 1$: test MSE, mean $\pm$ sample SD over ten seeds. Held-out graphs receive one or four degree-preserving switches per community. Bold marks the lowest listed mean in each condition.

Model	Original graph	One switch	Four switches
RICCIMP, unit	$1.93\text{e}{-2} \pm 8.3\text{e}{-4}$	$2.20\text{e}{-2} \pm 6.8\text{e}{-4}$	$3.06\text{e}{-2} \pm 9.7\text{e}{-4}$
RICCIMP, trainable static	$6.59\text{e}{-5} \pm 8.9\text{e}{-6}$	$9.71\text{e}{-3} \pm 1.6\text{e}{-3}$	$4.79\text{e}{-2} \pm 1.2\text{e}{-2}$
RICCIMP, adaptive	$5.83\text{e}{-5} \pm 1.1\text{e}{-5}$	$9.98\text{e}{-3} \pm 1.4\text{e}{-3}$	$4.96\text{e}{-2} \pm 1.1\text{e}{-2}$
GINE	$6.26\text{e}{-3} \pm 5.1\text{e}{-4}$	$9.47\text{e}{-3} \pm 3.9\text{e}{-4}$	$\mathbf{1.56}\text{e}{-\mathbf{2}} \pm 3.8\text{e}{-4}$
GINE + curvature profile	$6.29\text{e}{-5} \pm 1.1\text{e}{-5}$	$1.16\text{e}{-2} \pm 3.9\text{e}{-3}$	$6.73\text{e}{-2} \pm 2.7\text{e}{-2}$
GatedGCN + curvature profile	$\mathbf{1.68}\text{e}{-\mathbf{5}} \pm 4.2\text{e}{-6}$	$\mathbf{4.96}\text{e}{-\mathbf{3}} \pm 3.7\text{e}{-4}$	$1.71\text{e}{-2} \pm 7.5\text{e}{-4}$

What the denoising experiment establishes. By design, subtracting each community's noise mean makes exact community averaging recover the target. Training and unperturbed testing share the same topology up to vertex relabeling. Learning conductances improves on unit weights in this controlled setting, and trainable static weights capture most of the benefit. Curvature-profile backbones attain similarly low or lower error. With one feature-dependent conductance update before propagation, the one-layer adaptive model leaves the benefit of repeated updates across depth untested. After four topology switches per community, both learned-conductance variants lose to unit weights in all ten seeds. The original-graph improvement supports learning weights in this regime, while the transfer failures limit its generality. Neither bounded conductances nor the scalar filtering theorem guarantees transfer.

What the fitted weights show. At $\sigma = 1$, the mean fitted conductances are of order one. Here, adaptive conductance averages 1.214 on within-community edges and 1.027 on matching edges between communities. Static conductance averages 1.248 and 1.065, respectively. Every seed assigns higher mean conductance to internal edges, consistent with weighting connections that preserve the target. The cause of the error reduction remains unresolved because conductance also changes aggregate scale, the cycle-redundancy normalizer affects effective weights, and the comparison lacks a learned scalar-only control.

H.5 LEARNED-CONDUCTANCE ABLATION ACROSS REGIMES

Table 18 compares unit, learned-static, and adaptive conductance within RICCIMP on community recovery and node classification. These component ablations complement denoising (Table 17) and long-range prediction (Table 13) under their respective task protocols. The community and node comparisons leave the static initializer unclipped, whereas adaptive updates clip log-conductance to $[-4, 4]$. Their adaptive–static differences therefore combine the effects of adaptation and numerical bounds. The long-range comparison matches those bounds.

Community recovery. We use eight within-community and two cross-community neighbors, feature SNR 0.3, and the training protocol of Appendix G.4, with width 64, four layers, $R = 4$, and $K = 1$. The "neutral" condition uses that graph distribution without a triangle-based intervention. The

Table 18: Within-RICCIMP conductance ablation at $K = 1$, width 64, and four layers. Community ARI ↑ reports mean ± sample SD over 20 test graphs after averaging three initializations. Roman-empire accuracy (%) ↑ reports mean ± sample SD over ten official splits. Static weights are learned but fixed across layers. These static fits omit the log-conductance bound used in Table 13. Bold marks the best mean within each row, not statistical significance.

Setting	Unit	Learned static	Adaptive
Community, aligned triangles	0.100 ± 0.046	0.176 ± 0.072	$\mathbf{0.189} \pm 0.066$
Community, neutral triangles	0.121 ± 0.058	0.238 ± 0.094	$\mathbf{0.280} \pm 0.125$
Roman-empire	87.979 ± 0.515	$\mathbf{88.371} \pm 0.653$	88.343 ± 0.646

"aligned" condition adds 2,000 attempted degree-preserving switches and accepts moves that do not decrease the mean triangle-count gap between within- and between-community edges. Each vertex retains its degree into every block. Planted labels guide graph construction only, not the loss or validation selection. All three aligned arms select learning rate 10^{-3}. Both learned variants beat unit weights for each of the three training seeds. The adaptive–unit ARI gain is 0.089, with paired 95% Student-t interval $[0.070, 0.107]$ across 20 test graphs after averaging initializations. The paired interval conditions on the fitted models and one training-graph set. Static weights capture much of the gain. The neutral row also favors learning, but validation selects different learning rates for some variants. We selected these favorable examples from a broader 18-regime grid, which crosses within-/between-community degrees $8/2$, $6/4$, and $4/6$, SNR 0 or 0.3, and neutral, aligned, or opposed triangle interventions. The opposed intervention reverses the triangle-gap acceptance criterion. Only four adaptive–unit ARI gains at $K = 1$, including the two shown here, survive a conservative Holm adjustment over 4,644 comparisons across regimes, models, and three recovery metrics (ARI, normalized mutual information, and accuracy after optimal label matching). This selection does not establish a general advantage or identify triangle alignment alone as a predictor of success.

