TREE RESIDUAL NETWORK: RESTORING GRAPH CONNECTIVITY OVER A TREE BACKBONE

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph node classification requires both local connectivity and long-range context. Message-passing networks preserve graph edges but require repeated aggregation to communicate over long distances, whereas dense global attention is expensive. We propose Tree Residual Network (TRN), a cycle-completion architecture that assigns global transport to a tree and restores the constraints of omitted edges through the same routing coordinate system. Bidirectional aggregation over a spanning tree establishes graph-wide context with a linear-size backbone. Because a tree can turn one-hop relations into long detours, TRN measures signed discrepancies on omitted edges and redistributes the resulting corrections through the backbone rather than opening a second residual propagation channel. Complementary theoretical analyses, including a signed-pair reference model and a quadratic graph-smoothing surrogate, provide operator-level perspectives on this tree-to-graph correction. Across 14 homophilic and heterophilic node-classification benchmarks, TRN attains the highest or second-highest reported mean on every dataset; controlled ablations and a matched residual-aggregation control support the proposed decomposition.

1 INTRODUCTION

Graph representation learning needs both local connectivity and long-range context. Message-passing neural networks (MPNNs) (Gilmer et al., 2017), including GCNs (Kipf & Welling, 2017; Hamilton et al., 2017; Abu-El-Haija et al., 2019; Wu et al., 2019; Chen et al., 2020b) and GATs (Veličković et al., 2018; Brody et al., 2022), exploit graph edges efficiently, but distant information requires repeated aggregation and can suffer from over-smoothing (Chen et al., 2020a; Oono & Suzuki, 2020) or over-squashing (Alon & Yahav, 2021; Topping et al., 2022). Graph Transformers (Dwivedi & Bresson, 2020; Rampášek et al., 2022; Kreuzer et al., 2021; Deng et al., 2024; Shirzad et al., 2023) broaden the receptive field, but dense attention is costly and commonly augmented with structural encodings or local modules (Dwivedi et al., 2022; Belkin & Niyogi, 2003; Wang et al., 2022; Rampášek et al., 2022; Wu et al., 2022; 2023; Bo et al., 2023). The challenge is therefore not merely to enlarge the receptive field: a useful architecture must expose graph-wide context while retaining the local relations that make the graph informative.

We approach this challenge by separating *transport* from *constraint restoration*. A rooted spanning tree offers a particularly economical transport structure: one bottom-up and one top-down pass route information between every pair of nodes using only a linear-size backbone. The price of this economy is explicit and structured. A tree has a unique path between two nodes, so an original one-hop edge omitted during tree construction is represented only through a potentially long detour. Rather than treating this mismatch as incidental, we use it as the organizing principle of the model.

The omitted edges have precisely the right structure for this purpose. Every non-tree edge closes one fundamental cycle with its tree path; collectively, these edges index the cycle-space information absent from the tree. They therefore identify where tree-routed representations should be tested against the original graph. This observation distinguishes our objective from simply adding another residual message-passing layer: the residual graph should diagnose what remains unresolved after global routing, and the resulting correction should be redistributed through the same global transport system.

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.

Existing tree-based graph learning (Wu & Wang, 2022; Bonchi et al., 2025; Huang et al., 2024; Yang et al., 2025) uses trees as structural views, sampled backbones, or auxiliary propagation structures. Forest-based graph learning (FGL) (Li et al., 2026) further improves tree selection and edge coverage with homophily-aware construction and multiple sampled trees. TRN addresses a complementary question: given a single tree, how can every omitted edge contribute without creating a second global propagation channel? This formulation is particularly natural when informative edges are not necessarily homophilic (Pei et al., 2020; Platonov et al., 2023). Motivated by this view, we propose **Tree Residual Network** (TRN), a node-classification architecture that restores graph connectivity over a single tree backbone. TRN first constructs a spanning tree and performs bidirectional aggregation to establish global context. It then evaluates every omitted edge on the tree-routed states, forms a signed endpoint discrepancy, and re-injects that correction through the same tree operator. Residual edges thus serve as constraint channels rather than independent neighbor-aggregation channels. This cycle-completion design makes the model's division of labor explicit: the tree is the global transport mechanism, while residual edges provide discrepancy coordinates for restoring local connectivity. The analysis follows the same decomposition: an idealized signed-pair calculation illustrates the selectivity of discrepancy re-routing, while a quadratic graph-smoothing surrogate yields an exact tree-to-graph correction identity and finite-iteration certificate.

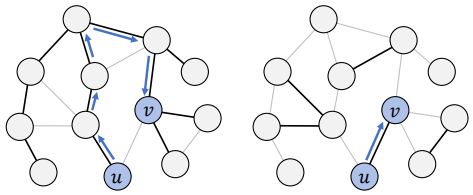

Figure 1: Connectivity mismatch caused by tree-only propagation: a one-hop relation in the residual/original graph (right) may become a long path in the tree backbone (left).

The main contributions of this paper are summarized as follows:

- We propose TRN for node classification, introducing a cycle-completion propagation principle that uses a single tree as a global routing coordinate system and restores omitted edge constraints through signed correction.
- We provide complementary theoretical perspectives on the tree–residual decomposition, including a signed-pair analysis and a quadratic graph-smoothing surrogate; the appendix provides further perspectives on correction dynamics and tree structure.
- We evaluate TRN on 14 homophilic and heterophilic benchmarks. Controlled ablations and sensitivity analyses assess the contribution and robustness of the proposed components; additional analyses report implementation-level efficiency.

2 Related Work

2.1 Message Passing Neural Networks

Message-passing neural networks (Gilmer et al., 2017) are a dominant paradigm for graph representation learning. Representative models such as GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), and GAT (Veličković et al., 2018) learn node representations by recursively aggregating information from local neighborhoods. This design provides a strong inductive bias for exploiting graph connectivity and has achieved strong performance on many node classification benchmarks. However, since information is propagated only through adjacent edges at each layer, long-range dependencies require stacking multiple message-passing layers. Deep propagation may lead to over-smoothing, where node representations become less distinguishable (Chen et al., 2020a; Oono & Suzuki, 2020), and over-squashing, where distant information is compressed through narrow graph bottlenecks (Alon & Yahav, 2021; Topping et al., 2022). Several works improve propagation through residual connections (Chen et al., 2020b), diffusion or higher-order neighborhoods (Klicpera et al., 2019; Chamberlain et al., 2021; Abu-El-Haija et al., 2019; Xu et al., 2018; Zhu et al., 2020), rewiring (Feng et al., 2020; Gutteridge et al., 2023), adaptive propagation (Chien et al., 2021; Shirzadi et al., 2026), or signed aggregation (Bo et al., 2021; Chien et al., 2021; Luan et al., 2022; Li et al., 2022; Jiaqi et al., 2025). Rewiring modifies the input graph to suit local message passing; TRN instead modifies the propagation structure itself: it makes the tree a routing coordinate system, repre-

sents each omitted edge as a signed fundamental-cycle constraint, and redistributes that discrepancy through the same global router rather than directly aggregating residual-edge messages.

2.2 Graph Transformers

Graph Transformers extend the Transformer architecture (Vaswani et al., 2017) to graph-structured data by enabling broader interactions between nodes (Ying et al., 2021; Rampášek et al., 2022; Wu et al., 2022; 2023; Deng et al., 2024). Compared with local message passing, attention-based models can capture long-range dependencies more directly and reduce the reliance on deep multi-hop propagation. However, vanilla self-attention does not explicitly encode graph topology and dense all-pair attention can be costly on large graphs. To address these issues, existing Graph Transformers typically introduce graph-specific structural or positional encodings (Dwivedi et al., 2022; Belkin & Niyogi, 2003; Dwivedi & Bresson, 2020; Kreuzer et al., 2021; Wang et al., 2022), combine global attention with local GNN modules (Rampášek et al., 2022; Wu et al., 2022; 2023; Bo et al., 2023), or improve scalability through sparse attention (Shirzad et al., 2023; Zhang et al., 2024; Wu et al., 2022; 2023), node sampling (Zhang et al., 2022; Fu et al., 2024), and tokenized graph representations (Chen et al., 2023; Ma et al., 2024; Chen et al., 2024). Despite these designs, they still depend on attention-based global interactions or sampled graph representations to model long-range dependencies.

