PROVABLE GLOBAL REACH VIA EFFECTIVE-RESISTANCE-WEIGHTED PROPAGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Message-passing graph neural networks (GNNs) have a finite receptive field and can struggle to transmit information across long graph distances and structural bottlenecks. Although existing remedies such as graph rewiring or virtual nodes can bypass these issues, they may alter the original communication geometry or mediate global interactions through a shared latent state. To overcome this with theoretical support, we propose ER-GCN, an architecture that augments local graph convolution with a global propagation channel derived from effective resistance. ER-GCN assigns positive weights to every pair of distinct nodes with weights decreasing as effective resistance increases. Afterwards, ER-GCN row-normalizes the resulting similarity matrix to obtain a global propagation operator. Theoretically, our node-wise Jacobian support analysis shows that augmenting local propagation with this effective-resistance-based global propagation operator in at least one layer can remove topology-induced structural zeros between distinct node pairs. Empirical evaluations on synthetic bottleneck benchmarks and real-world long-range graph datasets are consistent with our theoretical analyses that ER-GCN improves performance most consistently when long-range information is important, while ER-GCN remaining competitive on tasks dominated by local structure. Ablation studies further support the importance of dense global support and suggest a benefit from resistance-aware weighting.

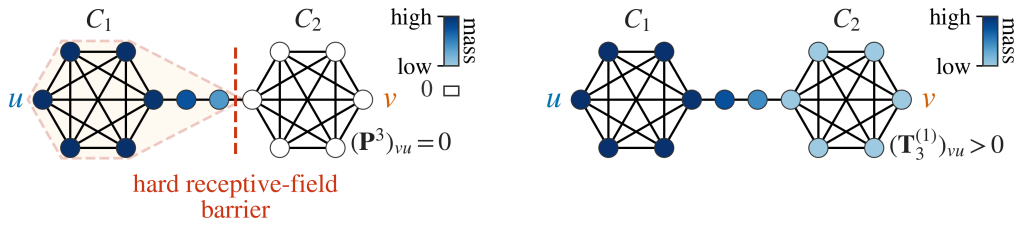
1 INTRODUCTION

Message-passing graph neural networks (MP-GNNs) face two related but distinct limitations in long-range learning. First, an L-layer local MP-GNN has a hard L-hop receptive field: nodes separated by more than L edges have exactly zero structural influence on one another. Second, even when a source node lies within the receptive field of a target node, its influence may be strongly attenuated when information from many nodes must be compressed into fixed-dimensional representations or transmitted through narrow structural bottlenecks. The latter phenomenon is commonly associated with over-squashing (Alon & Yahav, 2021; Di Giovanni et al., 2023). Increasing the network depth expands the receptive field, but may introduce optimization difficulties, over-smoothing, and further attenuation of long-range dependencies (Chen et al., 2020; Di Giovanni et al., 2023).

Existing approaches broaden graph communication in several ways. Deeper, decoupled, and memory-enhanced architectures improve optimization and information preservation over many propagation steps, such as JKNet (Xu et al., 2018), APPNP (Gasteiger et al., 2019), GCNII (Chen et al., 2020), AMP (Errica et al., 2025), and gLSTM (Blayney et al., 2026). These methods facilitate deeper information propagation, but continue to communicate primarily through local neighborhoods. Graph-rewiring methods instead shorten communication paths by modifying the graph or its layer-dependent neighborhoods (Topping et al., 2022; Gutteridge et al., 2023). Virtual-node and graph-transformer architectures provide graph-wide communication through a shared latent state or learned global interactions (Gilmer et al., 2017; Rampášek et al., 2022; Southern et al., 2025). These approaches provide complementary mechanisms for facilitating long-range communication beyond conventional local message passing.

We propose **ER-GCN**, which complements local graph convolution with a fixed global propagation operator derived from pairwise effective resistance. Applying a positive decreasing kernel to the resistances and row-normalizing the resulting similarities yields a dense operator $\boldsymbol{Q}$, where struc-

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.



(a) Hard receptive-field barrier under three local propagation layers.

(b) Removal of the structural zero by one ER-GCN hybrid layer.

Figure 1: Illustration of the hard finite-receptive-field barrier and its removal by ER-GCN on a barbell graph. Node colors indicate propagation mass originating from node u on a logarithmic scale. **Left:** After three purely local propagation steps, mass remains confined to the three-hop neighborhood of u. Since v lies outside this neighborhood, $(\boldsymbol{P}^3)_{vu} = 0$. **Right:** Replacing the middle local layer by the hybrid operator gives $\boldsymbol{T}_3^{(1)} = \boldsymbol{PMP}$. Because $\boldsymbol{Q}$ has strictly positive off-diagonal support, $(\boldsymbol{T}_3^{(1)})_{vu} > 0$, providing a candidate propagation path from u to v.

turally well-connected pairs receive larger weights and $Q_{uv} > 0$ for every $u \neq v$. ER-GCN thus enables direct nonlocal communication without rewiring the graph or replacing its original local operator. Figure 1 illustrates the hard receptive-field barrier and a hybrid layer removes the barrier of local propagation.

Our contributions are summarized as follows: (i) we introduce a dense and resistance-weighted propagation channel and derive its exponential kernel as the unique solution of an entropy-regularized resistance-minimization problem; (ii) we prove that inserting one hybrid layer at any depth removes all topology-induced off-diagonal structural zeros, and characterize the resulting Jacobian support for ER-GCN with the $\tanh$ activations; and (iii) we show empirically that ER-GCN is most effective on tasks with long-range dependencies while remaining competitive on predominantly local benchmarks.

2 Related Work

Long-range communication in message-passing GNNs is constrained by finite-depth local propagation and can be further weakened by structural bottlenecks. Prior work has studied the latter phenomenon through the lens of over-squashing, graph curvature, and the interaction between graph topology and network architecture (Alon & Yahav, 2021; Topping et al., 2022; Di Giovanni et al., 2023). A major family of remedies changes the communication structure. DRew (Gutteridge et al., 2023) constructs layer-dependent neighborhoods, while the densification–sparsification framework of Liang et al. (2025) first improves connectivity through densification and then sparsifies the resulting graph while approximately preserving its spectral structure. These methods shorten communication paths by modifying the graph used for message passing. In contrast, ER-GCN retains the original local propagation operator and adds a separate topology-aware global channel.

Virtual-node methods introduce a shared graph-level state through which distant nodes can exchange information (Gilmer et al., 2017). GraphGPS combines local message passing with global attention and structural or positional encodings (Rampášek et al., 2022). VN_G makes this communication topology-aware using structure-dependent node sensitivity (Southern et al., 2025). These methods provide graph-wide communication through either a shared latent state or learned global interactions. ER-GCN instead defines direct pairwise propagation through a fixed, row-stochastic matrix derived from graph topology; only the layer-wise local–global mixing coefficients are learned.

Effective resistance has previously been used to analyze over-squashing and graph bottlenecks (Black et al., 2023). It has also been incorporated into GNNs in several different roles. GTR uses effective resistance as an objective for selecting rewiring edges (Black et al., 2023); affinity-aware architectures use resistance-derived quantities as structural features (Velingker et al., 2023); and resistance distance has been studied as a positional encoding (Black et al., 2024). ER-GCN differs

in how effective resistance enters the computation: it converts all pairwise resistances into a dense row-stochastic propagation operator and combines that operator directly with local graph convolution. This gives every pair of distinct nodes positive structural propagation support without changing the original local graph or learning an all-pairs attention matrix.

3 Problem Setup and Theoretical Analysis

3.1 Problem Setup

Let $G = (V, E)$ be a connected undirected graph with input features $\boldsymbol{X} = \boldsymbol{H}^{(0)}$. Let $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$, $\tilde{\boldsymbol{D}} = \mathrm{diag}(\tilde{\boldsymbol{A}}\mathbf{1})$, and $\boldsymbol{P} = \tilde{\boldsymbol{D}}^{-1}\tilde{\boldsymbol{A}}$ denote the row-normalized self-looped propagation matrix. A GCN-style L-layer local message-passing model can be written as

$$\boldsymbol{H}^{(l+1)} = \phi_l\Big(\boldsymbol{P}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}\Big) \tag{1}$$

for $l = 0, \ldots, L-1$.

Writing node representations as column vectors, define $\boldsymbol{h}_v^{(l)} := \left(\boldsymbol{H}_{v,:}^{(l)}\right)^\top$ and $\boldsymbol{x}_u := \boldsymbol{h}_u^{(0)}$. For nodes u, v, we define the information sensitivity from u to v as

$$s_{u\to v}^{(L)} = \left\| \frac{\partial \boldsymbol{h}_v^{(L)}}{\partial \boldsymbol{x}_u} \right\|_F. \tag{2}$$

Because $\boldsymbol{P}$ is supported only on graph edges and self-loops, and the nonlinearities act independently at each node, an L-layer local Classical GNN can depend only on inputs within L graph hops (Alon & Yahav, 2021; Di Giovanni et al., 2023). Hence, wherever the Jacobian exists,

$$d_G(u, v) > L \quad \Longrightarrow \quad \frac{\partial \boldsymbol{h}_v^{(L)}}{\partial \boldsymbol{x}_u} = \mathbf{0} \quad \Longrightarrow \quad s_{u\to v}^{(L)} = 0. \tag{3}$$

A formal derivation based on the computational-path expansion is provided in Appendix A.1.1. Thus, purely local message passing induces topology-dependent structural zeros that cannot be removed by learning the feature transformations. Our goal is to eliminate this hard locality constraint without discarding the original local operator, while using graph topology to control the strength of nonlocal interactions.

3.2 Receptive-Field and Jacobian-Support Analysis

We analyze the nonlinear architecture directly through node-wise feature Jacobians. The sensitivity $s_{u\to v}^{(L)}$ defined in (2) measures the local sensitivity of the final representation at node v to perturbations of the input features at node u. A small value indicates that the representation at v is locally insensitive to information from u, making the corresponding dependency difficult to transmit and learn (Di Giovanni et al., 2023).

We now characterize how hybrid local–global propagation changes the structural support of the node-wise Jacobian underlying (2). Let $\boldsymbol{Q}$ denote the effective-resistance-weighted global operator constructed in Section 4, and let $\mathcal{K} \subseteq \{0, \ldots, L-1\}$ denote the set of layers at which resistance-weighted global propagation is used. We omit the superscript $\mathcal{K}$ when it is clear from context. For notational simplicity, we use a common mixing coefficient β throughout the theoretical analysis and define

$$\boldsymbol{S}_l^{(\mathcal{K})} = \begin{cases} \boldsymbol{M} = (1-\beta)\boldsymbol{P} + \beta\boldsymbol{Q}, & l \in \mathcal{K}, \\ \boldsymbol{P}, & l \notin \mathcal{K}. \end{cases}$$

Throughout this subsection, we assume that G is connected and that $\boldsymbol{P}$ and $\boldsymbol{Q}$ are entrywise nonnegative. We further assume that $P_{ji} = 0$ whenever $i \neq j$ and $\{i, j\} \notin E$, that $P_{ii} > 0$ for every $i \in V$, that $Q_{ji} > 0$ for every $i \neq j$, and that $0 \leq \beta \leq 1$.

Under these assumptions, we next introduce the notation needed for the computational-path and Jacobian-support analysis.

Given $\boldsymbol{H}^{(0)} = \boldsymbol{X}$, consider the layer updates $\boldsymbol{Z}^{(l+1)} = \boldsymbol{S}_l^{(\mathcal{K})} \boldsymbol{H}^{(l)} \boldsymbol{W}^{(l)}$ and $\boldsymbol{H}^{(l+1)} = \phi_l\Big(\boldsymbol{Z}^{(l+1)}\Big)$ where ϕ_l is the activation function and $l = 0, \dots, L-1$. For each node i, define the column-vector representations $\boldsymbol{h}_i^{(l)} := (\boldsymbol{H}_{i,:}^{(l)})^\top$ and $\boldsymbol{z}_i^{(l+1)} := (\boldsymbol{Z}_{i,:}^{(l+1)})^\top$. At every point where ϕ_l is differentiable, let $\boldsymbol{G}_i^{(l+1)} := \partial \boldsymbol{h}_i^{(l+1)} / \partial \boldsymbol{z}_i^{(l+1)}$ denote the local activation Jacobian. Since ϕ_l is applied coordinatewise, we have $\boldsymbol{G}_i^{(l+1)} = \operatorname{diag}\big(\phi_l'(\boldsymbol{z}_i^{(l+1)})\big)$.

For a source node u and an intermediate or target node j, let $\mathcal{J}_{j \leftarrow u}^{(l)} := \partial \boldsymbol{h}_j^{(l)} / \partial \boldsymbol{x}_u \in \mathbb{R}^{d_l \times d_0}$, where $\boldsymbol{x}_u := \boldsymbol{h}_u^{(0)}$. In particular, $s_{u \to v}^{(L)} = \|\mathcal{J}_{v \leftarrow u}^{(L)}\|_F$.

Finally, define the effective propagation operator associated with $\mathcal{K}$ by

$$\boldsymbol{T}_L^{(\mathcal{K})} := \boldsymbol{S}_{L-1}^{(\mathcal{K})} \cdots \boldsymbol{S}_0^{(\mathcal{K})}.$$

This operator summarizes the aggregate structural propagation weights over all node-level computational paths. The following proposition connects these paths to the realized node-to-node Jacobian.

Proposition 3.1 (Computational-path expansion and structural Jacobian support)**.** *At every point where the activations are differentiable, the node-to-node Jacobian admits the expansion*

$$\mathcal{J}_{v \leftarrow u}^{(L)} = \sum_{\substack{i_0 = u,\ i_L = v \\ i_1, \dots, i_{L-1} \in V}} \left(\prod_{l=0}^{L-1} (S_l)_{i_{l+1} i_l} \right) \boldsymbol{G}_{i_L}^{(L)} \boldsymbol{W}^{(L-1)\top} \cdots \boldsymbol{G}_{i_1}^{(1)} \boldsymbol{W}^{(0)\top}. \tag{4}$$

Moreover, the corresponding entry of the effective propagation operator is

$$\Big(\boldsymbol{T}_L^{(\mathcal{K})}\Big)_{vu} = \sum_{\substack{i_0 = u,\ i_L = v \\ i_1, \dots, i_{L-1} \in V}} \prod_{l=0}^{L-1} \Big(S_l^{(\mathcal{K})}\Big)_{i_{l+1} i_l}.$$

Consequently,

$$\Big(\boldsymbol{T}_L^{(\mathcal{K})}\Big)_{vu} = 0 \quad \Longrightarrow \quad \mathcal{J}_{v \leftarrow u}^{(L)} = \boldsymbol{0}. \tag{5}$$

Equivalently,

$$\Big\{(u, v) : \Big\|\mathcal{J}_{v \leftarrow u}^{(L)}\Big\|_F > 0\Big\} \subseteq \Big\{(u, v) : \Big(\boldsymbol{T}_L^{(\mathcal{K})}\Big)_{vu} > 0\Big\}.$$

Proof sketch. The chain rule gives $\mathcal{J}_{j \leftarrow u}^{(l+1)} = \boldsymbol{G}_j^{(l+1)} \boldsymbol{W}^{(l)\top} \sum_{i \in V} (S_l)_{ji} \mathcal{J}_{i \leftarrow u}^{(l)}$. Iterating this recursion yields the path expansion, while ordinary matrix multiplication gives the corresponding expansion of $(\boldsymbol{T}_L^{(\mathcal{K})})_{vu}$. Since all propagation coefficients are nonnegative, a zero entry of $\boldsymbol{T}_L^{(\mathcal{K})}$ forces every associated path coefficient to vanish. Every Jacobian summand therefore vanishes as well. See Appendix A.1 for the full proof.