Node classification. This ablation uses width 64, four layers, $R = 4$, and ten official splits, separately from the wider architecture comparison in Appendix G.2. Each arm selects a learning rate from $\{10^{-3}, 3 \times 10^{-3}\}$ and dropout from $\{0.2, 0.5\}$ by mean validation accuracy, with AdamW, weight decay 10^{-5}, at most 300 epochs, and patience 40. At $K = 1$, all three Roman-empire arms select 3×10^{-3} and dropout 0.2. Static and adaptive weights improve on unit weights by 0.392 and 0.364 accuracy points, respectively, each winning nine of ten splits. Both have higher mean accuracy than unit weights across the eight combinations of $K \in \{1, 3\}$ and the four learning-rate/dropout settings. These comparisons share graph splits and are not independent replications. The selected gains do not survive Holm correction across the 18 contrasts from three datasets, two orders, and three conductance pairs. Chameleon and Squirrel show no comparable pattern. We treat Roman-empire as a modest task-specific hint, with no clear adaptive increment at the selected $K = 1$ recipe.

These favorable cases motivate the hypothesis that learned weights help when local structure distinguishes connections that preserve the target from those that mix unrelated signals. Testing it requires separating relative edge weighting from aggregate scale and explaining the topology-transfer failure before expecting the gains to persist across graphs.

H.6 RUNTIME AND NUMERICAL SENSITIVITY

Runtimes include process startup, training, validation, and test evaluation of the validation-selected checkpoint, using cached structural inputs. Table 19 measures fresh counting separately on 32 uniformly sampled training graphs per dataset. These sequential CPU timings exclude initialization, loading, and other structural encodings, so they do not estimate full-dataset preprocessing time. Sampling uses seed 20260921 and the dataset-specific deterministic stream in the supplied sampler, without replacement. This timing environment used Python 3.10.20, NumPy 2.2.6, SciPy 1.15.2, and Numba 0.67.0 with four threads; the exact-cycle routines were compiled before measurement.

Sensitivity to numerical bounds. Without clipping the static initializer, 16 Peptides-func fits fail, leaving 9/10, 2/10, and 3/10 successful seeds at $K = 1, 2, 3$. With log-conductance clipped to $[-4, 4]$, all 70 static fits across the eight dataset/order settings complete with finite selected validation

Table 19: Fresh exact $R = 6$ curvature-profile construction on 32 uniformly sampled training graphs per dataset. Milliseconds per graph on an Intel Xeon Platinum 8480CL, measured sequentially after initialization. This excludes cache loading, compilation, and additional encodings. These are sampled graph timings, not full-dataset preprocessing estimates.

Dataset	Graphs	Mean ms	Median ms	Maximum ms
PascalVOC-SP	32	18.442	18.590	20.444
Peptides-func	32	0.996	0.984	1.293
Peptides-struct	32	1.043	1.033	1.286
ZINC	32	0.848	0.835	1.113

and test scores. Table 13 uses only these bounded-static fits, alongside 140 unit and adaptive controls. We exclude success-conditioned means from the incomplete unbounded settings. The bound also affects completed comparisons. On Peptides-struct at $K = 1$, the unbounded-static MAE is 0.25406, compared with 0.25712 for bounded static and 0.25661 for unit conductance. The apparent advantage of unbounded static over unit weights therefore does not persist with matched bounds. Finite completion on this grid leaves general numerical stability unresolved.

I Details on Experimental Setups

For reproducibility, we summarize the software, hardware, and datasets used in the experiments. The recorded H200 training environment used PyTorch 2.5.1+cu124 (CUDA 12.4), PyTorch Geometric 2.7.0, NumPy 2.2.6, SciPy 1.15.3, and pandas 2.3.3. Graph construction and synthetic generation use NetworkX and explicit graph-construction utilities in the supplied code. The learned pipelines use AdamW (Loshchilov & Hutter, 2019); Appendix G.1 lists additional dependencies, and the preceding protocols specify initialization seeds, optimization settings, and model-specific conventions.

Table 20 describes the H200 training system. Each fit uses one GPU; CPU and RAM specifications describe the host rather than the resources allocated to an individual fit. Fresh preprocessing timings use a separate CPU environment, with SciPy 1.15.2 and four threads, as detailed in Appendix H.6.

Table 20: Hardware specifications of the H200 training system.

Component	Specification
Architecture	x86_64
OS	Rocky Linux 8.10 (Green Obsidian)
CPU	AMD EPYC 9374F 32-Core Processor ($\times 2$)
GPU	NVIDIA H200
GPU memory	143771 MiB (approximately 140.4 GiB)
Host RAM	1.5 TiB

The datasets cover molecular property prediction (ZINC), long-range image-region labeling and peptide prediction (PascalVOC-SP, Peptides-func, Peptides-struct) (Dwivedi et al., 2023; 2022b), node classification (Cora, CiteSeer, Roman-empire, and the filtered Chameleon and Squirrel variants), and graph classification (MUTAG, ENZYMES, PROTEINS, IMDB-BINARY). The task-specific split and preprocessing rules are specified in Appendices G and H. Synthetic experiments use planted-community graphs for community recovery and structured graphs with noisy signals for denoising and topology transfer; their generators and evaluation protocols are described in Appendices G.4 and H.4. Table 21 lists the main libraries' licenses.

Table 21: Main software licenses. Library names link to the license files.

Software	License
PyTorch	BSD-3-Clause
PyTorch Geometric	MIT
NetworkX	BSD-3-Clause
NumPy	BSD-3-Clause
SciPy	BSD-3-Clause
pandas	BSD-3-Clause