2.3 Tree-based Graph Learning

Tree models serve as semantic hierarchies in NLP (Socher et al., 2013; Tai et al., 2015; Zhang et al., 2016; Miwa & Bansal, 2016) and as structural views or propagation backbones in graphs (Wu & Wang, 2022; Bonchi et al., 2025; Huang et al., 2024; Yang et al., 2025). FGL (Li et al., 2026), the closest learning architecture, improves tree selection and edge coverage through homophily-aware construction and multiple sampled trees. TRN addresses a different problem after a tree is fixed: it closes the fundamental cycles created by omitted edges, using their discrepancies as correction coordinates for the same global router. Classical sparsification and tree preconditioning approximate or accelerate a fixed Laplacian operator (Calandriello et al., 2018; Spielman & Woo, 2009); our surrogate analysis borrows this operator view, whereas the learned discrepancy-and-rerouting decomposition is the modeling contribution.

3 Preliminaries

3.1 Notations

Let $G = (V, E)$ be an undirected graph with $n = |V|$ nodes, $m = |E|$ edges, and node features $\mathbf{X} \in \mathbb{R}^{n \times d_0}$. We construct a spanning forest $T = (V, E_T)$, one tree per connected component, and define the residual graph $R = (V, E_R)$ by $E_R = E \setminus E_T$. We describe one connected component below; all operations apply component-wise. For any $Q = (V, E_Q)$, let $\mathcal{N}_Q(u) = \{v : (u, v) \in E_Q\}$ and $d_Q(u) = |\mathcal{N}_Q(u)|$. Each tree is rooted to schedule the two propagation passes; the implementation uses the shallow-root heuristic in Appendix F. For node v, let $\pi(v)$ and $\mathcal{C}(v)$ denote its parent and children, with $\pi(r) = \varnothing$ at root r. We write $\mathrm{Sub}(v)$ for the rooted subtree, $P_T(u, v)$ for the unique undirected tree path, and $\mathrm{LCA}(u, v)$ for the lowest common ancestor.

3.2 Tree Aggregation

We first formalize the bidirectional tree aggregation scheme used throughout this paper.

Bidirectional tree propagation. Tree aggregation consists of a bottom-up pass followed by a top-down pass. For a node v and its parent $p = \pi(v)$, we define the messages transmitted along the tree edge (v, p) in the bottom-up and top-down phases as $\mathbf{M}_{v\to p}$ and $\mathbf{M}_{p\to v}$, respectively:

$$\mathbf{M}_{v\to p} = f_T\left(\mathbf{X}_v, \mathbf{X}_p, \mathbf{X}_v + f_{\text{agg}}\left(\{\mathbf{M}_{u\to v} : u \in \mathcal{C}(v)\}\right)\right), \tag{1}$$

$$\mathbf{M}_{p\to v} = f_T\left(\mathbf{X}_p, \mathbf{X}_v, \mathbf{M}_{\pi(p)\to p} + f_{\text{agg}}\left(\{\mathbf{M}_{u\to p} : u \in \mathcal{C}(p) \setminus \{v\}\}\right)\right). \tag{2}$$

Here, f_T is a directed tree-edge transition function. Its first two arguments denote the source and target node representations, respectively, and its third argument denotes the signal propagated from the

source side to the target side. The function f_{agg} is a permutation-invariant aggregation function over incoming signals. For boundary cases, empty message sets and messages involving $\varnothing$ are replaced by the zero vector. Thus, a leaf sends its own state in the bottom-up pass, and every propagated source signal is explicitly defined. After the bottom-up and top-down passes, the representation of node v is obtained by combining its self representation, the outside-subtree message from its parent, and the subtree-side messages from its children through an output function f_{out}:

$$\mathbf{H}_v = \mathbf{X}_v + \mathbf{M}_{\pi(v)\to v} + \sum\nolimits_{u\in\mathcal{C}(v)} \mathbf{M}_{u\to v}. \tag{3}$$

As shown in Figure 2 (a), the edge-message formulation provides a node-centric view of global tree aggregation. For a target node v, child-to-node messages recursively summarize the subtree-side context from $\mathrm{Sub}(v)$, while the parent-to-node message carries the outside-subtree context from $V \setminus \mathrm{Sub}(v)$. Thus, Eq. (3) combines the self feature, the context inside $\mathrm{Sub}(v)$, and the complementary context outside $\mathrm{Sub}(v)$, making the aggregation at v global rather than purely local. Figure 2 (b) shows how this global view is obtained through bidirectional propagation. The bottom-up pass in Eq. (1) propagates messages from leaves to root to build summaries over rooted subtrees, while the top-down pass in Eq. (2) propagates messages from root to leaves to distribute the complementary parent-side and sibling-branch context. Consequently, after the two passes, information between any pair of nodes u and v is routed along the unique tree path $P_T(u, v)$, moving upward to $\mathrm{LCA}(u, v)$ and then downward to v. This yields a linear-size global propagation backbone in which long-range dependency is established under tree-constrained routing rather than dense pairwise interaction. Compared with a shallow local MPNN whose dependency is restricted to the L-hop neighborhood, bidirectional tree aggregation can connect distant nodes through their unique tree path under non-degenerate transitions.

Proposition 3.1 (Long-range sensitivity)**.** *For an L-layer local MPNN, $\partial\mathbf{H}_v^{(L)}/\partial\mathbf{X}_u = 0$ if $d_G(u, v) > L$. For bidirectional tree aggregation on T, let $P_T(u, v)$ be the unique tree path from u to v. If the composed Jacobian along $P_T(u, v)$ is non-degenerate, then $\partial\mathbf{H}_v/\partial\mathbf{X}_u \neq 0$. Thus, a single bidirectional tree aggregation step can give nonzero sensitivity between any two nodes in the tree.*

Connectivity gap after tree routing. Tree routing provides global reachability, but an original edge $(u, v) \in E$ omitted from the tree becomes a residual edge in $E_R = E \setminus E_T$. Its direct one-hop interaction is then represented only through the potentially long tree path $P_T(u, v)$. This path dilation identifies the local connectivity that the correction module must restore.

This loss has an exact algebraic form. Fix an arbitrary orientation and let B_T and B_R be the incidence matrices of the tree and residual edges. Then

$$L_G = L_T + L_R = B_T^\top B_T + B_R^\top B_R, \qquad \mathcal{E}_G(H) = \mathcal{E}_T(H) + \tfrac{1}{2}\|B_R H\|_F^2, \tag{4}$$

where $\mathcal{E}_Q(H) = \frac{1}{2}\operatorname{tr}(H^\top L_Q H)$. Every residual edge $e = (u, v)$ closes one fundamental cycle $P_T(u, v) \cup \{e\}$, and these cycles form a basis of the cycle space. Residual edges therefore provide one coordinate per independent cycle and index exactly the term missing from the tree energy in Eq. (4).

4 Tree Residual Network

We propose Tree Residual Network (TRN), a propagation architecture that decomposes graph message passing into tree-based global routing and residual-edge correction. The spanning tree provides long-range communication, while the residual graph supplies the local constraints omitted by tree construction.

4.1 Tree Aggregation in TRN

TRN uses the bidirectional tree aggregation operator defined in Subsection 3.2 as its global routing backbone. Given an input signal $\mathbf{Z}$, the operator $T_\theta(\mathbf{Z}; T)$ propagates $\mathbf{Z}$ over the rooted spanning tree through one bottom-up pass and one top-down pass. Following Subsection 3.2, both the incoming-message aggregation f_{agg} and the output readout f_{out} are instantiated as summation. For each

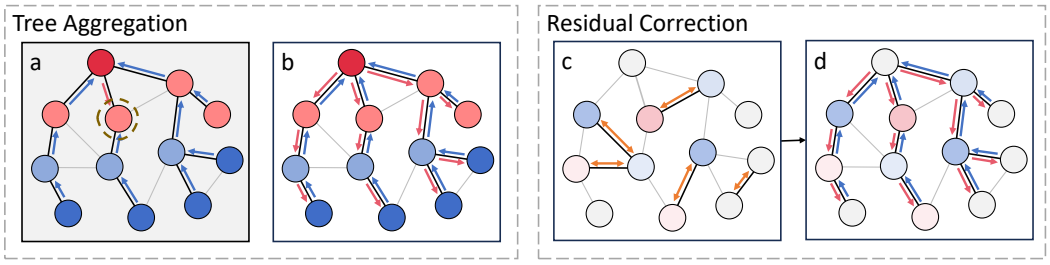

Figure 2: TRN computation. (a) For target v, child messages summarize its subtree and the parent message summarizes the complement. (b) Bottom-up and top-down passes route information along unique tree paths. (c) Each omitted edge computes a signed endpoint discrepancy. (d) The same fixed-gate tree operator redistributes that correction. Solid blue lines are tree edges; dashed red lines are residual edges.