Thus, $\boldsymbol{T}_L^{(\mathcal{K})}$ provides a structural upper envelope for the support of the realized node-to-node Jacobian. The converse need not hold: positive structural propagation support may still yield a zero Jacobian because of inactive activation gates or cancellation among signed feature-path contributions. We next specialize this characterization to a single hybrid layer. When the hybrid layer is placed at depth $k \in \{0, \dots, L-1\}$, write

$$\boldsymbol{T}_L^{(k)} := \boldsymbol{T}_L^{(\{k\})}.$$

In this case,

$$\boldsymbol{T}_L^{(k)} = \boldsymbol{P}^{L-1-k} \boldsymbol{M} \boldsymbol{P}^k = (1-\beta) \boldsymbol{P}^L + \beta \boldsymbol{P}^{L-1-k} \boldsymbol{Q} \boldsymbol{P}^k. \tag{6}$$

Theorem 3.2 (Global reach from a single hybrid layer)**.** *Let $k \in \{0, \dots, L-1\}$, and suppose that layer k uses*

$$\boldsymbol{M} = (1-\beta) \boldsymbol{P} + \beta \boldsymbol{Q},$$

while every other layer uses $\boldsymbol{P}$. If $0 < \beta \le 1$, then for every pair of distinct nodes $u \ne v$,

$$\Big(\boldsymbol{T}_L^{(k)}\Big)_{vu} \ge \beta (P_{vv})^{L-1-k} Q_{vu} (P_{uu})^k > 0.$$

Consequently, a single hybrid layer removes the graph-induced hard finite-receptive-field barrier at the level of structural propagation support regardless of the placement depth k. Every ordered pair of distinct nodes admits a positive-weight computational path through the network.

In addition, if $0 < \beta < 1$, then

$$\boldsymbol{T}_L^{(k)} > 0$$

entrywise.

Proof sketch. Expanding $\boldsymbol{T}_L^{(k)} = (1-\beta)\boldsymbol{P}^L + \beta\boldsymbol{P}^{L-1-k}\boldsymbol{Q}\boldsymbol{P}^k$, consider the path that stays at u for k local steps, moves directly from u to v through $\boldsymbol{Q}$, and then stays at v. Its weight is $\beta(P_{vv})^{L-1-k}Q_{vu}(P_{uu})^k > 0$. When $0 < \beta < 1$, the all-local term similarly gives positive diagonal entries. See Appendix A.2 for the full proof. The same path-selection argument extends to any nonempty set of hybrid layers and yields the following corollary.

Corollary 3.3 (Multiple hybrid layers)**.** *Suppose that $\mathcal{K} \neq \varnothing$ and $0 < \beta < 1$. Then*

$$\boldsymbol{T}_L^{(\mathcal{K})} > 0$$

entrywise. More precisely, for every $k \in \mathcal{K}$ and every $u \neq v$,

$$\left(\boldsymbol{T}_L^{(\mathcal{K})}\right)_{vu} \geq \beta(1-\beta)^{|\mathcal{K}|-1}(P_{vv})^{L-1-k}Q_{vu}(P_{uu})^k > 0. \tag{7}$$

Proof sketch. Expand each hybrid factor as $\boldsymbol{M} = (1-\beta)\boldsymbol{P} + \beta\boldsymbol{Q}$. For any fixed $k \in \mathcal{K}$, retain the nonnegative product term that uses $\beta\boldsymbol{Q}$ at layer k and $(1-\beta)\boldsymbol{P}$ at every other hybrid layer. This gives

$$\boldsymbol{T}_L^{(\mathcal{K})} \succeq \beta(1-\beta)^{|\mathcal{K}|-1}\boldsymbol{P}^{L-1-k}\boldsymbol{Q}\boldsymbol{P}^k.$$

The off-diagonal bound follows by considering the path that stays at u, moves from u to v through $\boldsymbol{Q}$, and then stays at v. The diagonal entries are positive because every propagation factor has a strictly positive diagonal when $0 < \beta < 1$. See Appendix A.3 for the full proof.

Accordingly, the insertion depth affects the magnitude of the effective propagation weights but not their off-diagonal support. Under the stated positivity assumptions, a single hybrid layer at any depth yields a strictly positive effective propagation weight between every pair of distinct nodes, thereby eliminating topology-induced structural zeros among candidate propagation paths.

We next specialize to networks with $\tanh$ activations.

Theorem 3.4 (Realized Jacobian support for $\tanh$ networks)**.** *Suppose that*

$$\phi_l = \tanh, \qquad l = 0, \ldots, L-1,$$

and define

$$\boldsymbol{B}_L := \boldsymbol{W}^{(0)} \cdots \boldsymbol{W}^{(L-1)}.$$

For any fixed finite input, if all weight matrices are entrywise nonnegative and $\boldsymbol{B}_L \neq \boldsymbol{0}$, then

$$\left(\boldsymbol{T}_L^{(\mathcal{K})}\right)_{vu} > 0 \quad \Longleftrightarrow \quad \left\|\mathcal{J}_{v\leftarrow u}^{(L)}\right\|_F > 0.$$

Without the entrywise-nonnegativity restriction, whenever

$$\left(\boldsymbol{T}_L^{(\mathcal{K})}\right)_{vu} > 0,$$

the Jacobian block $\mathcal{J}_{v\leftarrow u}^{(L)}$ is nonzero for Lebesgue-almost every collection of weight parameters.

Combining Theorem 3.4 with Theorem 3.2, a single hybrid layer at any depth yields a nonzero realized $\tanh$ Jacobian between every pair of distinct nodes under the stated nonnegative-weight condition. The same all-pairs conclusion holds generically for unconstrained signed weights. The proof is given in Appendix A.4. **The analysis can be extended to ReLU networks** under a randomized path-gate model (Appendix A.5), showing that full-support hybrid propagation also yields strictly positive expected sensitivity in this setting.

4 EFFECTIVE-RESISTANCE-WEIGHTED GLOBAL PROPAGATION

For a connected undirected graph G with $n \geq 2$, let $\boldsymbol{L}$ be its unnormalized Laplacian. The effective resistance between nodes u and v is (Klein & Randić, 1993; Spielman & Srivastava, 2011)

$$R_{\text{eff}}(u, v) = (\boldsymbol{e}_u - \boldsymbol{e}_v)^\top \boldsymbol{L}^+ (\boldsymbol{e}_u - \boldsymbol{e}_v), \tag{8}$$

where $\boldsymbol{L}^+$ is the Moore–Penrose pseudoinverse of $\boldsymbol{L}$ and $\boldsymbol{e}_u$ is the u-th standard basis vector. Under unit edge resistances, this quantity summarizes path redundancy: it is smaller for well-connected pairs and larger across sparse bottlenecks. We map it to the following positive decreasing kernel function

$$g_\sigma(R) = \exp\left(-\frac{R}{\sigma}\right), \qquad \sigma > 0. \tag{9}$$

We define the resistance-weighted global propagation matrix as

$$\boldsymbol{Q} = \boldsymbol{D}_K^{-1} \boldsymbol{K}, \tag{10}$$

where $\boldsymbol{K} = [K_{uv}] \in \mathbb{R}^{n \times n}$ is the resistance-based similarity matrix,

$$K_{uv} = \begin{cases} g_\sigma(R_{\text{eff}}(u, v)), & u \neq v, \\ 0, & u = v, \end{cases} \tag{11}$$

and $\boldsymbol{D}_K = \operatorname{diag}(\boldsymbol{K}\mathbf{1})$ is the row-normalizing degree matrix. Equivalently,

$$Q_{uv} = \begin{cases} \dfrac{g_\sigma(R_{\text{eff}}(u, v))}{\displaystyle\sum_{w \neq u} g_\sigma(R_{\text{eff}}(u, w))}, & u \neq v, \\ 0, & u = v. \end{cases} \tag{12}$$

Recall that g_σ is a monotonically decreasing function. Because all entries in the same row are normalized by the same positive denominator, we have $R_{\text{eff}}(u, v) < R_{\text{eff}}(u, w) \iff Q_{uv} > Q_{uw}$. Thus, a source node with smaller effective resistance to the target node receives a larger propagation weight.

Moreover, because the graph is connected, the effective resistance between every pair of nodes is finite. Therefore, for any finite $\sigma > 0$, we have $g_\sigma(R_{\text{eff}}(u, v)) > 0$ when $u \neq v$. Since the row-normalizing denominators are strictly positive, it follows that

$$Q_{uv} > 0 \quad \forall u \neq v, \quad \text{and} \quad Q_{uu} = 0.$$

Hence, $\boldsymbol{Q}$ has an entrywise strictly positive off-diagonal part and a zero diagonal.

Variational interpretation. Our choice of the exponential kernel g_σ and its bandwidth σ has a direct variational justification. For each target node u, let Δ_u denote the probability simplex over $V \setminus \{u\}$, and let $\boldsymbol{\pi}_u$ be the uniform distribution on this set. Recall $(Q_{uv})_{v \neq u}$ is the off-diagonal part of the u-th row of $\boldsymbol{Q}$ (refer to Equation (12)). Then

$$\boldsymbol{q}_u^\star := (Q_{uv})_{v \neq u}$$

is the unique solution to the following optimization problem:

$$\boldsymbol{q}_u^\star = \operatorname*{arg\,min}_{\boldsymbol{q}_u \in \Delta_u} \left\{ \sum_{v \neq u} q_{uv} R_{\text{eff}}(u, v) + \sigma \operatorname{KL}(\boldsymbol{q}_u \| \boldsymbol{\pi}_u) \right\}. \tag{13}$$

The resistance term controls an upper bound on the distortion incurred when aggregating $\sum_{v \neq u} q_{uv} \boldsymbol{h}_v$ in place of $\boldsymbol{h}_u$, whereas the KL term discourages concentration and preserves full support. The bandwidth σ therefore acts as a temperature parameter controlling the trade-off between resistance-aware concentration and diffuse global propagation. The distortion bound and the proof of the characterization are given in Appendix A.6.

5 ER-GCN ARCHITECTURE

ER-GCN uses local–global hybrid propagation in every propagation layer. For each layer $l = 0, \ldots, L-1$, we define

$$\boldsymbol{M}_l = (1-\beta_l)\boldsymbol{P} + \beta_l \boldsymbol{Q}, \qquad \beta_l = \mathrm{sigmoid}(\theta_l), \tag{14}$$

where $\theta_l \in \mathbb{R}$ is a learnable scalar. Thus, $0 < \beta_l < 1$, with smaller values favoring local propagation through $\boldsymbol{P}$ and larger values favoring resistance-weighted global propagation through $\boldsymbol{Q}$.

The layer-dependent propagation operator is

$$\boldsymbol{S}^{(l)} = \boldsymbol{M}_l, \qquad l = 0, \ldots, L-1. \tag{15}$$

The L-layer network then updates

$$\boldsymbol{H}^{(l+1)} = \phi_l\Big(\boldsymbol{M}_l \boldsymbol{H}^{(l)} \boldsymbol{W}^{(l)}\Big), \qquad l = 0, \ldots, L-1, \tag{16}$$

where $\boldsymbol{H}^{(0)} = \boldsymbol{X}$ and $\boldsymbol{W}^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ is learnable.

The activation is applied after every propagation layer. For each task, we use a common activation choice

$$\phi_l = \phi, \qquad \phi \in \{\tanh, \mathrm{ReLU}\}, \qquad l = 0, \ldots, L-1.$$

Because $\boldsymbol{P}$ and $\boldsymbol{Q}$ are nonnegative row-stochastic matrices, their convex combination $\boldsymbol{M}_l$ is also nonnegative and row-stochastic. Furthermore, since $P_{uu} > 0$, $Q_{uv} > 0$ for all $u \neq v$, and $0 < \beta_l < 1$, every entry of $\boldsymbol{M}_l$ is strictly positive. Thus, each hybrid layer enables direct propagation between every pair of nodes while preserving the original local propagation channel.

Computational complexity. For each graph, exact construction of $\boldsymbol{Q}$ requires $O(n^3)$ preprocessing time and $O(n^2)$ memory and is performed only once. For constant hidden width d, an L-layer ER-GCN with h hybrid layers has forward complexity

$$O\left(Lnd^2 + L(n+m)d + hn^2 d\right),$$

introducing $O(hn^2 d)$ cost beyond a standard GCN. The proposed architecture has $h = L$ hybrid layers. Full layer-wise, memory, and mini-batch analyses are given in Appendix B.

6 EXPERIMENTS

We organize the experiments around three questions: (Q1) Does ER-GCN improve information transmission on controlled tasks with explicit long-range dependencies and structural bottlenecks? (Q2) How does ER-GCN perform on real-world prediction tasks with different expected demands for nonlocal communication, ranging from long-range graph benchmarks to conventional graph classification? (Q3) How do the global operator, hybrid-layer placement, mixing coefficient, and network depth affect performance?

We report results on controlled long-range tasks and five representative graph-level benchmarks. We also presents an ablation of hybrid-layer placement. Appendix E reports the remaining TU benchmarks and heterophilic node-classification results. It also provides ablations of the global operator, resistance kernel, mixing coefficient, and network depth.

6.1 EXPERIMENTAL SETUP

Detailed model configurations and hyperparameters are provided in Appendix D. Information including a link to the anonymized implementation is provided in Reproducibility Statement Section. Unless otherwise stated, results are reported as mean $\pm$ standard deviation under the repetition protocols described below.

We evaluate controlled long-range communication using TreeNeighborsMatch (Alon & Yahav, 2021) and our generated Barbell signal-transfer task (Appendix C). Barbell requires a signal to travel between two dense subgraphs through a narrow path. These tasks directly test information

propagation across long graph distances and structural bottlenecks. For graph-level evaluation, we use Peptides-func and Peptides-struct from LRGB (Dwivedi et al., 2022), ZINC (Irwin et al., 2012; Dwivedi et al., 2023), and the TU datasets (Morris et al., 2020) DD (Dobson & Doig, 2003; Shervashidze et al., 2011) and ENZYMES (Borgwardt et al., 2005; Schomburg et al., 2004). Additional results on TU datasets are reported in Appendix E.1. Appendix E.2 also presents node-classification results on the heterophilic graphs chameleon and squirrel (Rozemberczki et al., 2021; Pei et al., 2020), and actor (Pei et al., 2020). These experiments examine whether ER-GCN remains competitive when long-range communication is not the dominant challenge. We further study how the placement of hybrid layers affects performance on Barbell and TreeNeighborsMatch. Additional analyses in Appendix E compare alternative global operators and resistance kernels, fixed and adaptive mixing coefficients, and network depths.