directed tree message $u \to v$, let $\mathbf{s}_{u\to v}$ denote the source-side signal to be transmitted along this tree edge. We instantiate the directed tree transition as

$$\mathbf{M}_{u\to v} = \exp(-\lambda) \cdot g_T(\mathbf{Z}_u, \mathbf{Z}_v) \odot \frac{\mathbf{s}_{u\to v}\mathbf{W}_T}{\sqrt{d_T(u)d_T(v)}}, \tag{5}$$

where $g_T(\mathbf{Z}_u, \mathbf{Z}_v) = \mathrm{sigmoid}(\mathbf{Z}_u\mathbf{W}_s^T + \mathbf{Z}_v\mathbf{W}_t^T)$ is an endpoint-conditioned gate, $\mathbf{W}_T$ is a learnable tree-message transformation, and $\lambda \geq 0$ controls tree-distance decay. The gate makes tree propagation edge-adaptive, the decay factor controls long tree-path transmission, and the symmetric degree normalization stabilizes message magnitudes across nodes with different branching degrees.

With the input projection $\mathbf{H} = \mathrm{LayerNorm}\,(\mathrm{SiLU}(\mathbf{X}\mathbf{W}_{\mathrm{in}}))$, the initial tree-routed representation is

$$\mathbf{F}^{(0)} = T_\theta(\mathbf{H}; T). \tag{6}$$

For this initial route, we evaluate each endpoint gate once from the projected states, namely $g_T(\mathbf{H}_u, \mathbf{H}_v)$, and cache the resulting tree-edge gates. Every correction re-route reuses this cache, so correction rounds change the routed signal while preserving the routing rule. This step gives each node access to global tree context. However, the communication routes are still constrained by the spanning tree: an omitted original edge can only interact through its tree path. Therefore, non-tree edges are treated as residual structural information and corrected explicitly in the next module.

4.2 Residual Correction

After tree aggregation, each node has received global context through the tree backbone. Residual edges are then used to test whether this tree-routed context remains locally consistent with the original graph connectivity. Thus, a residual edge is not used as an ordinary aggregation edge that directly imports neighbor features. Instead, it acts as a diagnostic channel: if two original neighbors remain compatible after tree routing, only a small correction is needed; if their relation is distorted by the tree path, the edge produces a signed correction signal. As illustrated in Figure 2(c), residual correction estimates these local signals over the residual graph.

Given the current representation $\mathbf{F}^{(k-1)}$, we compute a signed correction field over residual edges:

$$\mathbf{C}_u^{(k)} = \sum_{v\in\mathcal{N}_R(u)} g_R(\mathbf{F}_v^{(k-1)}, \mathbf{F}_u^{(k-1)}) \odot \frac{\mathbf{F}_v^{(k-1)}\mathbf{W}_T - \mathbf{F}_u^{(k-1)}}{2\sqrt{d_R(u)d_R(v)}}. \tag{7}$$

Here, the residual gate has the same endpoint-conditioned form as the tree gate, $g_R(\mathbf{a}, \mathbf{b}) = \mathrm{sigmoid}(\mathbf{a}\mathbf{W}_{R,s}^T + \mathbf{b}\mathbf{W}_{R,t}^T)$, with learnable residual-gate parameters $\mathbf{W}_{R,s}$ and $\mathbf{W}_{R,t}$. The transformation $\mathbf{W}_T$ is shared with the tree transition in Eq. (5). Every undirected residual edge contributes in both orientations, and the factor 2 compensates this double counting. Therefore, residual correction is expressed in the same message space as tree propagation, instead of introducing a separate residual transformation. This provides a unified propagation mechanism and reduces the burden on the global tree route to implicitly absorb local connectivity errors. The residual gate further acts as an adaptive filter, retaining informative residual differences while attenuating unstable edge-wise oscillations before re-routing.

Residual-energy interpretation. Equation (7) is a learned, feature-wise generalization of residual-energy descent. To see this, set $\mathbf{W}_T = I$ and use a symmetric scalar weight w_e for both orientations of residual edge $e = (u, v)$. Its endpoint pair is proportional to $(\mathbf{F}_v - \mathbf{F}_u, \mathbf{F}_u - \mathbf{F}_v)$, and summing all pairs gives

$$\mathbf{C} = -\widetilde{L}_R\mathbf{F} = -\nabla_{\mathbf{F}} \tfrac{1}{2}\operatorname{tr}(\mathbf{F}^\top \widetilde{L}_R\mathbf{F}), \tag{8}$$

for the corresponding weighted, normalized residual Laplacian $\widetilde{L}_R$. Hence the sign is not an arbitrary alternative to aggregation: in the symmetric linear limit, it is precisely the direction that reduces disagreement on the graph constraints omitted by the tree. Learned gates and $\mathbf{W}_T$ relax this descent direction coordinate-wise.

The resulting $\mathbf{C}^{(k)}$ is not a replacement representation. It is a correction field that is re-routed through the same tree backbone, as shown in Figure 2(d):

$$\mathbf{F}^{(k)} = \mathbf{F}^{(k-1)} + T_\theta^{\text{fix}}(\mathbf{C}^{(k)}; T), \qquad k = 1, \ldots, K. \tag{9}$$

Here, T_θ^{fix} denotes the tree-routing operator with the same transition parameters and the cached endpoint gates evaluated from $\mathbf{H}$ in the initial route. During correction re-routing, these values are fixed rather than recomputed from the transient correction field. This keeps the routing rule tied to the initial projected graph signal: residual edges estimate local discrepancies, while the tree backbone redistributes these corrections globally.

Complementary analytical views. Equation (8) identifies the correction sign. Appendix A.2 analyzes an idealized balanced signed pair under the fixed-gate router, where shared routed components cancel exactly. The learned directional gates and transformation make this a reference mechanism rather than an exact cancellation guarantee for every learned update. The following proposition characterizes the corresponding graph-level correction in a quadratic surrogate.

Quadratic surrogate analysis. For an operator-level interpretation, consider the quadratic graph smoother $H_G = (I + \tau L_G)^{-1}X$. Let $A = I + \tau L_T$ and let C_R be a weighted residual incidence matrix with $L_R = C_R^\top C_R$. The tree-only solution $H_T = A^{-1}X$ provides the starting point. The following result characterizes the correction that recovers the full-graph smoother.

Proposition 4.1 (Quadratic-surrogate correction space and finite-iteration certificate)**.** *For every input X,*

$$H_G - H_T = -\tau A^{-1}C_R^\top \left(I + \tau C_R A^{-1} C_R^\top\right)^{-1} C_R H_T. \tag{10}$$

Moreover, for this fixed surrogate, the image of the linear map $X \mapsto H_G - H_T$ is exactly $\operatorname{range}(A^{-1}C_R^\top)$. *Define $\mu_{\max} = \lambda_{\max}(A^{-1/2}(\tau L_R)A^{-1/2})$ and $s_R = \tau \operatorname{tr}(C_R A^{-1} C_R^\top)$. Then*

$$\sup_{X \neq 0} \frac{\|H_T - H_G\|_A}{\|H_T\|_A} = \frac{\mu_{\max}}{1 + \mu_{\max}} \leq \frac{s_R}{1 + s_R}. \tag{11}$$

Finally, initialize $H^{(k+1)} = H^{(k)} + \eta A^{-1}[X - (A + \tau L_R)H^{(k)}]$ at $H^{(0)} = H_T$. With $\eta = 2/(2 + \mu_{\max})$,

$$\frac{\|H^{(K)} - H_G\|_A}{\|H_T\|_A} \leq \left(\frac{\mu_{\max}}{2 + \mu_{\max}}\right)^K \frac{\mu_{\max}}{1 + \mu_{\max}} \leq \left(\frac{s_R}{2 + s_R}\right)^K \frac{s_R}{1 + s_R}. \tag{12}$$

Interpretation and design implications. Equation (10) isolates the structural principle behind residual correction. In the quadratic reference model, an omitted edge measures an endpoint discrepancy through $C_R H_T$, $C_R^\top$ turns that discrepancy into signed node sources, and A^{-1} redistributes them through the tree. This decomposition explains why residual edges are used as correction coordinates rather than as a second global aggregation channel. The quantity s_R summarizes the resulting tree–residual mismatch: large values indicate a larger certified correction burden for the tree-only surrogate. TRN instantiates the same division of labor with learned feature-dependent gates and a fixed tree router: residual edges evaluate local discrepancies, while the tree distributes their corrections globally. We use the quadratic model as a principled reference for this architectural separation, rather than as a literal derivation of the learned update. Appendix A.7 clarifies the scope of the surrogate identity. Appendix A.2 examines cancellation for balanced pairs under fixed-gate routing. Appendix A.5 gives a conditional finite-depth stability bound for the learned correction stack. Appendix A.6 relates tree stretch to the correction burden in the quadratic reference model.

Table 1: Node classification results over homophilic graphs (%). Baseline entries are compiled from published results and reproduced runs (Appendix C). Best and second-best results in each column are in bold and underlined, respectively.

	Cora	CiteSeer	PubMed	Computer	Photo	CS	Physics
Metric	Accuracy	Accuracy	Accuracy	Accuracy	Accuracy	Accuracy	Accuracy
GCN	86.53 ± 1.61	75.97 ± 1.93	88.51 ± 0.28	89.65 ± 0.52	92.70 ± 0.20	92.92 ± 0.12	96.18 ± 0.07
GAT	86.53 ± 1.27	74.31 ± 1.25	87.42 ± 0.43	90.78 ± 0.13	93.87 ± 0.11	93.61 ± 0.14	96.17 ± 0.08
GraphSAGE	87.62 ± 1.73	74.79 ± 1.59	89.20 ± 0.53	91.20 ± 0.29	94.59 ± 0.14	93.91 ± 0.13	96.49 ± 0.06
GCNII	<u>87.93</u> ± 1.56	76.84 ± 0.91	89.21 ± 0.41	91.04 ± 0.41	94.30 ± 0.20	92.22 ± 0.14	95.97 ± 0.11
GraphGPS	87.32 ± 1.20	<u>77.01</u> ± 2.13	88.68 ± 0.46	91.19 ± 0.54	95.06 ± 0.13	93.93 ± 0.12	97.12 ± 0.19
NAGphormer	87.68 ± 1.80	76.21 ± 2.72	89.35 ± 0.20	91.22 ± 0.14	95.49 ± 0.11	<u>95.75</u> ± 0.09	<u>97.34</u> ± 0.03
Exphormer	87.03 ± 1.70	76.18 ± 1.50	88.55 ± 0.47	91.47 ± 0.17	95.35 ± 0.22	94.93 ± 0.01	96.89 ± 0.09
NodeFormer	86.25 ± 1.47	75.31 ± 2.11	85.25 ± 0.73	86.98 ± 0.62	93.46 ± 0.35	95.64 ± 0.22	96.45 ± 0.28
SGFormer	87.86 ± 1.12	75.85 ± 2.04	88.75 ± 0.41	91.99 ± 0.76	95.10 ± 0.47	94.78 ± 0.20	96.60 ± 0.18
Polynormer	87.83 ± 1.94	76.93 ± 2.16	<u>89.48</u> ± 0.43	**93.68** ± 0.21	<u>96.46</u> ± 0.26	95.53 ± 0.16	97.27 ± 0.08
TRN	**88.89** ± 0.87	**78.40** ± 0.95	**89.86** ± 0.36	<u>93.56</u> ± 0.13	**97.00** ± 0.12	**96.01** ± 0.08	**97.70** ± 0.05

4.3 Overall Framework

TRN first obtains an initial tree-routed representation and then performs K rounds of residual correction with fixed-gate tree re-routing, as defined above. Let $\mathbf{F}^{(K)}$ denote the final corrected representation. The node representation used for classification is computed by a lightweight local readout over the original graph:

$$\mathbf{Z}_v = \alpha \mathbf{F}_v^{(K)} + (1-\alpha)\,\mathrm{Mean}\left(\{\mathbf{F}_u^{(K)} : u \in \mathcal{N}_G(v)\}\right), \qquad \alpha \in [0,1]. \tag{13}$$

Both readout branches consume the processed states $\mathbf{F}^{(K)}$, and α controls the balance between the corrected node state and its local contextual readout. A linear classifier is applied to $\mathbf{Z}_v$.

Time complexity. Let d be the hidden dimension and h_T the maximum tree height. For fixed d, the initial route, K residual scans and re-routes, and final readout are linear in graph size. With dense $d \times d$ message transformations applied per directed message, the explicit work bound is $O((K+1)md^2)$ and edge-state storage is $O(md)$. One bidirectional tree route gives graph-wide reachability without dense all-pairs interaction; its level-wise schedule has an $O(h_T)$ dependency per route and an $O((K+1)h_T)$ routing critical path overall. Appendix H reports measured time.

5 Experiments

We evaluate predictive performance, the proposed decomposition, and practical efficiency on homophilic and heterophilic node-classification benchmarks. The implementation is available at https://anonymous.4open.science/r/TRN-4E41.

Experimental setup. Appendix B summarizes the 14 datasets used in our experiments, covering both homophilic and heterophilic node classification benchmarks. We compare TRN with representative GNN, heterophily-oriented GNN, and Graph Transformer baselines; details of the baselines are provided in Appendix C. For TRN, we construct a BFS spanning tree for each graph, with roots selected or sampled to favor shallow tree structures. The hyperparameter settings, implementation details, and hardware information are provided in Appendices E, F, and D, respectively. Additional analyses report wall-clock time and large-graph performance (Appendix H), memory (Appendix G), heterophilic sensitivity (Appendix I), tree robustness (Appendix J), and comparison with FGL (Appendix K).

Table 2: Node classification results on heterophilic graphs (%). Baseline entries are compiled from published results and reproduced runs (Appendix C). Best and second-best results in each column are in bold and underlined, respectively.

	Texas	Wisconsin	Cornell	Actor	Chameleon	Tolokers	Questions
Metric	Accuracy	Accuracy	Accuracy	Accuracy	Accuracy	ROC-AUC	ROC-AUC
GCN	54.05 ± 4.36	50.39 ± 7.55	53.78 ± 8.59	28.78 ± 1.48	41.31 ± 3.05	83.64 ± 0.67	76.09 ± 1.27
GAT	57.30 ± 3.38	54.31 ± 5.62	54.59 ± 7.33	28.99 ± 1.44	39.21 ± 3.08	83.70 ± 0.47	76.79 ± 0.71
GraphSAGE	82.43 ± 6.14	81.18 ± 5.56	75.95 ± 5.01	34.23 ± 0.99	37.77 ± 4.14	82.43 ± 0.44	76.44 ± 0.62
H2GCN	<u>84.86</u> ± 6.77	<u>86.67</u> ± 4.69	<u>82.16</u> ± 4.80	35.86 ± 1.03	26.75 ± 3.64	73.35 ± 1.01	63.59 ± 1.46
FAGCN	84.32 ± 6.02	83.33 ± 2.01	81.35 ± 5.05	36.18 ± 1.52	41.90 ± 2.72	77.75 ± 1.05	77.24 ± 1.26
DirGNN	76.25 ± 6.31	80.50 ± 5.50	76.51 ± 6.14	35.76 ± 1.68	42.94 ± 1.66	81.19 ± 1.05	76.13 ± 1.24
GraphGPS	74.88 ± 3.86	79.01 ± 2.92	72.37 ± 3.21	37.21 ± 1.17	40.79 ± 4.03	83.71 ± 0.48	71.73 ± 1.47
NAGphormer	64.44 ± 6.55	63.37 ± 6.32	58.89 ± 7.56	34.24 ± 0.93	40.36 ± 1.77	78.32 ± 0.95	75.13 ± 0.70
Exphormer	77.86 ± 3.13	70.03 ± 3.41	56.35 ± 4.31	35.77 ± 0.70	41.23 ± 3.16	83.53 ± 0.28	73.86 ± 0.58
NodeFormer	69.43 ± 3.01	74.21 ± 4.03	64.94 ± 4.78	35.66 ± 1.21	34.73 ± 4.14	78.10 ± 1.03	74.27 ± 1.46
SGFormer	76.68 ± 2.78	78.71 ± 2.89	73.07 ± 4.37	<u>37.53</u> ± 0.56	<u>44.93</u> ± 3.91	82.07 ± 1.18	72.15 ± 1.31
Polynormer	77.31 ± 3.64	72.93 ± 3.25	71.74 ± 4.33	36.42 ± 1.28	41.82 ± 3.45	**85.91** ± 0.74	**78.92** ± 0.89
TRN	**88.11** ± 5.43	**91.76** ± 2.88	**84.32** ± 3.15	**38.34** ± 1.12	**45.43** ± 2.73	<u>83.82</u> ± 0.69	<u>78.62</u> ± 0.82

Evaluation protocol. The controlled ablations provide the main component-level evidence under a shared protocol. Full baseline provenance and hyperparameter choices are provided in Appendices C and E, respectively.