The core baselines are the local message-passing models GCN (Kipf & Welling, 2017), GIN (Xu et al., 2019), and GAT (Veličković et al., 2018); the deep or decoupled models JKNet (Xu et al., 2018), APPNP (Gasteiger et al., 2019), and GCNII (Chen et al., 2020); the virtual-node models GCN+VN (Gilmer et al., 2017) and VN_G (Southern et al., 2025); the adaptive propagation model AMP (Errica et al., 2025); and the effective-resistance-based graph-rewiring method GTR (Black et al., 2023).

6.2 Results

Controlled long-range communication. Table 1 shows that ER-GCN achieves the lowest MAE at all Barbell bridge lengths, with little degradation as the bridge grows. On TreeNeighborsMatch, it maintains near-perfect accuracy through the maximum depth $d = 8$, matching the strongest graph-wide baselines while substantially outperforming most local and deep-propagation models. These findings are consistent with our support analysis, as dense hybrid propagation removes topology-induced gaps in computational-path support.

Table 1: Controlled long-range results (mean $\pm$ standard deviation over 10 seeds). Barbell is evaluated by test MAE and TreeNeighborsMatch by accuracy. Boldface denotes the best Barbell result at each bridge length and accuracy above 99% on TreeNeighborsMatch. N/A denotes local models whose receptive fields do not span the reported bridge lengths. † and ‡ indicate statistically significant paired differences from ER-GCN at $p < 0.05$ and $p < 0.01$, respectively.

	Barbell: test MAE (↓)			**TreeNeighborsMatch: accuracy (%, ↑)**		
Model	pl=10	pl=12	pl=14	d=3	d=6	d=8
GCN	N/A	N/A	N/A	$\mathbf{99.95 \pm 0.16}$	$12.35 \pm 3.19^{\ddagger}$	$14.15 \pm 3.06^{\ddagger}$
GIN	N/A	N/A	N/A	$\mathbf{100.00 \pm 0.00}$	$13.90 \pm 3.38^{\ddagger}$	$13.45 \pm 3.37^{\ddagger}$
GAT	N/A	N/A	N/A	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$	$41.20 \pm 36.08^{\ddagger}$
GCNII	N/A	N/A	N/A	$\mathbf{100.00 \pm 0.00}$	$13.55 \pm 5.20^{\ddagger}$	$12.05 \pm 2.39^{\ddagger}$
JKNet	N/A	N/A	N/A	$\mathbf{99.85 \pm 0.47}$	$45.20 \pm 37.39^{\ddagger}$	$14.85 \pm 2.49^{\ddagger}$
APPNP	N/A	N/A	N/A	$\mathbf{99.70 \pm 0.42}$	97.70 ± 4.60	$16.05 \pm 3.49^{\ddagger}$
AMP	N/A	N/A	N/A	$\mathbf{100.00 \pm 0.00}$	$30.90 \pm 24.16^{\ddagger}$	$18.85 \pm 15.34^{\ddagger}$
GCN+VN	$0.4678 \pm 0.11^{\ddagger}$	$0.4654 \pm 0.10^{\ddagger}$	$0.4698 \pm 0.097^{\ddagger}$	$\mathbf{99.80 \pm 0.42}$	$\mathbf{99.60 \pm 0.52}$	$\mathbf{99.80 \pm 0.35}$
VN_G	$0.8678 \pm 0.19^{\ddagger}$	$0.8121 \pm 0.21^{\ddagger}$	$0.8739 \pm 0.21^{\ddagger}$	$\mathbf{100.00 \pm 0.00}$	$\mathbf{99.85 \pm 0.34}$	$\mathbf{100.00 \pm 0.00}$
GTR	$0.2588 \pm 0.018^{\ddagger}$	$0.2886 \pm 0.017^{\ddagger}$	$0.3202 \pm 0.019^{\ddagger}$	$\mathbf{99.75 \pm 0.49}$	$12.30 \pm 3.33^{\ddagger}$	$14.85 \pm 2.49^{\ddagger}$
ER-GCN	$\mathbf{0.1213 \pm 0.017}$	$\mathbf{0.1233 \pm 0.016}$	$\mathbf{0.1229 \pm 0.017}$	$\mathbf{100.00 \pm 0.00}$	$\mathbf{99.70 \pm 0.67}$	$\mathbf{99.65 \pm 1.11}$

Graph-level prediction. As shown in Table 2, ER-GCN performs best on Peptides-func and DD and second best on Peptides-struct, showing that resistance-weighted propagation can benefit some graph-level prediction tasks. It remains competitive on ZINC but is less effective on ENZYMES, indicating that its gains are not universal. Notably, several models that outperform ER-GCN on individual datasets use substantially more trainable parameters, such as VN_G on Peptides-struct and AMP on ZINC, whereas ER-GCN has nearly the same trainable parameter count as GCN. Additional TU and heterophilic node-classification results are reported in Appendix E.

Ablations and additional analyses. Table 3 shows that a first-layer-only global channel performs well on Barbell but fails on deep TreeNeighborsMatch instances. In contrast, using the global channel only at the final propagation layer $L - 1$ is insufficient on Barbell. Applying hybrid propagation in every layer gives the most robust performance across both tasks and is therefore our default. Appendix E shows that dense global support accounts for most of the gains on long-range tasks, with

Table 2: Graph-level prediction performance (mean ± standard deviation). Peptides-func is evaluated using average precision (AP; ↑), Peptides-struct and ZINC using mean absolute error (MAE; ↓), and DD and ENZYMES using classification accuracy (%; ↑). Results are obtained over 10 seeds for Peptides and ZINC and 10-fold cross-validation for DD and ENZYMES. All models use the same number of propagation layers and hidden dimension. Trainable parameter counts are reported relative to ER-GCN. Boldface indicates the best result, and underlining indicates the three best results in each column. Statistically significant differences from ER-GCN under paired t-tests are denoted by † ($p < 0.05$) and ‡ ($p < 0.01$).

Model	#Params	Peptides		ZINC	DD	ENZYMES
	(× ER-GCN)	Func. AP ↑	Struct. MAE ↓	MAE ↓	Acc. ↑	Acc. ↑
GCN	1.00×	$0.4715 \pm 0.0022^{\ddagger}$	$0.3828 \pm 0.00071^{\ddagger}$	0.5192 ± 0.0082	$71.99 \pm 3.81^{\ddagger}$	24.83 ± 6.82
GIN	1.99×	$0.5083 \pm 0.0042^{\ddagger}$	$0.3730 \pm 0.0012^{\ddagger}$	$0.5303 \pm 0.010^{\ddagger}$	73.35 ± 4.00	$\underline{28.17 \pm 7.55}$
GAT	1.02×	$0.5077 \pm 0.0046^{\ddagger}$	$0.3681 \pm 0.0014^{\ddagger}$	$\underline{0.4919 \pm 0.0067}^{\ddagger}$	$71.23 \pm 3.42^{\dagger}$	26.83 ± 5.74
JKNet	1.06×	$0.4700 \pm 0.0017^{\ddagger}$	$0.3805 \pm 0.00070^{\ddagger}$	0.5226 ± 0.0070	72.92 ± 5.04	25.67 ± 6.72
APPNP	0.04×	$0.3642 \pm 0.00045^{\ddagger}$	$0.4700 \pm 0.00086^{\ddagger}$	$0.7388 \pm 0.00051^{\ddagger}$	$71.48 \pm 3.92^{\ddagger}$	$18.67 \pm 2.70^{\ddagger}$
GCNII	1.00×	$0.4533 \pm 0.0025^{\ddagger}$	$0.3911 \pm 0.00082^{\ddagger}$	$0.5748 \pm 0.0049^{\ddagger}$	$70.38 \pm 3.23^{\ddagger}$	23.67 ± 3.91
GCN+VN	1.96×	$\underline{0.6330 \pm 0.0054}^{\ddagger}$	$\underline{0.2910 \pm 0.0012}^{\ddagger}$	$0.5545 \pm 0.0058^{\ddagger}$	$\underline{74.03 \pm 4.24}$	$\underline{\mathbf{29.83 \pm 5.41}}^{\dagger}$
VN_G	1.96×	$\underline{0.6468 \pm 0.0043}^{\dagger}$	$\underline{\mathbf{0.2625 \pm 0.0014}}^{\ddagger}$	0.5207 ± 0.0047	$\underline{73.85 \pm 5.08}$	$\underline{\mathbf{29.83 \pm 3.72}}^{\dagger}$
AMP	2.99×	$0.5107 \pm 0.0046^{\ddagger}$	$0.3754 \pm 0.0035^{\ddagger}$	$\underline{\mathbf{0.4731 \pm 0.0042}}^{\ddagger}$	$70.80 \pm 4.00^{\dagger}$	24.17 ± 6.59
GTR	1.00×	$0.5009 \pm 0.0042^{\ddagger}$	$0.3224 \pm 0.00039^{\ddagger}$	$0.6113 \pm 0.0074^{\ddagger}$	$71.40 \pm 5.66^{\dagger}$	21.83 ± 3.28
ER-GCN	1.00×	$\underline{\mathbf{0.6542 \pm 0.0067}}$	$\underline{0.2664 \pm 0.00078}$	$\underline{0.5180 \pm 0.0048}$	$\underline{\mathbf{75.47 \pm 4.77}}$	25.50 ± 6.81

resistance-aware weighting providing further improvement. Adaptive mixing with learned layer-wise β_l coefficients also outperforms fixed-β variants. Notably, the learned β_l coefficients exhibit a clear and consistent depth-dependent pattern: shallow layers favor local propagation while deeper layers favor global propagation. ER-GCN remains competitive on short-range graph classification but provides no consistent benefit on heterophilic node classification, indicating that it primarily addresses long-range communication.

Table 3: Hybrid-layer placement ablation on Barbell and TreeNeighborsMatch with fixed $\beta = 0.1$. Barbell reports test MAE (↓), and TreeNeighborsMatch reports accuracy in % (↑). The placement $\{0, \ldots, L-1\}$ denotes the default ER-GCN placement used in the main experiments. Underlining denotes performance better than the default placement in the corresponding column.

	Barbell (MAE ↓)		TreeMatch (accuracy ↑)	
Position	pl=6	pl=10	$d = 6$	$d = 8$
None (GCN)	0.5914 ± 0.0131	0.6641 ± 0.0000	25.10 ± 27.95	12.50 ± 3.50
$\{0\}$	$\underline{0.0685 \pm 0.0117}$	$\underline{0.0786 \pm 0.0146}$	29.75 ± 26.83	15.30 ± 2.10
$\{L-1\}$	0.4817 ± 0.0161	0.5760 ± 0.0051	97.20 ± 5.88	63.35 ± 26.77
$\{0, L-1\}$	$\underline{0.1341 \pm 0.0080}$	$\underline{0.1100 \pm 0.0121}$	100.00 ± 0.00	58.35 ± 32.32
$\{1, \ldots, L-1\}$	0.1539 ± 0.0129	0.1886 ± 0.0149	99.95 ± 0.15	99.95 ± 0.15
$\{0, \ldots, L-1\}$ (default)	0.1343 ± 0.0132	0.1629 ± 0.0143	99.65 ± 0.63	99.90 ± 0.20

7 Conclusion

We introduced ER-GCN, which augments local graph convolution with a dense global propagation operator derived from effective resistance. Our analysis and proofs show that a single hybrid layer at any depth removes topology-induced structural zeros between distinct node pairs.

Experiments show that ER-GCN is most effective when long-range communication is important, while remaining competitive on predominantly local tasks. However, on graphs where long-range dependencies are not a primary challenge, ER-GCN provides less improvement over existing models. This suggests that dense global propagation may offer limited benefit when predictive signals are mainly local. Another limitation is the quadratic storage and propagation cost of the dense global operator. Future work could develop mechanisms that adapt the extent of global communication to task requirements, as well as scalable resistance approximations and sparse operators that retain the theoretical support benefits of ER-GCN for larger graphs.

AI USE STATEMENT

In this work, we used generative AI tools to generate synthetic data sets, formulate mathematical claims, assist in the writing of proofs (only improve the readability), help develop conceptual frameworks, provide feedback on research methodology or experiments, and implement methods. We did not use generative AI tools to provide critical ingredients for proving mathematical claims, clean and reformat data sets, support qualitative and thematic data analysis, or interpret results. The use of generative AI tools to assist with translation or to propose or refine hypotheses was not applicable to this work. Additionally, we used generative AI tools to edit software code and improve the readability of the research paper. We have reviewed all AI-assisted work. We checked LLM-generated research ideas for potential plagiarism through a manual literature survey, and LLM-generated code was verified and tested for correctness by two authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide model-construction details in Sections 4 and 5. The theoretical assumptions and complete proofs are provided in Appendix A, and the computational-complexity analysis is given in Appendix B. Dataset construction, preprocessing procedures, hyperparameters, and evaluation protocols are documented in Appendices C and D. An anonymized implementation is provided on https://anonymous.4open.science/r/ER-GCN .

REFERENCES

Uri Alon and Eran Yahav. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *International Conference on Learning Representations*, 2021.

Mitchell Black, Zhengchao Wan, Amir Nayyeri, and Yusu Wang. Understanding Oversquashing in GNNs through the Lens of Effective Resistance. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 2528–2547. PMLR, July 2023.

Mitchell Black, Zhengchao Wan, Gal Mishne, Amir Nayyeri, and Yusu Wang. Comparing Graph Transformers via Positional Encodings. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4103–4139. PMLR, July 2024.

Hugh Blayney, Alvaro Arroyo, Xiaowen Dong, and Michael Bronstein. gLSTM: Mitigating Over-Squashing by Increasing Storage Capacity. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 122463–122492, 2026.

Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alex J. Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21 (suppl_1):i47–i56, June 2005. ISSN 1367-4803. doi: 10.1093/bioinformatics/bti1007.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and Deep Graph Convolutional Networks. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1725–1735. PMLR, July 2020.

Asim Kumar Debnath, Rosa L. Lopez De Compadre, Gargi Debnath, Alan J. Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, 34(2):786–797, February 1991. ISSN 0022-2623, 1520-4804. doi: 10.1021/jm00106a046.

Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio, and Michael M. Bronstein. On Over-Squashing in Message Passing Neural Networks: The Impact of Width,

Depth, and Topology. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 7865–7885. PMLR, July 2023.

Paul D. Dobson and Andrew J. Doig. Distinguishing Enzyme Structures from Non-enzymes Without Alignments. *Journal of Molecular Biology*, 330(4):771–783, July 2003. ISSN 00222836. doi: 10.1016/S0022-2836(03)00628-4.