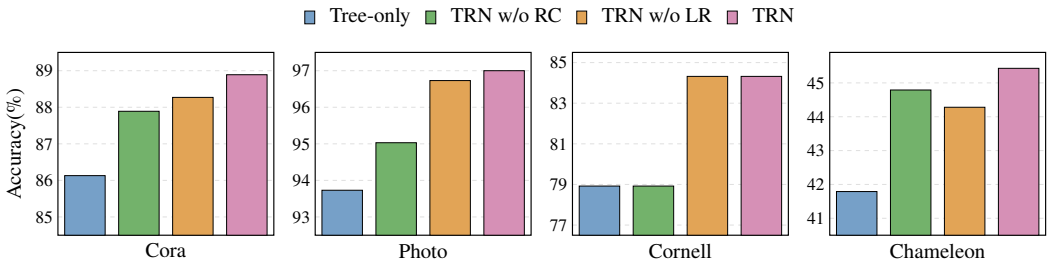

Figure 3: Ablation study of TRN on representative datasets. Each panel uses a clearly labeled, truncated dataset-specific axis to make small within-dataset differences legible; bar heights must not be compared across panels. TREE-ONLY uses only the tree backbone; TRN W/O RC removes residual correction; TRN W/O LR removes local readout; and TRN denotes the full model.

5.1 PERFORMANCE ON HOMOPHILIC AND HETEROPHILIC GRAPHS

Homophilic graphs. Table 1 reports the results on seven homophilic datasets. Across citation, co-purchase, and co-authorship graphs, TRN attains a highest or second-highest listed mean on every dataset and the highest listed mean on six of seven. The consistent pattern across graph families supports the transferability of the tree-routing and connectivity-restoration decomposition. On the three citation networks, TRN reaches 88.89 on Cora, 78.40 on CiteSeer, and 89.86 on PubMed. It also attains the highest listed mean on Photo (97.00), CS (96.01), and Physics (97.70), while remaining within 0.12 points of the highest listed mean on Computer. Thus, the observed strength is visible on both sparse citation graphs and denser co-purchase/co-authorship graphs rather than being concentrated in one benchmark family.

Heterophilic graphs. Table 2 reports the results on seven heterophilic datasets. Across the heterogeneous regimes, TRN attains a highest or second-highest listed mean on every dataset and the highest listed mean on five of seven. The ablations below identify the contribution of residual correction and

local readout under a shared protocol. The strongest listed means occur on the three WebKB graphs: 88.11 on Texas, 91.76 on Wisconsin, and 84.32 on Cornell. TRN also attains the highest listed mean on Actor and Chameleon, and remains second on the large heterophilic Tolokers and Questions benchmarks. On Texas and Cornell, where the selected model uses residual correction, these results complement the controlled component analysis; on Wisconsin and Actor, the validation-selected $K = 0$ setting instead highlights the reachability of the tree-routing backbone. Together, the results show that the decomposition adapts to heterogeneous local structure rather than requiring one fixed reliance on every component.

5.2 Ablation Analysis on TRN

The correction depth K and the mixing coefficient α are selected on validation. Figure 3 compares four configurations under a shared protocol: Tree-only retains only the initial bidirectional tree route; TRN w/o RC removes residual correction by setting $K = 0$ while retaining the final local readout; TRN w/o LR removes the local neighborhood readout while retaining correction; and TRN uses both modules. This construction isolates the contribution of restoring omitted-edge constraints and of the final local decoder relative to the same tree-routing backbone. The full configuration has the highest displayed mean on all four representative datasets. The largest component separation appears on Cornell, where removing correction changes accuracy from 84.3 to 78.8. On Photo, the complete decomposition raises the displayed mean from 93.7 for tree-only routing to 97.0. These controlled comparisons support the complementary roles of global tree routing, residual correction, and local readout; their uncertainty is reported with the full experimental results in the appendix.

5.3 Homophilic Sensitivity

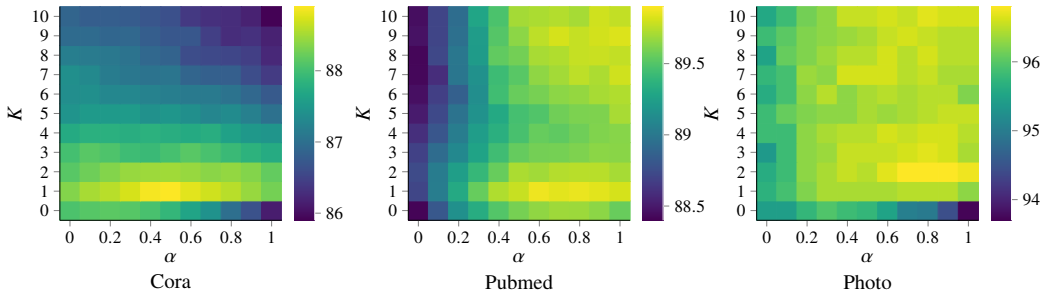

Figure 4: Sensitivity to correction order K and mixing coefficient α on homophilic graphs. Each panel uses its own labeled accuracy color scale to show within-dataset variation; colors are not comparable across datasets.

Figure 4 maps accuracy against the correction order K and the local-readout mixing coefficient α. The heatmaps show that the proposed decomposition is useful over regions of the hyperparameter space rather than at a single isolated setting. Cora favors a small correction order, whereas PubMed and Photo retain broad high-performing regions at moderate or larger K. The useful α range is also broad on PubMed and Photo, indicating that both the corrected tree state and local contextual readout can contribute without requiring a finely tuned balance. These patterns motivate validation-based selection of correction depth while demonstrating stable behavior on the larger homophilic graphs.

6 Conclusions

TRN combines a spanning-tree routing backbone with signed residual channels: the tree supplies global context, while omitted edges restore local constraints through the same routing coordinates. Complementary theoretical analyses provide operator-level perspectives on this cycle-completion mechanism. Across 14 benchmarks, TRN attains the highest or second-highest reported mean on every dataset; controlled ablations, a matched residual-aggregation control, and sensitivity analyses support the role of the proposed decomposition. TRN thus provides a reusable principle for combining sparse graph-wide transport with local structural fidelity.

REPRODUCIBILITY STATEMENT

Appendix B specifies the datasets and splits; Appendices E and F provide the search spaces, selected hyperparameters, tree construction, and forward algorithm; and Appendix D records the hardware. We report mean and standard deviation over ten runs, document baseline sources in Appendix C, and provide an anonymous implementation at https://anonymous.4open.science/r/TRN-4E41.

AI USE STATEMENT

Generative AI tools assisted with proof checking, language editing, and LaTeX formatting. They were not used to generate experimental data or results. the authors take responsibility for the final manuscript.

REFERENCES

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *Proceedings of the International Conference on Machine Learning*, 2019.

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *Proceedings of the International Conference on Learning Representations*, 2021.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Deyu Bo, Chuan Shi, Lele Wang, and Renjie Liao. Specformer: Spectral graph neural networks meet transformers. In *Proceedings of the International Conference on Learning Representations*, 2023.

Francesco Bonchi, Claudio Gentile, Francesco Paolo Nerini, André Panisson, and Fabio Vitale. Fast and effective gnn training through sequences of random path graphs. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, pp. 49–60, 2025.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *Proceedings of the International Conference on Learning Representations*, 2022.