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 35*, pp. 22326–22340, New Orleans, Louisiana, USA, 2022. Neural Information Processing Systems Foundation, Inc. (NeurIPS). ISBN 978-1-7138-7108-8. doi: 10.52202/068431-1622.

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.

Federico Errica, Henrik Christiansen, Viktor Zaverkin, Takashi Maruyama, Mathias Niepert, and Francesco Alesiani. Adaptive Message Passing: A General Framework to Mitigate Oversmoothing, Oversquashing, and Underreaching. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 15490–15515. PMLR, July 2025.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Matthias Fey, Jinu Sunil, Akihiro Nitta, Rishi Puri, Manan Shah, Blaž Stojanovič, Ramona Bendias, Barghi Alexandria, Vid Kocijan, Zecheng Zhang, Xinwei He, Jan E. Lenssen, and Jure Leskovec. PyG 2.0: Scalable learning on real world graphs. In *Temporal Graph Learning Workshop @ KDD*, 2025.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *International Conference on Learning Representations*, 2019.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272. PMLR, August 2017.

Benjamin Gutteridge, Xiaowen Dong, Michael M. Bronstein, and Francesco Di Giovanni. DRew: Dynamically Rewired Message Passing with Delay. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 12252–12267. PMLR, July 2023.

John J. Irwin, Teague Sterling, Michael M. Mysinger, Erin S. Bolstad, and Ryan G. Coleman. ZINC: A Free Tool to Discover Chemistry for Biology. *Journal of Chemical Information and Modeling*, 52(7):1757–1768, July 2012. ISSN 1549-9596, 1549-960X. doi: 10.1021/ci3001277.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*, 2017.

D. J. Klein and M. Randić. Resistance distance. *Journal of Mathematical Chemistry*, 12(1):81–95, December 1993. ISSN 1572-8897. doi: 10.1007/BF01164627.

Nils M. Kriege and Petra Mutzel. Subgraph Matching Kernels for Attributed Graphs. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012.

Langzhang Liang, Fanchen Bu, Zixing Song, Zenglin Xu, Shirui Pan, and Kijung Shin. Mitigating Over-Squashing in Graph Neural Networks by Spectrum-Preserving Sparsification. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 37051–37070. PMLR, July 2025.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-GCN: Geometric Graph Convolutional Networks. In *International Conference on Learning Representations*, 2020.

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 35*, pp. 14501–14515, New Orleans, Louisiana, USA, 2022. Neural Information Processing Systems Foundation, Inc. (NeurIPS). ISBN 978-1-7138-7108-8. doi: 10.52202/068431-1054.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-Scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, April 2021. ISSN 2051-1329. doi: 10.1093/comnet/cnab014.

Ida Schomburg, Antje Chang, Christian Ebeling, Marion Gremse, Christian Heldt, Gregor Huhn, and Dietmar Schomburg. BRENDA, the enzyme database: updates and major new developments. *Nucleic Acids Research*, 32(suppl_1):D431–D433, January 2004. ISSN 0305-1048. doi: 10.1093/nar/gkh081.

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(77): 2539–2561, 2011.

Joshua Southern, Francesco Di Giovanni, Michael Bronstein, and Johannes Lutzeyer. Understanding Virtual Nodes: Oversquashing and Node Heterogeneity. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 27512–27540, 2025.

Daniel A. Spielman and Nikhil Srivastava. Graph Sparsification by Effective Resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011. doi: 10.1137/080734029.

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.

Ameya Velingker, Ali Sinop, Ira Ktena, Petar Veličković, and Sreenivas Gollapudi. Affinity-Aware Graph Networks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 67847–67865. Curran Associates, Inc., 2023. doi: 10.52202/075280-2967.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *International Conference on Learning Representations*, 2018.

Nikil Wale and George Karypis. Comparison of Descriptor Spaces for Chemical Compound Retrieval and Classification. In *Sixth International Conference on Data Mining (ICDM'06)*, pp. 678–689, 2006. doi: 10.1109/ICDM.2006.39.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation Learning on Graphs with Jumping Knowledge Networks. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5453–5462. PMLR, July 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*, 2019.

Pinar Yanardag and S.V.N. Vishwanathan. Deep Graph Kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pp. 1365–1374, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 978-1-4503-3664-2. doi: 10.1145/2783258.2783417.

A Additional Theoretical Analysis

A.1 Node-Wise Jacobian and Computational-Path Expansion

Proof of Proposition 3.1. For each node j, transposing the j-th row of

$$\boldsymbol{Z}^{(l+1)} = \boldsymbol{S}_l \boldsymbol{H}^{(l)} \boldsymbol{W}^{(l)}$$

gives

$$\boldsymbol{z}_j^{(l+1)} = \boldsymbol{W}^{(l)\top} \sum_{i \in V} (S_l)_{ji} \boldsymbol{h}_i^{(l)}.$$

Differentiating with respect to the input features $\boldsymbol{x}_u = \boldsymbol{h}_u^{(0)}$ and applying the chain rule yields

$$\begin{aligned} \mathcal{J}_{j \leftarrow u}^{(l+1)} &= \frac{\partial \boldsymbol{h}_j^{(l+1)}}{\partial \boldsymbol{z}_j^{(l+1)}} \frac{\partial \boldsymbol{z}_j^{(l+1)}}{\partial \boldsymbol{x}_u} \\ &= \boldsymbol{G}_j^{(l+1)} \boldsymbol{W}^{(l)\top} \sum_{i \in V} (S_l)_{ji} \mathcal{J}_{i \leftarrow u}^{(l)}. \end{aligned} \tag{17}$$

The initial Jacobian blocks are

$$\mathcal{J}_{i \leftarrow u}^{(0)} = \frac{\partial \boldsymbol{x}_i}{\partial \boldsymbol{x}_u} = \delta_{iu} \boldsymbol{I}_{d_0}.$$

We now repeatedly substitute (17). After the first layer,

$$\mathcal{J}_{i_1 \leftarrow u}^{(1)} = (S_0)_{i_1 u} \boldsymbol{G}_{i_1}^{(1)} \boldsymbol{W}^{(0)\top}.$$

After two layers,

$$\mathcal{J}_{i_2 \leftarrow u}^{(2)} = \sum_{i_1 \in V} (S_1)_{i_2 i_1} (S_0)_{i_1 u} \boldsymbol{G}_{i_2}^{(2)} \boldsymbol{W}^{(1)\top} \boldsymbol{G}_{i_1}^{(1)} \boldsymbol{W}^{(0)\top}.$$

Continuing recursively through all L layers gives

$$\mathcal{J}_{v \leftarrow u}^{(L)} = \sum_{\substack{i_0 = u,\ i_L = v \\ i_1, \ldots, i_{L-1} \in V}} \left(\prod_{l=0}^{L-1} (S_l)_{i_{l+1} i_l} \right) \boldsymbol{G}_{i_L}^{(L)} \boldsymbol{W}^{(L-1)\top} \cdots \boldsymbol{G}_{i_1}^{(1)} \boldsymbol{W}^{(0)\top}, \tag{18}$$

which proves the computational-path expansion (4).

Next, by the definition

$$\boldsymbol{T}_L^{(\mathcal{K})} = \boldsymbol{S}_{L-1}^{(\mathcal{K})} \cdots \boldsymbol{S}_0^{(\mathcal{K})},$$

standard matrix multiplication gives

$$\left(\boldsymbol{T}_L^{(\mathcal{K})}\right)_{vu} = \sum_{\substack{i_0=u,\, i_L=v \\ i_1,\ldots,i_{L-1}\in V}} \prod_{l=0}^{L-1} \left(S_l^{(\mathcal{K})}\right)_{i_{l+1}i_l}.$$

Every scalar propagation factor in this expression is nonnegative. Therefore, if

$$\left(\boldsymbol{T}_L^{(\mathcal{K})}\right)_{vu} = 0,$$

then every path coefficient must vanish:

$$\prod_{l=0}^{L-1} \left(S_l^{(\mathcal{K})}\right)_{i_{l+1}i_l} = 0$$

for every node sequence satisfying $i_0 = u$ and $i_L = v$. Every term in the Jacobian expansion (4) is therefore zero, independently of the values or signs of the feature weight matrices and activation Jacobians. Hence,

$$\mathcal{J}_{v\leftarrow u}^{(L)} = \mathbf{0}.$$

This proves (5) and the corresponding support inclusion. □

A.1.1 Purely Local Propagation as a Special Case

When $\mathcal{K} = \varnothing$, every layer uses the local propagation operator $\boldsymbol{P}$, so

$$\boldsymbol{T}_L^{(\varnothing)} = \boldsymbol{P}^L.$$

By matrix multiplication,

$$(\boldsymbol{P}^L)_{vu} = \sum_{\substack{i_0=u,\, i_L=v \\ i_1,\ldots,i_{L-1}\in V}} \prod_{l=0}^{L-1} P_{i_{l+1}i_l}.$$

Because $P_{ji} > 0$ only if $j = i$ or $\{i, j\} \in E$, every nonzero summand corresponds to a length-L sequence whose steps either traverse an edge or remain at the current node. Removing the self-loop steps produces a walk from u to v containing at most L graph edges. Therefore,

$$d_G(u, v) > L \quad \Longrightarrow \quad (\boldsymbol{P}^L)_{vu} = 0.$$

Applying Proposition 3.1 then gives

$$\mathcal{J}_{v\leftarrow u}^{(L)} = \mathbf{0},$$

and hence

$$s_{u\to v}^{(L)} = \left\| \mathcal{J}_{v\leftarrow u}^{(L)} \right\|_F = 0,$$

which establishes (3).

A.1.2 Single-Hybrid-Layer Factorization

For completeness, suppose that $\mathcal{K} = \{k\}$. There are k local propagation factors before the hybrid layer and $L - 1 - k$ local factors after it. Therefore,

$$\boldsymbol{T}_L^{(k)} = \boldsymbol{P}^{L-1-k} \boldsymbol{M} \boldsymbol{P}^k.$$

Substituting

$$\boldsymbol{M} = (1-\beta)\boldsymbol{P} + \beta \boldsymbol{Q}$$

gives

$$\begin{aligned} \boldsymbol{T}_L^{(k)} &= \boldsymbol{P}^{L-1-k}\big((1-\beta)\boldsymbol{P} + \beta\boldsymbol{Q}\big)\boldsymbol{P}^k \\ &= (1-\beta)\boldsymbol{P}^L + \beta \boldsymbol{P}^{L-1-k}\boldsymbol{Q}\boldsymbol{P}^k, \end{aligned} \tag{19}$$

which establishes (6).

A.2 Proof of global reach from a single hybrid layer

Proof. By (6),

$$\boldsymbol{T}_L^{(k)} = (1-\beta)\boldsymbol{P}^L + \beta\boldsymbol{P}^{L-1-k}\boldsymbol{Q}\boldsymbol{P}^k.$$

All matrices in this expression are nonnegative. Therefore, for $u \neq v$,

$$\begin{aligned}\left(\boldsymbol{T}_L^{(k)}\right)_{vu} &\geq \beta\left(\boldsymbol{P}^{L-1-k}\boldsymbol{Q}\boldsymbol{P}^k\right)_{vu} \\ &\geq \beta\left(\boldsymbol{P}^{L-1-k}\right)_{vv} Q_{vu}\left(\boldsymbol{P}^k\right)_{uu} \\ &\geq \beta(P_{vv})^{L-1-k}Q_{vu}(P_{uu})^k. \end{aligned} \tag{20}$$

The last expression is strictly positive because

$$P_{uu} > 0, \qquad P_{vv} > 0, \qquad Q_{vu} > 0 \quad \text{for } u \neq v \text{ and } 0 < \beta \leq 1.$$

The convention that a scalar raised to the zeroth power equals one covers the endpoint placements $k = 0$ and $k = L-1$.

If $0 < \beta < 1$, then the diagonal entries are also positive since

$$\left(\boldsymbol{T}_L^{(k)}\right)_{uu} \geq (1-\beta)(\boldsymbol{P}^L)_{uu} \geq (1-\beta)(P_{uu})^L > 0.$$

Therefore, $\boldsymbol{T}_L^{(k)} > 0$ entrywise. □

A.3 Proof of the Multiple-Hybrid-Layer Corollary

Proof of Corollary 3.3. Expanding each hybrid factor as

$$\boldsymbol{M} = (1-\beta)\boldsymbol{P} + \beta\boldsymbol{Q},$$

the product $\boldsymbol{T}_L^{(\mathcal{K})}$ is a sum of nonnegative matrix products. Fix $k \in \mathcal{K}$ and select the term containing $\beta\boldsymbol{Q}$ at layer k and $(1-\beta)\boldsymbol{P}$ at every other hybrid layer. All non-hybrid layers already contribute $\boldsymbol{P}$. Therefore, this selected term gives the entrywise bound

$$\boldsymbol{T}_L^{(\mathcal{K})} \succeq \beta(1-\beta)^{|\mathcal{K}|-1}\boldsymbol{P}^{L-1-k}\boldsymbol{Q}\boldsymbol{P}^k,$$

where $\succeq$ denotes entrywise inequality.

For $u \neq v$, consider the contribution that remains at u for the first k local propagation steps, moves directly from u to v through $\boldsymbol{Q}$, and then remains at v for the remaining $L-1-k$ steps. It follows that

$$\begin{aligned}\left(\boldsymbol{T}_L^{(\mathcal{K})}\right)_{vu} &\geq \beta(1-\beta)^{|\mathcal{K}|-1}\left(\boldsymbol{P}^{L-1-k}\boldsymbol{Q}\boldsymbol{P}^k\right)_{vu} \\ &\geq \beta(1-\beta)^{|\mathcal{K}|-1}(P_{vv})^{L-1-k}Q_{vu}(P_{uu})^k > 0. \end{aligned} \tag{21}$$

This proves (7).

It remains to establish positivity of the diagonal entries. Every local factor has

$$P_{uu} > 0.$$

At a hybrid layer $l \in \mathcal{K}$,

$$\left(S_l^{(\mathcal{K})}\right)_{uu} = (1-\beta)P_{uu} + \beta Q_{uu} = (1-\beta)P_{uu} > 0,$$

because $Q_{uu} = 0$ and $0 < \beta < 1$. Hence, every propagation factor has a strictly positive diagonal, and therefore

$$\left(\boldsymbol{T}_L^{(\mathcal{K})}\right)_{uu} \geq \prod_{l=0}^{L-1}\left(S_l^{(\mathcal{K})}\right)_{uu} > 0.$$

Together with the positive off-diagonal entries established above, this gives

$$\boldsymbol{T}_L^{(\mathcal{K})} > 0$$

entrywise. □

A.4 Nonlinear Jacobian Support for Tanh Networks

Proof of Theorem 3.4. We specialize to networks with $\tanh$ activations in every propagation layer:

$$\phi_l = \tanh, \qquad l = 0, \dots, L-1.$$

For every node i and layer $r = 1, \dots, L$, the local feature Jacobian is

$$\boldsymbol{G}_i^{(r)} = \operatorname{diag}\Big(1 - \tanh^2\Big(\boldsymbol{z}_i^{(r)}\Big)\Big),$$

whose diagonal entries are strictly positive for finite preactivations.