Daniele Calandriello, Alessandro Lazaric, Ioannis Koutis, and Michal Valko. Improved large-scale graph learning through ridge spectral sparsification. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 688–697, 2018.

Benjamin Paul Chamberlain, James Rowbottom, Maria I. Gorinova, Michael M. Bronstein, Stefan Webb, and Emanuele Rossi. Grand: Graph neural diffusion. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1407–1418, 2021.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020a.

Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. Nagphormer: A tokenized graph transformer for node classification in large graphs. In *Proceedings of the International Conference on Learning Representations*, 2023.

Jinsong Chen, Siyu Jiang, and Kun He. Ntformer: A composite node tokenized graph transformer for node classification. *CoRR*, abs/2406.19249, 2024.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *Proceedings of the International Conference on Machine Learning*, 2020b.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive Universal Generalized PageRank Graph Neural Network. In *Proceedings of the International Conference on Learning Representations*, 2021.

Chenhui Deng, Zichao Yue, and Zhiru Zhang. Polynormer: Polynomial-expressive graph transformer in linear time. In *Proceedings of the International Conference on Learning Representations*, 2024.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *International Conference on Learning Representations*, 2022.

Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural networks for semi-supervised learning on graphs. *Advances in neural information processing systems*, 33:22092–22103, 2020.

Dongqi Fu, Zhigang Hua, Yan Xie, Jin Fang, Si Zhang, Kaan Sancak, Hao Wu, Andrey Malevich, Jingrui He, and Bo Long. Vcr-graphormer: A mini-batch graph transformer via virtual connections. In *Proceedings of the International Conference on Learning Representations*, 2024.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural Message Passing for Quantum Chemistry. In *Proceedings of the International Conference on Machine Learning*, 2017.

Benjamin Gutteridge, Xiaowen Dong, Michael M Bronstein, and Francesco Di Giovanni. Drew: Dynamically rewired message passing with delay. In *International Conference on Machine Learning*, pp. 12252–12267, 2023.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Asela Hevapathige, Asiri Wijesinghe, and Ahad N Zehmakan. Beyond fixed depth: Adaptive graph neural networks for node classification under varying homophily. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 21717–21725, 2026.

Zhongyu Huang, Yingheng Wang, Chaozhuo Li, and Huiguang He. Growing like a tree: Finding trunks from graph skeleton trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(5):2838–2851, 2024.

WANG Jiaqi, Xinyi Wu, James Cheng, and Yifei Wang. A signed graph approach to understanding and mitigating oversmoothing. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

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

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *Proceedings of the International Conference on Learning Representations*, 2019.

Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking Graph Transformers with Spectral Attention. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2021.

Jin Li, Shenghao Gao, Kaichen Zhang, Xinlong Chen, Ying Sun, and Hui Xiong. Forest-based graph learning for semi-supervised node classification. In *The Fourteenth International Conference on Learning Representations*, 2026.

Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. Finding global homophily in graph neural networks when meeting heterophily. In *Proceedings of the International Conference on Machine Learning*, 2022.

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2022.

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Classic gnns are strong baselines: Reassessing gnns for node classification. *Advances in Neural Information Processing Systems*, 37:97650–97669, 2024.

Jiahong Ma, Mingguo He, and Zhewei Wei. Polyformer: Scalable node-wise filters via polynomial graph transformer. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024.

Makoto Miwa and Mohit Bansal. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1105–1116, 2016.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2020.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *Proceedings of the International Conference on Learning Representations*, 2020.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *International Conference on Learning Representations*, 2023.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2022.

Emanuele Rossi, Bertrand Charpentier, Francesco Di Giovanni, Fabrizio Frasca, Stephan Günnemann, and Michael M Bronstein. Edge directionality improves learning on heterophilic graphs. In *Learning on graphs conference*, pp. 25–1. PMLR, 2024.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. In *International Conference on Machine Learning*, pp. 31613–31632, 2023.

Mohammad Shirzadi, Ali Safarpoor Dehkordi, and Ahad N Zehmakan. Adaptive initial residual connections for gnns with theoretical guarantees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 25392–25400, 2026.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Daniel A. Spielman and Jaeoh Woo. A note on preconditioning by low-stretch spanning trees. *CoRR*, abs/0903.2816, 2009.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1556–1566, 2015.

Jiachen Tang, Zhonghao Wang, Sirui Chen, Sheng Zhou, Jiawei Chen, and Jiajun Bu. Opengt: A comprehensive benchmark for graph transformers. *arXiv preprint arXiv:2506.04765*, 2025.

Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 807–816, 2009.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. Equivariant and stable positional encoding for more powerful graph neural networks. In *International Conference on Learning Representations*, 2022.

David Bruce Wilson. Generating random spanning trees more quickly than the cover time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 296–303, 1996.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying Graph Convolutional Networks. In *Proceedings of the International Conference on Machine Learning*, 2019.

Nan Wu and Chaofan Wang. Gtnet: A tree-based deep graph learning architecture. *arXiv preprint arXiv:2204.12802*, 2022.

Qitian Wu, Wentao Zhao, Zenan Li, David Wipf, and Junchi Yan. Nodeformer: A scalable graph structure learning transformer for node classification. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2022.

Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. Simplifying and empowering transformers for large-graph representations. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2023.

Yujie Xing, Xiao Wang, Bin Wu, Hai Huang, and Chuan Shi. Unifying and enhancing graph transformers via a hierarchical mask framework. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *Proceedings of the International conference on machine learning*, 2018.

Jie Yang, Yuwen Wang, Kaixuan Chen, Tongya Zheng, Yihe Zhou, Zhenbang Xiao, Ji Cao, Mingli Song, and Shunyu Liu. From gnns to trees: Multi-granular interpretability for graph neural networks. In *The Thirteenth International Conference on Learning Representations*, 2025.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021.

Chun-Yang Zhang, Wu-Peng Fang, Hai-Chun Cai, C. L. Philip Chen, and Yue-Na Lin. Sparse graph transformer with contrastive learning. *IEEE Trans. Comput. Soc. Syst.*, 11(1):892–904, 2024.

Xingxing Zhang, Liang Lu, and Mirella Lapata. Top-down tree long short-term memory networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 310–320, 2016.

Zaixi Zhang, Qi Liu, Qingyong Hu, and Chee-Kong Lee. Hierarchical Graph Transformer with Adaptive Node Sampling. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2022.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2020.

A Omitted Proofs

A.1 Proof of Proposition 3.1

Proof. For an L-layer local MPNN, let $\mathbf{h}_v^{(\ell)}$ be the representation of node v after layer ℓ. Each layer updates $\mathbf{h}_v^{(\ell)}$ using only $\mathbf{h}_v^{(\ell-1)}$ and the representations of one-hop neighbors in $\mathcal{N}_G(v)$. By induction on ℓ, $\mathbf{h}_v^{(\ell)}$ can depend only on input features within the ℓ-hop neighborhood of v. The base case $\ell = 0$ is immediate. If the claim holds for $\ell - 1$, then each neighbor representation used to update v at layer ℓ depends only on nodes within distance $\ell - 1$ from that neighbor, and hence within distance ℓ from v. Therefore, if $d_G(u, v) > L$, then $\mathbf{h}_v^{(L)}$ is independent of $\mathbf{x}_u$, which gives $\partial \mathbf{h}_v^{(L)} / \partial \mathbf{x}_u = 0$.

For bidirectional tree aggregation, consider two nodes u and v in the same tree and let $P_T(u, v) = (u = p_0, p_1, \ldots, p_q = v)$ be their unique tree path. The bottom-up and top-down passes route information from u to v along this path, moving upward to $\mathrm{LCA}(u, v)$ and then downward to v. Let $J_{p_i \to p_{i+1}}$ denote the local Jacobian of the directed transition along edge $p_i \to p_{i+1}$ with respect to the propagated signal. By the chain rule, the path-wise contribution from $\mathbf{x}_u$ to the output at v contains the composed Jacobian

$$J_{p_{q-1} \to p_q} J_{p_{q-2} \to p_{q-1}} \cdots J_{p_0 \to p_1} \frac{\partial \mathbf{s}_u}{\partial \mathbf{x}_u},$$

where $\mathbf{s}_u$ is the source-side signal initialized from $\mathbf{x}_u$. If this composed Jacobian is non-degenerate, then $\partial \mathbf{H}_v / \partial \mathbf{x}_u \neq 0$. Thus, unlike an L-layer local MPNN, whose sensitivity is restricted to the L-hop neighborhood, one bidirectional tree aggregation step can establish nonzero sensitivity between any two nodes in the same tree under non-degenerate path transitions. □

A.2 Balanced signed pairs under fixed-gate routing

The following proposition formalizes the signed-versus-unsigned distinction in an idealized balanced-pair setting. Because the learned residual gates are directional and the correction uses a learned transformation, practical endpoint pairs need not be exactly balanced; the result is therefore a reference case for interpretation rather than a guaranteed property of the learned model.

Proposition A.1 (Balanced-pair cancellation)**.** *Let $\mathcal{R}_{p \to x}$ be the fixed-gate route from p to x. For residual edge (u, v), route a signed pair $(\boldsymbol{\delta}, -\boldsymbol{\delta})$ and, as a comparator, an unsigned pair $(\boldsymbol{\delta}, \boldsymbol{\delta})$ through the same tree operator. Their effects at x are*

$$\Delta \mathbf{F}_x^{\pm} = (\mathcal{R}_{u \to x} - \mathcal{R}_{v \to x})\boldsymbol{\delta}, \qquad \Delta \mathbf{F}_x^{+} = (\mathcal{R}_{u \to x} + \mathcal{R}_{v \to x})\boldsymbol{\delta}. \tag{14}$$

Hence, if $\mathcal{R}_{u \to x} = \mathcal{R}_{v \to x} = \mathcal{R} \neq 0$, signed correction cancels exactly whereas unsigned aggregation propagates $2\mathcal{R}\boldsymbol{\delta}$.

Proof. Under fixed tree gates, tree re-routing is linear with respect to the correction field. Thus, for a source node p and a target node x, the routed contribution from p to x can be written as $R_{p \to x}\mathbf{c}_p$, where $R_{p \to x}$ is determined by the tree path, transition matrix, fixed gates, decay factors, and degree normalization terms.

Consider a residual edge (u, v) with a balanced signed correction pair $\mathbf{c}_u = \boldsymbol{\delta}$ and $\mathbf{c}_v = -\boldsymbol{\delta}$. By linearity and superposition, its total contribution to node x after tree re-routing is

$$\Delta \mathbf{F}_x = R_{u \to x}\boldsymbol{\delta} - R_{v \to x}\boldsymbol{\delta} = (R_{u \to x} - R_{v \to x})\boldsymbol{\delta}.$$

For the unsigned comparator, $\mathbf{c}_u = \mathbf{c}_v = \boldsymbol{\delta}$, so the same argument gives

$$\Delta \mathbf{F}_x^{+} = (R_{u \to x} + R_{v \to x})\boldsymbol{\delta}.$$

If $R_{u \to x} = R_{v \to x} = R$, the signed contribution is zero, whereas the unsigned contribution is $2R\boldsymbol{\delta}$. This proves Eq. (14) and the claimed strict separation whenever $R\boldsymbol{\delta} \neq 0$. □

A.3 Reference factorization and special cases

For a symmetric linear reference specialization of one correction round, let C_R be an oriented incidence matrix whose rows correspond to omitted edges, let D_R collect the learned edge-wise

gates and feature transformations, and let $\mathcal{R}_T$ denote the fixed tree re-routing operator. The reference correction has the factorized form

$$\Delta F = \mathcal{R}_T C_R^\top D_R C_R F. \tag{15}$$

The rightmost incidence map measures endpoint discrepancies, the transpose creates balanced signed sources, and only then does the tree operator redistribute them globally. This factorization motivates TRN's distinction from a second residual message-passing backbone: the omitted edges provide cycle coordinates, while the tree remains the sole global transport mechanism. The learned update in Eq. (7) uses directional gates and need not realize this factorization exactly; the reference form isolates the architectural ordering that it motivates.

Several nearby models arise by deleting one of these operations. Setting $K = 0$ gives Tree-only; omitting $\mathcal{R}_T$ gives local signed residual aggregation; replacing $C_R^\top D_R C_R$ by direct neighbor messages gives a gated residual MPNN such as TRN-GRA; and replacing one tree by multiple sampled trees gives the forest-style alternative represented by FGL. These are useful boundary cases, but none preserves the complete discrepancy–cycle–rerouting factorization in Eq. (15).

A.4 Proof of Proposition 4.1

Proof. Let $Q = A^{-1/2} C_R^\top$. Applying the Woodbury identity to $A + \tau C_R^\top C_R$ and right-multiplying by X gives Eq. (10). As a linear map of X, its correction operator is

$$A^{-1/2}\big[\tau Q(I + \tau Q^\top Q)^{-1} Q^\top\big] A^{-1/2}.$$

If $Q = U\Sigma V^\top$ is a compact singular value decomposition, the bracketed matrix has eigenvalues $\tau\sigma_i^2/(1+\tau\sigma_i^2) > 0$ on $\mathrm{range}(Q)$ and is zero on its orthogonal complement. Its range is therefore exactly $\mathrm{range}(Q)$, and multiplication by the invertible outer factors gives $\mathrm{range}(A^{-1}C_R^\top)$. This proves the stated image equality for the fixed quadratic surrogate.

Set $S = A^{-1/2}(\tau L_R)A^{-1/2}$ and $Y = A^{1/2}H_T = A^{-1/2}X$. Then

$$A^{1/2}H_G = (I+S)^{-1}Y, \qquad A^{1/2}(H_T - H_G) = S(I+S)^{-1}Y.$$

The supremum in Eq. (11) is therefore the spectral norm of $S(I+S)^{-1}$, namely $\mu_{\max}/(1+\mu_{\max})$. Moreover, $\mu_{\max} \le \mathrm{tr}(S) = s_R$, which gives the second inequality.

For the finite iteration, let $E^{(k)} = H^{(k)} - H_G$. Subtracting $(A + \tau L_R)H_G = X$ from the update and transforming by $A^{1/2}$ gives

$$A^{1/2}E^{(k+1)} = [I - \eta(I+S)]A^{1/2}E^{(k)}.$$

Since the eigenvalues of S lie in $[0, \mu_{\max}]$, the worst-case one-step contraction is

$$q(\eta) = \max_{\mu \in [0,\mu_{\max}]} |1 - \eta(1+\mu)|.$$

Balancing the two endpoint magnitudes yields $\eta^\star = 2/(2+\mu_{\max})$ and $q(\eta^\star) = \mu_{\max}/(2+\mu_{\max})$. Applying this contraction K times to $H^{(0)} = H_T$, invoking the tree-only bias bound, and using $\mu_{\max} \le s_R$ proves Eq. (12). □

A.5 Finite-Depth Perturbation Bound for the Correction Stack

The learned correction field is nonlinear, but the actual correction stack has a direct finite-depth perturbation bound once its tree-routing gates are fixed, as they are during correction re-routing.

Proposition A.2 (Conditional finite-depth stability)**.** *Let $\|F\|_{\infty,2} = \max_v \|F_v\|_2$. For correction round k, let $\mathcal{C}_k$ denote the nonlinear residual field in Eq. (7) and let $\mathcal{R}$ denote the fixed-gate tree re-routing operator. Suppose on a bounded state domain $\mathcal{D}$ that*

$$\|\mathcal{C}_k(F) - \mathcal{C}_k(\widehat{F})\|_{\infty,2} \le L_k \|F - \widehat{F}\|_{\infty,2}, \qquad \|\mathcal{R}(Y)\|_{\infty,2} \le \rho \|Y\|_{\infty,2}.$$

For two K-round correction trajectories that remain in $\mathcal{D}$ and use the same fixed routing gates,

$$\|F^{(K)} - \widehat{F}^{(K)}\|_{\infty,2} \le \prod_{k=1}^{K} (1 + \rho L_k) \|F^{(0)} - \widehat{F}^{(0)}\|_{\infty,2}. \tag{16}$$

The local mean readout is non-expansive in this norm, so the same bound holds for Z before the final linear classifier.