The implication

$$\Big(\boldsymbol{T}_L^{(\mathcal{K})}\Big)_{vu} = 0 \quad \Longrightarrow \quad \mathcal{J}_{v\leftarrow u}^{(L)} = \mathbf{0}$$

already follows from Proposition 3.1. It therefore remains to establish the reverse implication under the stated nonnegativity conditions.

Suppose that all weight matrices are entrywise nonnegative and

$$\boldsymbol{B}_L := \boldsymbol{W}^{(0)} \cdots \boldsymbol{W}^{(L-1)} \neq \mathbf{0}.$$

Every term in the computational-path expansion (4) is then entrywise nonnegative, so exact cancellation cannot occur.

More explicitly, if

$$\Big(\boldsymbol{T}_L^{(\mathcal{K})}\Big)_{vu} > 0,$$

then, because all propagation coefficients are nonnegative, there exists a node sequence

$$i_0 = u,\ i_1, \dots, i_{L-1},\ i_L = v$$

such that

$$\prod_{l=0}^{L-1} (S_l)_{i_{l+1} i_l} > 0.$$

Furthermore, since $\boldsymbol{B}_L$ is nonzero and entrywise nonnegative, there exist feature coordinates $a_0, \dots, a_L$ such that

$$\prod_{l=0}^{L-1} W_{a_l a_{l+1}}^{(l)} > 0.$$

Indeed, this follows from the feature-path expansion

$$(\boldsymbol{B}_L)_{a_0 a_L} = \sum_{a_1, \dots, a_{L-1}} \prod_{l=0}^{L-1} W_{a_l a_{l+1}}^{(l)}.$$

The joint node–feature path specified by

$$(i_0, a_0), \dots, (i_L, a_L)$$

therefore contributes to the (a_L, a_0)-th entry of $\mathcal{J}_{v\leftarrow u}^{(L)}$ the strictly positive quantity

$$\left(\prod_{l=0}^{L-1} (S_l)_{i_{l+1} i_l}\right) \left(\prod_{r=1}^{L} (\boldsymbol{G}_{i_r}^{(r)})_{a_r a_r}\right) \left(\prod_{l=0}^{L-1} W_{a_l a_{l+1}}^{(l)}\right).$$

All remaining path contributions are nonnegative. Consequently,

$$\left\| \mathcal{J}_{v\leftarrow u}^{(L)} \right\|_F > 0.$$

Combining the two directions gives

$$\left\| \mathcal{J}_{v\leftarrow u}^{(L)} \right\|_F > 0 \quad \Longleftrightarrow \quad \Big(\boldsymbol{T}_L^{(\mathcal{K})}\Big)_{vu} > 0.$$

We next remove the entrywise nonnegativity assumption on the weight matrices. Contributions from different computational paths may then cancel exactly. However, such cancellations occur only for an exceptional set of parameters.

Let $\boldsymbol{\theta}_W$ denote the vector containing all entries of the weight matrices. For any fixed finite input, each entry of the Jacobian block

$$\mathcal{J}_{v\leftarrow u}^{(L)}(\boldsymbol{\theta}_W)$$

is a real-analytic function of $\boldsymbol{\theta}_W$, since $\tanh$ is real analytic and the network is a finite composition of analytic operations.

Whenever

$$\left(\boldsymbol{T}_L^{(\mathcal{K})}\right)_{vu} > 0,$$

an entrywise positive choice of the weight matrices yields a nonzero Jacobian block, as established above. Hence, at least one entry

$$f_{uv}(\boldsymbol{\theta}_W)$$

of $\mathcal{J}_{v\leftarrow u}^{(L)}(\boldsymbol{\theta}_W)$ is not identically zero. The zero set of a nontrivial real-analytic function has Lebesgue measure zero. Therefore,

$$\left\{\boldsymbol{\theta}_W : \mathcal{J}_{v\leftarrow u}^{(L)}(\boldsymbol{\theta}_W) = \mathbf{0}\right\} \subseteq \{\boldsymbol{\theta}_W : f_{uv}(\boldsymbol{\theta}_W) = 0\}$$

also has Lebesgue measure zero. Consequently,

$$\mathcal{J}_{v\leftarrow u}^{(L)}(\boldsymbol{\theta}_W) \neq \mathbf{0}$$

for almost every choice of the weight parameters whenever

$$\left(\boldsymbol{T}_L^{(\mathcal{K})}\right)_{vu} > 0.$$

In particular, if a single hybrid layer is inserted at any $k \in \{0, \dots, L-1\}$ and $0 < \beta \leq 1$, then Theorem 3.2 gives

$$\left(\boldsymbol{T}_L^{(k)}\right)_{vu} > 0 \qquad \text{for every } u \neq v.$$

Under the entrywise nonnegative-weight condition and $\boldsymbol{B}_L \neq \mathbf{0}$, it follows that

$$\left\|\mathcal{J}_{v\leftarrow u}^{(L)}\right\|_F > 0 \qquad \text{for every } u \neq v.$$

For unconstrained signed weights, each ordered pair $u \neq v$ has only a measure-zero exceptional parameter set. Because the graph has finitely many ordered node pairs, the union of these exceptional sets also has measure zero. Thus, outside a single measure-zero set of weight parameters,

$$\mathcal{J}_{v\leftarrow u}^{(L)} \neq \mathbf{0} \qquad \text{simultaneously for every } u \neq v.$$

Therefore, a single hybrid layer at any depth removes the hard finite-receptive-field barrier at the level of the realized $\tanh$ Jacobian under the nonnegative-weight assumptions, and generically under unconstrained signed weights. □

A.5 Expected Sensitivity under Randomized ReLU Gates

We consider networks with ReLU activations in every propagation layer:

$$\phi_l = \mathrm{ReLU}, \qquad l = 0, \dots, L-1.$$

As before, define

$$\boldsymbol{B}_L := \boldsymbol{W}^{(0)} \cdots \boldsymbol{W}^{(L-1)}.$$

At points where no layer preactivation is zero, the local feature Jacobian is well defined and satisfies

$$\boldsymbol{G}_i^{(r)} = \mathrm{diag}\left(\mathbf{1}\left\{\boldsymbol{z}_i^{(r)} > 0\right\}\right), \qquad r = 1, \dots, L,$$

where the indicator is applied coordinatewise. Thus, each diagonal entry of $\boldsymbol{G}_i^{(r)}$ is either one or zero according to whether the corresponding ReLU unit is active or inactive.

Unlike the $\tanh$ case, positive node-level propagation support does not guarantee a nonzero realized Jacobian: inactive ReLU gates may block all associated node–feature paths, and signed path contributions may cancel.

To separate this gate-dependent effect from the graph-topological obstruction, we adopt the path-level randomized activation model used in prior GNN influence analyses (Xu et al., 2018; Di Giovanni et al., 2023).

A complete node–feature computational path q specifies a node sequence

$$i_0 = u,\ i_1, \dots, i_{L-1},\ i_L = v$$

and a feature-coordinate sequence

$$a_0, \dots, a_L$$

through the L layers. Let

$$\Xi_q := \prod_{r=1}^{L} \mathbf{1}\left\{ Z^{(r)}_{i_r, a_r} > 0 \right\} \in \{0, 1\}$$

denote the product of the layer-wise ReLU derivative gates along this path. The empty product is understood to equal one.

For a fixed input and conditional on the weight matrices, assume that

$$\mathbb{E}_{\text{gate}}\left[\Xi_q \,\middle|\, \{\boldsymbol{W}^{(l)}\}_{l=0}^{L-1} \right] = \rho > 0 \tag{22}$$

for every complete node–feature computational path q. No independence among the variables $\{\Xi_q\}_q$ is required for the following first-moment calculation.

Proposition A.1 (Expected sensitivity under randomized ReLU gates)**.** *Under the randomized-gate model above,*

$$\mathbb{E}_{\text{gate}}\left[\mathcal{J}^{(L)}_{v \leftarrow u} \,\middle|\, \{\boldsymbol{W}^{(l)}\}_{l=0}^{L-1} \right] = \rho \left(\boldsymbol{T}_L^{(\mathcal{K})} \right)_{vu} \boldsymbol{B}_L^{\top}.$$

Consequently,

$$\mathbb{E}_{\text{gate}}\left[\left\| \mathcal{J}^{(L)}_{v \leftarrow u} \right\|_F \,\middle|\, \{\boldsymbol{W}^{(l)}\}_{l=0}^{L-1} \right] \geq \rho \left(\boldsymbol{T}_L^{(\mathcal{K})} \right)_{vu} \|\boldsymbol{B}_L\|_F. \tag{23}$$

Proof of Proposition A.1. Expanding both the node and feature coordinates in (4), the (a_L, a_0)-th entry of the node-to-node Jacobian can be written as

$$\begin{aligned} \left(\mathcal{J}^{(L)}_{v \leftarrow u} \right)_{a_L a_0} = \sum_{\substack{i_0 = u,\ i_L = v \\ i_1, \dots, i_{L-1} \in V}} \sum_{a_1, \dots, a_{L-1}} \left(\prod_{l=0}^{L-1} (S_l)_{i_{l+1} i_l} \right) \\ \times \left(\prod_{l=0}^{L-1} W^{(l)}_{a_l a_{l+1}} \right) \Xi_q. \end{aligned} \tag{24}$$

Taking the conditional expectation and using (22) gives

$$\begin{aligned} &\mathbb{E}_{\text{gate}}\left[\left(\mathcal{J}^{(L)}_{v \leftarrow u} \right)_{a_L a_0} \,\middle|\, \{\boldsymbol{W}^{(l)}\}_{l=0}^{L-1} \right] \\ &\quad = \rho \sum_{\substack{i_0 = u,\ i_L = v \\ i_1, \dots, i_{L-1} \in V}} \prod_{l=0}^{L-1} (S_l)_{i_{l+1} i_l} \sum_{a_1, \dots, a_{L-1}} \prod_{l=0}^{L-1} W^{(l)}_{a_l a_{l+1}}. \end{aligned} \tag{25}$$

Linearity of expectation is sufficient here; no independence among the path-gate variables is needed.

The node-path sum is

$$\sum_{\substack{i_0 = u,\ i_L = v \\ i_1, \dots, i_{L-1} \in V}} \prod_{l=0}^{L-1} (S_l)_{i_{l+1} i_l} = \left(\boldsymbol{T}_L^{(\mathcal{K})} \right)_{vu}.$$

Similarly, using

$$\boldsymbol{B}_L = \boldsymbol{W}^{(0)} \cdots \boldsymbol{W}^{(L-1)},$$

the feature-path sum satisfies

$$\sum_{a_1,\ldots,a_{L-1}} \prod_{l=0}^{L-1} W^{(l)}_{a_l a_{l+1}} = (\boldsymbol{B}_L)_{a_0 a_L}.$$

Therefore,

$$\mathbb{E}_{\text{gate}}\left[\left(\mathcal{J}^{(L)}_{v\leftarrow u}\right)_{a_L a_0} \,\middle|\, \{\boldsymbol{W}^{(l)}\}_{l=0}^{L-1}\right] = \rho\left(\boldsymbol{T}_L^{(\mathcal{K})}\right)_{vu} (\boldsymbol{B}_L)_{a_0 a_L}.$$

Collecting all feature-coordinate entries yields

$$\mathbb{E}_{\text{gate}}\left[\mathcal{J}^{(L)}_{v\leftarrow u} \,\middle|\, \{\boldsymbol{W}^{(l)}\}_{l=0}^{L-1}\right] = \rho\left(\boldsymbol{T}_L^{(\mathcal{K})}\right)_{vu} \boldsymbol{B}_L^{\top}.$$

Finally, Jensen's inequality for the convex function $\boldsymbol{A} \mapsto \|\boldsymbol{A}\|_F$ gives

$$\begin{aligned}
&\mathbb{E}_{\text{gate}}\left[\left\|\mathcal{J}^{(L)}_{v\leftarrow u}\right\|_F \,\middle|\, \{\boldsymbol{W}^{(l)}\}_{l=0}^{L-1}\right] \\
&\quad \geq \left\|\mathbb{E}_{\text{gate}}\left[\mathcal{J}^{(L)}_{v\leftarrow u} \,\middle|\, \{\boldsymbol{W}^{(l)}\}_{l=0}^{L-1}\right]\right\|_F \\
&\quad = \rho\left(\boldsymbol{T}_L^{(\mathcal{K})}\right)_{vu} \|\boldsymbol{B}_L\|_F, \qquad (26)
\end{aligned}$$

where we used the nonnegativity of $\left(\boldsymbol{T}_L^{(\mathcal{K})}\right)_{vu}$. This proves (23). □

Single hybrid layer. Suppose that a single hybrid layer is inserted at an arbitrary depth $k \in \{0, \ldots, L-1\}$ and that $0 < \beta \leq 1$. Theorem 3.2 gives, for every $u \neq v$,

$$\left(\boldsymbol{T}_L^{(k)}\right)_{vu} \geq \beta (P_{vv})^{L-1-k} Q_{vu} (P_{uu})^k > 0.$$

Combining this result with (23) yields

$$\begin{aligned}
&\mathbb{E}_{\text{gate}}\left[\left\|\mathcal{J}^{(L)}_{v\leftarrow u}\right\|_F \,\middle|\, \{\boldsymbol{W}^{(l)}\}_{l=0}^{L-1}\right] \\
&\quad \geq \rho\beta (P_{vv})^{L-1-k} Q_{vu} (P_{uu})^k \|\boldsymbol{B}_L\|_F. \qquad (27)
\end{aligned}$$

Hence, whenever $\boldsymbol{B}_L \neq \boldsymbol{0}$,

$$\mathbb{E}_{\text{gate}}\left[\left\|\mathcal{J}^{(L)}_{v\leftarrow u}\right\|_F \,\middle|\, \{\boldsymbol{W}^{(l)}\}_{l=0}^{L-1}\right] > 0 \qquad \text{for every } u \neq v.$$

Since the insertion depth k was arbitrary, a single resistance-weighted global layer removes all graph-induced zero sensitivities at the level of the expected Jacobian norm, independently of its placement.

Multiple hybrid layers. The same argument extends to multiple hybrid layers. If $\mathcal{K} \neq \varnothing$ and $0 < \beta < 1$, then Corollary 3.3 implies that, for every $k \in \mathcal{K}$ and $u \neq v$,

$$\left(\boldsymbol{T}_L^{(\mathcal{K})}\right)_{vu} \geq \beta (1-\beta)^{|\mathcal{K}|-1} (P_{vv})^{L-1-k} Q_{vu} (P_{uu})^k.$$

Therefore,

$$\begin{aligned}
&\mathbb{E}_{\text{gate}}\left[\left\|\mathcal{J}^{(L)}_{v\leftarrow u}\right\|_F \,\middle|\, \{\boldsymbol{W}^{(l)}\}_{l=0}^{L-1}\right] \\
&\quad \geq \rho\beta (1-\beta)^{|\mathcal{K}|-1} (P_{vv})^{L-1-k} Q_{vu} (P_{uu})^k \|\boldsymbol{B}_L\|_F. \qquad (28)
\end{aligned}$$

In particular, this expected sensitivity is strictly positive for every pair $u \neq v$ whenever $\boldsymbol{B}_L \neq \boldsymbol{0}$.

Nondegeneracy under continuous weight distributions. If the joint distribution of all weight entries is absolutely continuous with respect to Lebesgue measure, then

$$\boldsymbol{B}_L \neq \mathbf{0}$$

almost surely. To see this, choose arbitrary feature coordinates $a_0, \ldots, a_L$ and consider the polynomial

$$p(\boldsymbol{\theta}_W) := (\boldsymbol{B}_L)_{a_0 a_L}$$

in the entries $\boldsymbol{\theta}_W$ of the weight matrices. This polynomial is not identically zero: for example, setting

$$W^{(l)}_{a_l a_{l+1}} = 1, \qquad l = 0, \ldots, L-1,$$

and setting all other weight entries to zero gives

$$p(\boldsymbol{\theta}_W) = 1.$$

The zero set of a nontrivial polynomial has Lebesgue measure zero. Since

$$\{\boldsymbol{\theta}_W : \boldsymbol{B}_L = \mathbf{0}\} \subseteq \{\boldsymbol{\theta}_W : p(\boldsymbol{\theta}_W) = 0\},$$

the event $\boldsymbol{B}_L = \mathbf{0}$ has probability zero under any joint weight distribution that is absolutely continuous with respect to Lebesgue measure.

Expected versus realized sensitivity. The above result concerns expected sensitivity under the randomized path-gate model. It does not imply that the realized Jacobian of every finite-width ReLU network is nonzero. Even when

$$\left(\boldsymbol{T}_L^{(\mathcal{K})}\right)_{vu} > 0,$$

inactive ReLU gates may simultaneously block all relevant node–feature paths, and signed contributions from active paths may cancel exactly. The hybrid operator removes the graph-topological absence of candidate paths, while the randomized-gate result guarantees that their expected sensitivity is positive under the stated assumptions.

A.6 Variational Characterization of the Exponential Resistance Kernel

We provide the aggregation-distortion bound and the variational derivation underlying the exponential resistance kernel.

Fix a target node u, and define the probability simplex

$$\Delta_u = \left\{ \boldsymbol{q}_u = (q_{uv})_{v \neq u} : q_{uv} \geq 0,\ \sum_{v \neq u} q_{uv} = 1 \right\}.$$

For $\boldsymbol{q}_u \in \Delta_u$, define the corresponding global aggregate by

$$\overline{\boldsymbol{h}}_u(\boldsymbol{q}_u) = \sum_{v \neq u} q_{uv} \boldsymbol{h}_v.$$

Lemma A.2 (Resistance-controlled aggregation distortion)**.** *For every $\boldsymbol{q}_u \in \Delta_u$,*

$$\left\| \overline{\boldsymbol{h}}_u(\boldsymbol{q}_u) - \boldsymbol{h}_u \right\|_2^2 \leq \operatorname{tr}(\boldsymbol{H}^\top \boldsymbol{L} \boldsymbol{H}) \sum_{v \neq u} q_{uv} R_{\text{eff}}(u, v). \tag{29}$$

Proof. Let

$$\boldsymbol{b}_{uv} := \boldsymbol{e}_u - \boldsymbol{e}_v.$$

For an arbitrary scalar graph signal $\boldsymbol{y} \in \mathbb{R}^n$, the vectors $\boldsymbol{b}_{uv}$ and $\boldsymbol{L}^{1/2}\boldsymbol{y}$ lie in the subspace orthogonal to the constant vector. Therefore,

$$y_u - y_v = \boldsymbol{b}_{uv}^\top \boldsymbol{y} = \left(\boldsymbol{L}^{+1/2} \boldsymbol{b}_{uv}\right)^\top \left(\boldsymbol{L}^{1/2} \boldsymbol{y}\right).$$

By the Cauchy–Schwarz inequality,

$$\begin{aligned}(y_u - y_v)^2 &\le \left\|\boldsymbol{L}^{+1/2}\boldsymbol{b}_{uv}\right\|_2^2 \left\|\boldsymbol{L}^{1/2}\boldsymbol{y}\right\|_2^2 \\ &= \left(\boldsymbol{b}_{uv}^\top \boldsymbol{L}^+ \boldsymbol{b}_{uv}\right)\left(\boldsymbol{y}^\top \boldsymbol{L}\boldsymbol{y}\right) \\ &= R_{\text{eff}}(u,v)\,\boldsymbol{y}^\top \boldsymbol{L}\boldsymbol{y}. \end{aligned} \tag{30}$$

Apply (30) to each feature column of $\boldsymbol{H}$ and sum over the feature coordinates. This gives

$$\|\boldsymbol{h}_u - \boldsymbol{h}_v\|_2^2 \le R_{\text{eff}}(u,v)\ \operatorname{tr}(\boldsymbol{H}^\top \boldsymbol{L}\boldsymbol{H}). \tag{31}$$

Because $\boldsymbol{q}_u \in \Delta_u$,

$$\overline{\boldsymbol{h}}_u(\boldsymbol{q}_u) - \boldsymbol{h}_u = \sum_{v\neq u} q_{uv}(\boldsymbol{h}_v - \boldsymbol{h}_u).$$

Convexity of the squared Euclidean norm and (31) imply

$$\begin{aligned}\left\|\overline{\boldsymbol{h}}_u(\boldsymbol{q}_u) - \boldsymbol{h}_u\right\|_2^2 &\le \sum_{v\neq u} q_{uv}\|\boldsymbol{h}_v - \boldsymbol{h}_u\|_2^2 \\ &\le \operatorname{tr}(\boldsymbol{H}^\top \boldsymbol{L}\boldsymbol{H}) \sum_{v\neq u} q_{uv} R_{\text{eff}}(u,v). \end{aligned} \tag{32}$$

□

Lemma A.2 shows that the weighted average resistance controls an upper bound on the representation distortion introduced by global aggregation. Minimizing this bound without regularization would generally concentrate all probability mass on the minimum-resistance nodes. To obtain a diffuse, full-support distribution, let

$$\pi_{uv} = \frac{1}{n-1}, \qquad v \neq u,$$

and consider the entropy-regularized problem

$$\min_{\boldsymbol{q}_u \in \Delta_u} \left\{ \sum_{v\neq u} q_{uv} R_{\text{eff}}(u,v) + \sigma\, \mathrm{KL}\left(\boldsymbol{q}_u \| \boldsymbol{\pi}_u\right) \right\}. \tag{13}$$

Proposition A.3 (Entropy-regularized resistance weighting)**.** *For every $\sigma > 0$, problem* (13) *has the unique solution*

$$q^\star_{uv} = \frac{\exp(-R_{\text{eff}}(u,v)/\sigma)}{\sum_{w\neq u} \exp(-R_{\text{eff}}(u,w)/\sigma)}, \qquad v \neq u. \tag{33}$$

Consequently,

$$q^\star_{uv} = Q_{uv}.$$

Proof. For brevity, write

$$R_{uv} := R_{\text{eff}}(u,v)$$

and define the partition function

$$Z_u := \sum_{w\neq u} \pi_{uw} \exp\left(-\frac{R_{uw}}{\sigma}\right).$$

Consider the distribution

$$q^\star_{uv} := \frac{\pi_{uv}\exp(-R_{uv}/\sigma)}{Z_u}, \qquad v \neq u.$$

Because $\pi_{uv} > 0$, $R_{uv} < \infty$, and $\sigma > 0$, this distribution has strictly positive entries and belongs to Δ_u.

For any $\boldsymbol{q}_u \in \Delta_u$,

$$\log \frac{q_{uv}}{q^\star_{uv}} = \log \frac{q_{uv}}{\pi_{uv}} + \frac{R_{uv}}{\sigma} + \log Z_u.$$

Multiplying by σq_{uv} and summing over $v \neq u$ yields

$$\sigma \,\mathrm{KL}\left(\boldsymbol{q}_u \| \boldsymbol{q}^\star_u\right) = \sigma \,\mathrm{KL}\left(\boldsymbol{q}_u \| \boldsymbol{\pi}_u\right) + \sum_{v \neq u} q_{uv} R_{uv} + \sigma \log Z_u. \tag{34}$$

Equivalently,

$$\sum_{v \neq u} q_{uv} R_{uv} + \sigma \,\mathrm{KL}\left(\boldsymbol{q}_u \| \boldsymbol{\pi}_u\right) = \sigma \,\mathrm{KL}\left(\boldsymbol{q}_u \| \boldsymbol{q}^\star_u\right) - \sigma \log Z_u. \tag{35}$$

The final term in (35) is independent of $\boldsymbol{q}_u$. Since the KL divergence is nonnegative and vanishes if and only if $\boldsymbol{q}_u = \boldsymbol{q}^\star_u$, the objective has the unique minimizer $\boldsymbol{q}^\star_u$.

Finally, because $\boldsymbol{\pi}_u$ is uniform, its common factor cancels between the numerator and denominator:

$$q^\star_{uv} = \frac{\exp(-R_{uv}/\sigma)}{\sum_{w \neq u} \exp(-R_{uw}/\sigma)}.$$

This is precisely the row-normalized exponential resistance weight Q_{uv}. □

Combining Lemma A.2 and Proposition A.3, the exponential kernel is the unique distribution that minimizes a resistance-based aggregation-distortion proxy together with a KL penalty against the uniform distribution. The parameter σ controls the strength of the entropy regularization: smaller values concentrate mass on low-resistance nodes, whereas larger values produce more diffuse global propagation. This characterization is used only to motivate the construction of $\boldsymbol{Q}$; it does not introduce an additional optimization objective during model training.

A.7 Role of the Bandwidth σ

The bandwidth parameter $\sigma > 0$ controls how sensitively the resistance-based weights respond to differences in effective resistance. For a fixed target node u, the kernel weights are given by Equation (11).

When σ is small, the weights decay rapidly as the effective resistance increases. Consequently, each row of $\boldsymbol{Q}$ concentrates its mass on source nodes with the smallest effective resistance to the target node. More precisely, let

$$\mathcal{S}_u = \arg\min_{w \neq u} R_{\text{eff}}(u, w), \qquad k_u = |\mathcal{S}_u|.$$

Then

$$\lim_{\sigma \to 0^+} Q_{uv} = \begin{cases} 1/k_u, & v \in \mathcal{S}_u, \\ 0, & v \notin \mathcal{S}_u. \end{cases}$$

Importantly, effective resistance reflects the aggregate connectivity and redundancy of all paths between two nodes, rather than only their shortest-path distance. Thus, a node with low effective resistance need not be geographically close in terms of graph distance, and the small-σ regime does not reduce to purely local propagation.

When σ is large, differences in effective resistance become less significant, and the kernel weights become increasingly uniform. In particular,

$$\lim_{\sigma \to \infty} K_{uv} = 1, \qquad u \neq v,$$

and, because $K_{uu} = 0$,

$$\lim_{\sigma \to \infty} Q_{uv} = \begin{cases} \dfrac{1}{n-1}, & u \neq v, \\ 0, & u = v. \end{cases}$$

Thus, $\boldsymbol{Q}$ approaches uniform global propagation.

B Computational Complexity

We analyze the preprocessing, storage, and per-forward-pass complexity of ER-GCN. Let $n = |V|$, $m = |E|$, and let d_l denote the feature dimension at layer l. Let $\mathcal{K} \subseteq \{0, \dots, L-1\}$ be the set of layers using hybrid propagation and write $h = |\mathcal{K}|$.

Resistance preprocessing. For a connected graph, the complete effective-resistance matrix can be computed from the pseudoinverse of the unnormalized Laplacian:

$$R_{\text{eff}}(u, v) = (\boldsymbol{L}^+)_{uu} + (\boldsymbol{L}^+)_{vv} - 2(\boldsymbol{L}^+)_{uv}.$$

Using a dense eigendecomposition or singular-value decomposition, computing $\boldsymbol{L}^+$ requires $O(n^3)$ time and $O(n^2)$ memory. Once $\boldsymbol{L}^+$ is available, evaluating all $n(n-1)$ pairwise resistances, applying the kernel, and row-normalizing the resulting matrix require $O(n^2)$ additional time. Therefore, the exact construction of $\boldsymbol{Q}$ has total preprocessing complexity

$$O(n^3)$$

in time and

$$O(n^2)$$

in memory. This is a one-time cost per graph because $\boldsymbol{Q}$ is fixed and reused throughout training.

Layer-wise propagation. A hybrid layer can be evaluated without explicitly forming $\boldsymbol{M}_l = (1 - \beta_l)\boldsymbol{P} + \beta_l \boldsymbol{Q}$. Instead, we compute

$$\boldsymbol{M}_l \boldsymbol{H} = (1 - \beta_l)\boldsymbol{P}\boldsymbol{H} + \beta_l \boldsymbol{Q}\boldsymbol{H}.$$

Because $\boldsymbol{P}$ has $O(n + m)$ nonzero entries, local propagation costs

$$O\big((n + m)d_l\big).$$

Since $\boldsymbol{Q}$ is dense, resistance-weighted global propagation costs

$$O(n^2 d_l).$$

The feature transformation $\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}$ costs

$$O(n d_l d_{l+1}).$$

By associativity, propagation may be applied before or after the feature transformation. Defining

$$r_l := \min\{d_l, d_{l+1}\},$$

the layer can therefore be implemented with propagation complexity $O((n + m)r_l)$ for the local channel and $O(n^2 r_l)$ for the global channel.

Consequently, one forward pass through the L-layer network costs

$$O\left(\sum_{l=0}^{L-1} n d_l d_{l+1} + \sum_{l=0}^{L-1} (n + m) r_l + \sum_{l \in \mathcal{K}} n^2 r_l\right). \tag{36}$$

The first term is the cost of the feature transformations, the second is the local propagation cost shared with an ordinary GCN, and the third is the additional cost of the resistance-weighted global channel.

For constant hidden width $d_l = d$, this simplifies to

$$O\left(Lnd^2 + L(n + m)d + hn^2 d\right), \tag{37}$$

where $h = |\mathcal{K}|$. A standard L-layer GCN has forward complexity

$$O\left(Lnd^2 + L(n + m)d\right),$$

so the additional forward cost of ER-GCN is

$$O(hn^2 d).$$

For the proposed architecture, which uses hybrid propagation in every propagation layer,

$$h = L.$$

Thus, for $L \geq 1$, the additional dense-propagation cost is

$$O\left(Ln^2 d\right).$$

Memory complexity. Storing the sparse local operator requires $O(n+m)$ memory, whereas storing the dense resistance operator requires $O(n^2)$ memory. Ignoring parameters and saved activations, the graph-operator storage is therefore

$$O(n^2+n+m) = O(n^2+m).$$

During training, storing all layer activations adds

$$O\left(\sum_{l=0}^{L} n d_l\right)$$

memory. The learnable mixing coefficients add only $O(h)$ scalar parameters and are negligible compared with the feature-transformation parameters.

Effective operator used in the analysis. The effective propagation operator

$$\boldsymbol{T}_L^{(\mathcal{K})} = \boldsymbol{S}_{L-1}^{(\mathcal{K})} \cdots \boldsymbol{S}_0^{(\mathcal{K})}$$

is introduced only to characterize propagation and Jacobian support. It is not explicitly constructed during training or inference. Explicitly multiplying these dense matrices would require up to $O(Ln^3)$ time, but this cost is avoided by applying the propagation operators sequentially to node features, as described above.

Graph mini-batches. For a mini-batch containing graphs $G_1, \ldots, G_B$, with n_b nodes and m_b edges in graph b, global propagation is performed independently within each graph. Accordingly, for constant hidden width d, its cost is

$$O\left(hd \sum_{b=1}^{B} n_b^2\right),$$

rather than

$$O\left(hd \left(\sum_{b=1}^{B} n_b\right)^2\right).$$

Likewise, the storage required for the resistance operators is

$$O\left(\sum_{b=1}^{B} n_b^2\right).$$

Thus, the dense channel is practical for collections of small and medium-sized graphs.

C Synthetic Barbell Signal-Transfer Benchmark

Purpose. We generate a controlled node-regression benchmark to isolate information transmission across increasing graph distance and a narrow structural bottleneck. For each bridge parameter, one graph is fixed, a scalar target is determined solely by features on the left clique, and every node of the right clique must recover that scalar. The network depth remains fixed at $L = 10$, so increasing the bridge parameter progressively changes which source–target dependencies lie within the receptive field of a local message-passing model.

Graph construction. For clique size c_s and bridge parameter $p_\ell = \texttt{pl}$, we construct the graph using NetworkX's `barbell_graph`(c_s, p_ℓ). It consists of two copies of the complete graph K_{c_s} joined by a path containing p_ℓ additional bridge nodes. The node sets are

$$C_1 = \{0, \ldots, c_s - 1\}, \tag{38}$$

$$B = \{c_s, \ldots, c_s + p_\ell - 1\}, \tag{39}$$

$$C_2 = \{c_s + p_\ell, \ldots, 2c_s + p_\ell - 1\}, \tag{40}$$

where C_1 is the source clique, B is the bridge, and C_2 is the target clique. The left and right attachment nodes are

$$a_1 = c_s - 1, \qquad a_2 = c_s + p_\ell,$$

respectively. The bridge is connected to the cliques through the edges

$$(c_s - 1, c_s) \quad \text{and} \quad (c_s + p_\ell - 1, c_s + p_\ell).$$

The graph has

$$n = 2c_s + p_\ell \tag{41}$$

nodes and

$$m = 2\binom{c_s}{2} + (p_\ell - 1) + 2 = c_s(c_s - 1) + p_\ell + 1 \tag{42}$$

edges. Every edge on the chain connecting the two cliques is a bridge of the graph. Interior bridge nodes have degree two, non-attachment clique nodes have degree $c_s - 1$, and each attachment node has degree c_s.

The parameter p_ℓ counts bridge nodes rather than graph hops. The distance between the two attachment nodes is

$$d_G(a_1, a_2) = p_\ell + 1, \tag{43}$$

whereas the distance between a non-attachment node in C_1 and a non-attachment node in C_2 is the graph diameter,

$$\operatorname{diam}(G) = p_\ell + 3. \tag{44}$$

The additional two hops connect the peripheral source and target nodes to their respective attachment nodes.

Under unit edge resistances, the unique path between the attachment nodes gives

$$R_{\text{eff}}(a_1, a_2) = p_\ell + 1. \tag{45}$$

Because the effective resistance between two distinct nodes of K_{c_s} is $2/c_s$, the resistance between non-attachment nodes $u \in C_1 \setminus \{a_1\}$ and $v \in C_2 \setminus \{a_2\}$ is

$$R_{\text{eff}}(u, v) = p_\ell + 1 + \frac{4}{c_s}. \tag{46}$$

All reported experiments use $c_s = 12$. The smaller schematic based on `barbell_graph`$(6, 8)$, which has 20 nodes and diameter 11, is included only for illustration and is not the graph used to produce the reported results.

Regression task. A projection vector $\boldsymbol{w} \in \mathbb{R}^d$ is drawn once and held fixed across all samples. Each sample draws fresh features only for nodes in the source clique:

$$\boldsymbol{x}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d), \quad i \in C_1, \qquad \boldsymbol{x}_i = \mathbf{0}, \quad i \notin C_1. \tag{47}$$

The scalar regression target is

$$y = \frac{\dfrac{1}{c_s} \displaystyle\sum_{i \in C_1} \boldsymbol{x}_i^\top \boldsymbol{w}}{\sqrt{d/c_s}}. \tag{48}$$

Every node of the target clique receives the same label:

$$y_v = y, \qquad v \in C_2.$$

Conditioned on $\boldsymbol{w}$, the variance of the unnormalized clique average is

$$\operatorname{Var}\left(\frac{1}{c_s} \sum_{i \in C_1} \boldsymbol{x}_i^\top \boldsymbol{w} \,\middle|\, \boldsymbol{w} \right) = \frac{\|\boldsymbol{w}\|_2^2}{c_s}.$$

The denominator in (48) therefore places the target on an approximately unit scale when $\|\boldsymbol{w}\|_2^2 \approx d$. In the experiments, $d = 16$ and $c_s = 12$, giving the normalization factor $\sqrt{4/3}$.

The training objective is the mean squared error over the target clique only:

$$\mathcal{L}_{\text{train}} = \frac{1}{|\mathcal{D}_{\text{train}}||C_2|} \sum_{t \in \mathcal{D}_{\text{train}}} \sum_{v \in C_2} \left(\widehat{y}_v^{(t)} - y^{(t)} \right)^2. \tag{49}$$

Nodes outside C_2 are not supervised. Because the bridge and target clique have identically zero input features, a prediction on C_2 can depend on y only if information from C_1 reaches the target node. There is no target-side local cue.

Effect of the bridge parameter at fixed depth. The local propagation operator is the self-looped random-walk matrix

$$\widetilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}, \qquad \widetilde{\boldsymbol{D}} = \operatorname{diag}(\widetilde{\boldsymbol{A}}\mathbf{1}), \qquad \boldsymbol{P} = \widetilde{\boldsymbol{D}}^{-1}\widetilde{\boldsymbol{A}}. \tag{50}$$

An L-layer local message-passing model can transmit information from node u to node v only if

$$d_G(u, v) \le L.$$

Self-loops allow a propagation path to remain at a node, so the condition is at most L hops rather than exactly L hops.

For $c_s = 12$ and $L = 10$, every source node can reach every target node when $p_\ell \in \{4, 6\}$. Nevertheless, all inter-clique information must pass through the unique chain of degree-two bridge nodes. At each interior bridge node, the self-looped random-walk operator assigns weight $1/3$ to each neighbor and $1/3$ to the node itself, so the signal is attenuated even while all source–target pairs remain within the receptive field.

At $p_\ell = 8$, the dependence becomes only partially supported. The right attachment node can receive information from every node of C_1, but each of the other target nodes can receive information only from the left attachment node; the remaining source features are at distance $11 > L$. For $p_\ell \ge 10$, the minimum distance between C_1 and C_2 is $p_\ell + 1 > 10$. Consequently, no source node lies in the receptive field of any target node, and every C_1-to-C_2 Jacobian block of a purely local 10-layer model is structurally zero.

The Barbell models are bias-free. Thus, when the input receptive field of a target node is identically zero, its representation and prediction remain zero because $\tanh(0) = 0$. On the fixed eight-sample test set, the zero predictor has

$$\text{MAE}(0, y_{\text{test}}) = \frac{1}{8} \sum_{t=1}^{8} \left| y^{(t)} \right| = 0.664. \tag{51}$$

This value is the horizontal reference in the Barbell plots and is the score forced on a bias-free local model when $p_\ell \ge 10$.

Data and optimization protocol. For each value of p_ℓ, the data are generated using

```
barbell_data( cs=12, pl, fd=16, seed=0, n_train=16, n_test=8).
```

This produces one graph, 16 fixed training feature matrices, and 8 fixed test feature matrices. The graph, projection vector $\boldsymbol{w}$, and train–test split are shared by every method and do not depend on the model seed. Test features are generated using PyTorch seed 9999. The ten model seeds therefore vary only the network initialization.

Models are trained for 500 epochs using Adam, hidden width 64, depth $L = 10$, no dropout, and cosine learning-rate annealing. The base learning rate is 10^{-3}, and the weight decay is 10^{-5}. Evaluation is performed every ten epochs.

Test MAE is computed over the 12 target nodes and eight fixed test samples:

$$\text{MAE}_{\text{test}} = \frac{1}{8|C_2|} \sum_{t=1}^{8} \sum_{v \in C_2} \left| \widehat{y}_v^{(t)} - y^{(t)} \right|. \tag{52}$$

Following the benchmark implementation, the reported score for each model seed is the lowest test MAE observed during training. The graph and data split are identical across methods. Across values of p_ℓ, only the number of bridge nodes—and hence the graph size, distance, resistance, and bottleneck severity—changes.

Table 4: Resistance-kernel bandwidths used in the main experiments. A single value is used for all TreeNeighborsMatch depths and all Barbell path lengths.

σ	Datasets
5	MUTAG, IMDB-BINARY, ZINC, TreeNeighborsMatch
10	PROTEINS, Peptides-func, chameleon, squirrel, actor, Barbell
20	NCI1, DD, ENZYMES, REDDIT-BINARY, Peptides-struct

D Experiment Configurations and Hyperparameters

Evaluation protocol. Unless stated otherwise, we report the test performance obtained at the epoch with the best validation score. The TU graph-classification datasets use shared 10-fold stratified cross-validation (`random_state=42`) across all methods. Within each training fold, 10% of the graphs are reserved for validation. ZINC, Peptides, TreeNeighborsMatch, and Barbell use ten runs with seeds $0, \ldots, 9$, while the heterophilic node-classification datasets use the ten fixed Geom-GCN splits (Pei et al., 2020). Each seed controls the PyTorch, NumPy, and Python random-number generators.

The reported metrics are accuracy for TU, TreeNeighborsMatch, and heterophilic node classification; average precision (AP) for Peptides-func; and mean absolute error (MAE) for ZINC, Peptides-struct, and Barbell. Statistical significance is assessed using paired t-tests over matched folds, seeds, or official splits. The symbols † and ‡ indicate $p < 0.05$ and $p < 0.01$, respectively, and ↓ indicates that ER-GCN is significantly worse than the corresponding baseline. Accuracy means and standard deviations are reported to two decimal places; AP and MAE means use four significant figures and their standard deviations use two significant figures.

ER-GCN construction. For each graph, we compute the effective-resistance matrix from the pseudoinverse of the unnormalized graph Laplacian:

$$R_{uv} = (\boldsymbol{L}^{+})_{uu} + (\boldsymbol{L}^{+})_{vv} - 2(\boldsymbol{L}^{+})_{uv}.$$

The resistance computation and kernel construction are performed in double precision. The main experiments use the exponential kernel

$$K_{uv} = \begin{cases} \exp(-R_{uv}/\sigma), & u \neq v, \\ 0, & u = v, \end{cases} \qquad \boldsymbol{Q} = \operatorname{diag}(\boldsymbol{K}\mathbf{1})^{-1}\boldsymbol{K}.$$

Thus, $\boldsymbol{Q}$ is exactly row-normalized, has zero diagonal, and is cast to single precision before training. The local operator is

$$\boldsymbol{P} = \operatorname{diag}((\boldsymbol{A} + \boldsymbol{I})\mathbf{1})^{-1}(\boldsymbol{A} + \boldsymbol{I}).$$

In all main experiments, hybrid propagation is applied at every propagation layer:

$$\boldsymbol{S}^{(l)} = (1 - \beta_l)\boldsymbol{P} + \beta_l \boldsymbol{Q}, \qquad \beta_l = \operatorname{sigmoid}(\theta_l).$$

Each layer has an independent mixing coefficient initialized at $\beta_l = 0.1$. The parameters $\{\theta_l\}$ form a separate optimizer group with ten times the base learning rate and zero weight decay. The graph-prediction models otherwise use bias-free linear transformations followed by batch normalization, ReLU, and dropout, with mean pooling for graph-level prediction. ZINC omits batch normalization, TreeNeighborsMatch uses the root-node representation, and Barbell uses $\tanh$ activations without batch normalization or dropout.

The kernel bandwidth σ is selected once per dataset using short seed-0 runs. The selected values are summarized in Table 4. The heterophilic datasets use the fixed value $\sigma = 10$. The Gaussian kernel is considered only in the kernel ablation; all main results use the exponential kernel.

Benchmark-specific configurations. Table 5 summarizes the principal model and optimization settings. Gradient norms are clipped at 1.0 except on Barbell, where the threshold is 5.0. For TU datasets containing graphs with more than 500 nodes, the batch size is reduced from 32 to 4. Training automatically retries with a smaller batch size after an out-of-memory error.

Table 5: Main ER-GCN training configurations. Here d_{hid}, L, and p denote hidden width, number of propagation layers, and dropout, respectively. The learning rate of the mixing parameters is ten times the listed base learning rate.

Benchmark	Architecture	Optimization	Batch and model selection
TU	$d_{\mathrm{hid}} = 64$, $L = 3$, $p = 0.5$	Adam, lr 10^{-3}, wd $5{\times}10^{-4}$; 200 epochs; early stopping patience 50; ReduceLROnPlateau with factor 0.5 and patience 20	Batch size 32, or 4 for large-graph datasets; test accuracy at the best-validation epoch
ZINC	$d_{\mathrm{hid}} = 128$, $L = 4$, $p = 0$; no batch normalization; orthogonal initialization	Adam, lr 10^{-3}, wd 10^{-5}; L1 loss; 300 epochs; early stopping patience 80; ReduceLROnPlateau with patience 30	Batch size 64; test MAE at the best-validation epoch
Peptides	$d_{\mathrm{hid}} = 128$, $L = 5$, $p = 0.1$; approximately 70k parameters	Peptides-func: Adam; Peptides-struct: AdamW; lr 10^{-3}, wd 0; 100 epochs; cosine learning-rate schedule	Batch size 32; test AP or MAE at the best-validation epoch
Heterophilic	$d_{\mathrm{hid}} = 64$, $L = 8$, $p = 0.5$	Adam, lr 10^{-3}, wd $5{\times}10^{-4}$; 500 epochs; early stopping patience 80; ReduceLROnPlateau with patience 30	Full-batch training; test accuracy at the best-validation epoch
TreeNeighbors Match	$d_{\mathrm{hid}} = 64$, $L = d + 2$, $p = 0.1$; root-node readout	Adam, lr 10^{-3}, wd 10^{-5}; 200 epochs; early stopping patience 50; ReduceLROnPlateau with patience 20	Batch size $\max(1, \min(64, \lfloor 12000/n \rfloor))$; best-validation selection
Barbell	$d_{\mathrm{hid}} = 64$, $L = 10$; $\tanh$; no batch normalization or dropout	Adam, lr 10^{-3}, wd 10^{-5}; 500 epochs; cosine schedule with minimum lr 10^{-5}; MSE training loss	Full-batch training; evaluation every 10 epochs and best test MAE reported

For TU datasets, available node attributes are used as inputs. Datasets without node attributes use degree one-hot encodings capped at dimension 64. ZINC uses a 28-dimensional one-hot encoding of atom type. Peptides uses the provided nine-dimensional node features. TreeNeighborsMatch uses nine-dimensional inputs and contains $1{,}000/200/200$ training, validation, and test graphs at each depth. Its graph size is $n = 2^{d+1} - 1$, and all graphs at the same depth share a common tree topology, allowing the resistance operator to be computed once per depth.

Baselines. All baselines use the same data splits, model-selection protocol, hidden width, depth, dropout, batch size, and optimization budget as ER-GCN for the corresponding benchmark. GAT uses four attention heads; APPNP uses teleport probability $\alpha = 0.1$; and GCNII uses $\alpha = 0.1$, $\theta = 0.5$, and $\beta_l = \log(1 + \theta/l)$. AMP uses `theta_prior_scale` $= 10$ without its optional ELBO auxiliary loss. GCN+VN and VN$_G$ use their respective virtual-node updates while retaining the same benchmark-specific training shell.

Implementation and compute. Experiments are implemented in Python 3.11 using PyTorch (Paszke et al., 2019) and PyTorch Geometric (PyG) (Fey & Lenssen, 2019; Fey et al., 2025). Resistance operators and rewired graphs are cached per dataset and reused across repetitions. Every pipeline is resume-safe, and all individual fold-, seed-, and split-level scores are retained for statistical analysis.

E Additional Experimental Results

This section reports additional experiments that evaluate ER-GCN beyond the explicitly long-range settings considered in the main text and examine the effects of the global operator, hybrid-layer placement, adaptive mixing, and network depth.

E.1 SHORT-RANGE GRAPH CLASSIFICATION

We evaluate ER-GCN on five conventional graph-classification benchmarks from the TU collection (Morris et al., 2020): MUTAG (Debnath et al., 1991; Kriege & Mutzel, 2012), PROTEINS (Borgwardt et al., 2005; Dobson & Doig, 2003), NCI1 (Wale & Karypis, 2006; Shervashidze et al., 2011), and IMDB-BINARY and REDDIT-BINARY (Yanardag & Vishwanathan, 2015). These datasets are not explicitly constructed to require information propagation over prescribed graph distances or structural bottlenecks. We test whether dense resistance-weighted propagation preserves the performance of local message passing when no explicit long-range dependency is imposed.

Table 6: Short-range graph classification on five TU datasets (accuracy in %, ↑; 10-fold cross-validation). Boldface indicates the highest mean in each column. † and ‡ denote paired differences from ER-GCN at $p < 0.05$ and $p < 0.01$, respectively; ↓ indicates that the baseline significantly outperforms ER-GCN.

Model	MUTAG	PROTEINS	NCI1	IMDB-B	REDDIT-B
GCN	71.87 ± 8.09	68.73 ± 3.69	67.08 ± 2.33	73.20 ± 4.78	81.90 ± 3.22
GIN	68.10 ± 3.33	68.91 ± 3.68	68.69 ± 2.78	$\mathbf{74.20 \pm 4.34}$	$84.65 \pm 3.07^{\dagger}\downarrow$
GAT	70.73 ± 6.25	$66.76 \pm 4.79^{\ddagger}$	$65.06 \pm 1.28^{\ddagger}$	72.60 ± 3.60	$76.70 \pm 3.89^{\ddagger}$
JKNet	71.87 ± 6.85	67.93 ± 5.37	67.23 ± 2.20	73.30 ± 5.48	81.95 ± 2.44
APPNP	68.07 ± 2.59	$62.45 \pm 3.81^{\ddagger}$	$57.62 \pm 1.77^{\ddagger}$	$71.10 \pm 4.68^{\dagger}$	$69.90 \pm 2.96^{\ddagger}$
GCNII	$\mathbf{73.42 \pm 6.48}$	$66.76 \pm 3.86^{\dagger}$	$66.35 \pm 2.49^{\dagger}$	73.40 ± 3.72	$76.00 \pm 4.12^{\ddagger}$
GCN+VN	68.60 ± 4.71	$\mathbf{71.43 \pm 4.14}$	66.28 ± 3.23	71.90 ± 6.30	84.25 ± 1.78
VN_G	66.49 ± 2.40	71.25 ± 4.41	$\mathbf{68.98 \pm 3.59}$	72.80 ± 5.01	$\mathbf{87.00 \pm 2.86}^{\ddagger}\downarrow$
AMP	71.81 ± 7.42	67.93 ± 3.79	68.25 ± 2.71	73.50 ± 4.06	83.75 ± 1.72
GTR	70.79 ± 5.58	69.72 ± 4.40	$65.52 \pm 1.66^{\ddagger}$	72.40 ± 5.08	82.35 ± 2.57
ER-GCN	70.79 ± 8.62	69.81 ± 4.58	67.57 ± 2.24	73.60 ± 5.17	82.15 ± 3.62

ER-GCN remains competitive with its GCN backbone across these benchmarks but does not consistently attain the best accuracy. The largest gap occurs on REDDIT-BINARY, where GIN and VN_G significantly outperform ER-GCN. These results indicate that dense resistance-weighted propagation provides limited additional benefit on benchmarks not designed around long-range dependencies, while generally preserving the performance of local message passing.

E.2 HETEROPHILIC NODE CLASSIFICATION

We next evaluate ER-GCN on heterophilic node-classification benchmarks, which present a challenge distinct from long-range communication. In heterophilic graphs, adjacent or structurally well-connected nodes may have dissimilar features and labels. Because effective resistance measures structural connectivity rather than label compatibility, low-resistance node pairs do not necessarily provide informative signals for one another. Dense resistance-weighted propagation may therefore mix representations from semantically dissimilar nodes, making these benchmarks particularly challenging for ER-GCN.

ER-GCN remains competitive on these benchmarks, performing comparably to its GCN backbone and several other baselines and generally occupying the middle-to-upper range of the evaluated methods on chameleon and squirrel. Although models such as JKNet and AMP achieve stronger overall results, ER-GCN does not exhibit substantial performance degradation despite the mismatch between its structural prior and the heterophilic setting. Effective-resistance proximity does not necessarily align with label similarity, so dense global propagation does not directly resolve the discrepancy between graph connectivity and class structure. These results indicate that ER-GCN remains reasonably robust under heterophily, while its primary advantage lies in addressing long-range communication.

E.3 GLOBAL-OPERATOR ABLATION

To distinguish the effect of dense global support from the effect of resistance-aware weighting, we replace the ER-GCN global operator with local, uniform, and random alternatives.

Table 7: Heterophilic node classification (accuracy in %, ↑) over the 10 official splits. Boldface indicates the highest mean in each column. † and ‡ denote paired differences from ER-GCN at $p < 0.05$ and $p < 0.01$, respectively; ↓ indicates that the baseline significantly outperforms ER-GCN.

Model	chameleon	squirrel	actor
GCN	51.43 ± 2.29	37.26 ± 2.07	25.60 ± 1.54
GIN	$40.13 \pm 3.92^{\ddagger}$	$27.16 \pm 2.54^{\ddagger}$	25.45 ± 1.48
GAT	53.25 ± 4.14	$20.66 \pm 2.08^{\ddagger}$	25.41 ± 1.37
JKNet	$\mathbf{59.39 \pm 1.60}^{\ddagger}\downarrow$	$44.88 \pm 1.93^{\ddagger}\downarrow$	$\mathbf{34.65 \pm 1.11}^{\ddagger}\downarrow$
APPNP	$46.40 \pm 3.40^{\ddagger}$	35.80 ± 1.83	$31.78 \pm 1.11^{\ddagger}\downarrow$
GCNII	53.31 ± 3.56	$37.74 \pm 1.49^{\dagger}\downarrow$	$30.39 \pm 3.34^{\ddagger}\downarrow$
GCN+VN	50.15 ± 4.33	$30.63 \pm 2.91^{\ddagger}$	25.25 ± 1.30
VN_G	$42.30 \pm 9.64^{\dagger}$	$28.03 \pm 4.53^{\ddagger}$	25.36 ± 1.23
AMP	$58.71 \pm 1.72^{\ddagger}\downarrow$	$\mathbf{45.87 \pm 1.53}^{\ddagger}\downarrow$	$30.91 \pm 1.77^{\ddagger}\downarrow$
GTR	$34.52 \pm 4.89^{\ddagger}$	$22.49 \pm 1.86^{\ddagger}$	25.18 ± 1.12
ER-GCN	51.95 ± 4.13	36.20 ± 1.74	25.38 ± 1.04

Table 8: Global-operator ablation on Barbell with $pl = 10$ and adaptive mixing coefficients (test MAE, ↓; 10 seeds).

Shortcut type	MAE ↓
None (GCN)	0.6641 ± 0.00
Local adjacency (1-hop)	0.6641 ± 0.00
Identity (self-only)	0.6641 ± 0.00
Uniform all-to-all	0.1286 ± 0.021
Random dense (avg. 3)	0.1288 ± 0.021
Resistance-exp (*ours*)	0.1213 ± 0.017

The GCN, one-hop local-adjacency, and self-only variants remain at the zero-predictor error because they preserve one-hop local support at every propagation layer. Specifically, the local-adjacency alternative uses $\boldsymbol{Q}_{\text{local}} = \boldsymbol{D}^{-1}\boldsymbol{A}$, which differs from the self-looped base operator but introduces no nonlocal edges. Because the Barbell models are bias-free and the target-side input features are zero, these local variants reduce to the zero predictor. In contrast, the uniform, random nonlocal, and resistance-weighted operators introduce cross-graph propagation paths and substantially reduce the MAE. The resistance-weighted operator achieves the lowest mean MAE, suggesting an additional benefit from topology-aware weighting.

E.4 Resistance-Kernel Ablation

We examine whether the direction and functional form of the mapping from effective resistance to propagation weight affect performance. The comparison includes decreasing kernels, which assign greater weight to structurally well-connected node pairs; increasing kernels, which favor high-resistance pairs; and uniform global propagation. For each parameterized kernel, the scale σ is selected using validation performance.

All decreasing kernels such as $\exp(-R/\sigma)$, $\max(1 - R/R_{\max}, 0)$, and $(1 + R/\sigma)^{-1}$ achieve lower mean MAE than the increasing kernels $1 - \exp(-R/\sigma)$, $\text{sigmoid}(R/\sigma)$, and $\tanh(R/\sigma)$. Uniform weighting $g(R) = 1$ yields intermediate performance. These results support the use of a decreasing resistance kernel, with the exponential form achieving the best overall performance.

E.5 Adaptive Versus Fixed Mixing

We compare the learned layer-wise mixing coefficients with fixed coefficients to determine whether adaptive local–global mixing is beneficial beyond selecting a single global propagation strength.

Table 9: Resistance-kernel ablation on Barbell with $pl = 10$ and adaptive mixing coefficients (test MAE, ↓; 10 seeds). Each parameterized kernel is evaluated using its validation-selected scale.

Resistance kernel $g(R)$	MAE ↓
$\exp(-R/\sigma)$ (*ours*)	0.1233 ± 0.017
$\max(1 - R/R_{\max}, 0)$	0.1236 ± 0.019
$(1 + R/\sigma)^{-1}$	0.1281 ± 0.015
1	0.1286 ± 0.021
$1 - \exp(-R/\sigma)$	0.1579 ± 0.025
$\mathrm{sigmoid}(R/\sigma)$	0.1393 ± 0.022
$\tanh(R/\sigma)$	0.1590 ± 0.025

Table 10: Fixed and adaptive mixing coefficients on Peptides-func using the exponential resistance kernel with $\sigma = 10$. Results are reported over three shared seeds. The adaptive configuration is the main ER-GCN model evaluated on the same seeds. † and ‡ denote paired differences from the adaptive configuration at $p < 0.05$ and $p < 0.01$, respectively.

Mixing coefficient β	Peptides-func AP ↑
β=0.1	$0.5255 \pm 0.0021^{\ddagger}$
β=0.5	0.6433 ± 0.0021
β=0.9	$0.6454 \pm 0.0019^{\dagger}$
adaptive β (*ours*)	$\mathbf{0.6567 \pm 0.0040}$

Adaptive mixing achieves the highest mean average precision among the evaluated configurations. The weak performance at $\beta = 0.1$ indicates that this task benefits from a substantial global contribution, while the stronger results at $\beta \in \{0.5, 0.9\}$ show that fixed global mixing can recover much of the gain. Learning separate coefficients nevertheless avoids manual selection of a single mixing strength and allows different layers to specialize.

E.6 Depth and Layer-Wise Mixing

We examine how network depth affects both predictive performance and the learned allocation between local and global propagation.

Table 11: Depth sweep on Peptides-func, seed 0 (subset, exp kernel σ=10): learned per-layer β_l (shallow → deep) and test AP.

#Propagation layers	Learned β_l (shallow → deep)	AP ↑
2	[0.705, 0.985]	0.4591
3	[0.118, 0.980, 0.967]	0.5033
4	[0.069, 0.391, 0.977, 0.938]	0.5036
5	[0.035, 0.044, 0.963, 0.970, 0.943]	0.5183
6	[0.027, 0.035, 0.392, 0.965, 0.968, 0.941]	0.5494

Test average precision increases throughout the evaluated depth sweep, from 0.4621 with $L = 2$ propagation layers to 0.5337 with $L = 6$ propagation layers. As depth increases, the shallowest mixing coefficients approach zero, whereas the deepest coefficients remain close to one. This pattern is consistent with an emergent division of labor in which early layers emphasize local feature extraction and later layers integrate global information.

Table 12: Depth sweep on the Peptides-func subset using three seeds and the exponential resistance kernel with $\sigma = 10$. The table reports the learned layer-wise coefficients from shallow to deep layers and test average precision. Coefficients are averaged over seeds.

#Propagation layers	Learned β_l (shallow $\rightarrow$ deep)	AP $\uparrow$
2	[0.760, 0.980]	0.4621 ± 0.0050
3	[0.116, 0.978, 0.962]	0.4971 ± 0.0081
4	[0.073, 0.375, 0.982, 0.953]	0.5062 ± 0.0084
5	[0.040, 0.047, 0.948, 0.968, 0.938]	0.5211 ± 0.0039
6	[0.032, 0.031, 0.352, 0.971, 0.958, 0.935]	0.5337 ± 0.0110