Proof. For one round, linearity of the fixed-gate router and the two assumed bounds give

$$\begin{aligned}\|F^{(k)} - \widehat{F}^{(k)}\|_{\infty,2} &\le \|F^{(k-1)} - \widehat{F}^{(k-1)}\|_{\infty,2} \\ &\quad + \rho\|\mathcal{C}_k(F^{(k-1)}) - \mathcal{C}_k(\widehat{F}^{(k-1)})\|_{\infty,2} \\ &\le (1+\rho L_k)\|F^{(k-1)} - \widehat{F}^{(k-1)}\|_{\infty,2}.\end{aligned}$$

Iteration proves Eq. (16). A neighborhood mean is a convex combination of node states and is therefore non-expansive under $\|\cdot\|_{\infty,2}$; the convex mixture in Eq. (13) preserves this property. □

On any bounded state domain, the constants L_k are finite because Eq. (7) composes affine maps, sigmoid gates, Hadamard products, and fixed degree normalizers; ρ is finite for the finite tree router. The result applies to the nonlinear residual fields actually used by TRN, but it is conditional on the routing gates shared by the two trajectories. If an input perturbation also changes those gates, the sensitivity of the gate-generating map must be added; Eq. (16) is therefore a finite-depth stability certificate for the correction stack, not a convergence theorem.

A.6 Effective Resistance as Residual-Correction Burden

Effective resistance and tree stretch are classical tools for graph sparsification and Laplacian preconditioning (Spielman & Woo, 2009; Calandriello et al., 2018). Here they provide a mechanism-level interpretation of the tree–residual decomposition rather than a new linear-system solver. Consider a weighted residual chord $e = (u, v)$ with conductance $w_e > 0$, oriented incidence vector $b_e = e_u - e_v$, and the regularized tree system $A = I + \tau L_T$. Define its regularized tree leverage score

$$\ell_e^{(\tau)} = \tau w_e \, b_e^\top A^{-1} b_e. \tag{17}$$

The factor $b_e^\top A^{-1} b_e$ is a ridge effective resistance on the tree. In particular,

$$\lim_{\tau\to\infty} \tau b_e^\top (I + \tau L_T)^{-1} b_e = b_e^\top L_T^\dagger b_e = R_T(u, v), \tag{18}$$

where $R_T(u, v)$ is the sum of edge resistances along the unique tree path from u to v; for an unweighted tree it equals $d_T(u, v)$. Thus the tree detour of a residual chord is the large-τ limit of its regularized correction burden.

Proposition A.3 (Exact response to one residual chord)**.** *Let $H_T = A^{-1}X$ be the tree-only quadratic smoother, and let $H_{T+e} = (A + \tau w_e b_e b_e^\top)^{-1}X$ be the smoother after restoring chord e. Then*

$$H_{T+e} - H_T = -\frac{\tau w_e}{1 + \ell_e^{(\tau)}} A^{-1} b_e \big(b_e^\top H_T\big). \tag{19}$$

Proof. Apply the Sherman–Morrison identity to $A + \tau w_e b_e b_e^\top$ and right-multiply by X. □

Equation (19) separates the exact surrogate response into three roles that have structural analogues in TRN: $b_e^\top H_T = H_{T,u} - H_{T,v}$ measures the endpoint discrepancy, $b_e(b_e^\top H_T)$ forms a balanced signed source pair, and A^{-1} redistributes that pair through the tree. The denominator $1 + \ell_e^{(\tau)}$ is the leverage-dependent normalization induced by restoring the chord, rather than an equal-strength additive message. Actual TRN replaces all of these linear objects by feature-wise gates, learned transformations, and gated tree routing, so the identity is not an equality for its learned update.

A.7 Exact Cycle-Space Correction in the Quadratic Surrogate

The one-chord identity extends jointly to all omitted edges through Eq. (10). Its four factors mirror the TRN computation: $C_R H_T$ measures endpoint discrepancies, the middle inverse couples discrepancies across fundamental cycles, $C_R^\top$ creates balanced signed node sources, and A^{-1} reroutes them through the tree. The range equality in Proposition 4.1 characterizes the image of this particular linear correction map across all inputs. It does not assert that increments of nonlinear TRN lie in the same range or that its residual parameterization is minimal, necessary, or optimal among neural architectures. The identity follows from a standard matrix identity and is not claimed as a new linear-system solver.

The certificate is one-sided: small regularized residual stretch guarantees that the tree-only surrogate is accurate, whereas a large value does not imply poor accuracy. It also controls correction depth only for the linear surrogate; it does not predict the validation-selected K of nonlinear TRN without an empirical calibration.

The same scores also explain the global mismatch coefficient in Proposition 4.1. Writing $B = \tau L_R = \sum_{e \in E_R} \tau w_e b_e b_e^\top$ gives

$$\max_{e \in E_R} \ell_e^{(\tau)} \;\le\; \mu_{\max} \;\le\; \sum_{e \in E_R} \ell_e^{(\tau)}. \tag{20}$$

Indeed, $A^{-1/2} B A^{-1/2}$ is the sum of positive-semidefinite rank-one matrices whose sole nonzero eigenvalues are $\ell_e^{(\tau)}$; its largest eigenvalue is at least the largest summand eigenvalue and at most its trace. Hence a single high-resistance chord lower-bounds the worst correction mode, while the total regularized residual stretch upper-bounds it in the linear surrogate. This suggests a tree-selection or residual-prior criterion, but the present experiments neither optimize this quantity nor test whether it predicts accuracy, robustness, or the validation-selected correction order K. We therefore do not use it to explain empirical differences among datasets or trees.

B Dataset Details

We evaluate TRN on 14 real-world node classification benchmarks, covering both homophilic and heterophilic graphs. Table 3 summarizes the dataset statistics, including the number of nodes, edges, input features, classes, and train/validation/test splits. The datasets include three citation networks, Cora, CiteSeer, and PubMed (Sen et al., 2008); two co-purchase networks, Computer and Photo (Shchur et al., 2018); two co-authorship networks, CS and Physics (Shchur et al., 2018); the WebKB benchmarks Texas, Wisconsin, and Cornell (Pei et al., 2020); Actor (Tang et al., 2009); the Wikipedia page-page network Chameleon (Rozemberczki et al., 2021); and two interaction-based heterophilic benchmarks, Tolokers and Questions (Platonov et al., 2023). For Chameleon, we follow the corrected split from the heterophilic graph benchmark (Platonov et al., 2023), since the conventional split may introduce train-test overlap; the corrected split filters out overlapping nodes and provides a more reliable evaluation. For Cora, CiteSeer, and PubMed, we use 50%, 25%, and 25% of nodes for training, validation, and testing, respectively, following (Xing et al., 2025). For the remaining datasets, we use the original splits provided by the corresponding benchmark.

Table 3: Overview of the datasets used for node classification.

Dataset	Nodes	Edges	Features	Classes	Train/Val/Test
Cora	2,708	5,278	1,433	7	1354/677/677
CiteSeer	3,327	4,522	3,703	6	1663/831/833
PubMed	19,717	44,324	500	3	9858/4929/4930
Computer	13,752	245,861	767	10	8251/2750/2751
Photo	7,650	119,081	745	8	4590/1530/1530
CS	18,333	81,894	6,805	15	10999/3666/3668
Physics	34,493	247,962	8,415	5	20695/6898/6900
Texas	183	309	1,703	5	87/59/37
Wisconsin	251	499	1,703	5	120/80/51
Cornell	183	285	1,703	5	87/59/37
Actor	7,600	33,544	931	7	3648/2432/1520
Chameleon	890	8,854	2,325	5	409/287/194
Tolokers	11,758	519,000	10	2	5879/2939/2940
Questions	48,921	153,540	301	2	24460/12230/12231

C Baseline Details

We compare TRN with representative GNN and Graph Transformer baselines. For homophilic datasets, we include GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), GraphSAGE (Hamil-

ton et al., 2017), and GCNII (Chen et al., 2020b) as classical message-passing baselines, and GraphGPS (Rampášek et al., 2022), NAGphormer (Chen et al., 2023), Exphormer (Shirzad et al., 2023), NodeFormer (Wu et al., 2022), SGFormer (Wu et al., 2023), and Polynormer (Deng et al., 2024) as Graph Transformer or scalable Transformer-style baselines. For heterophilic datasets, we compare with GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), GraphSAGE (Hamilton et al., 2017), and heterophily-oriented GNNs including H2GCN (Zhu et al., 2020), FAGCN (Bo et al., 2021), and DirGNN (Rossi et al., 2024), together with the same Graph Transformer baselines: GraphGPS (Rampášek et al., 2022), NAGphormer (Chen et al., 2023), Exphormer (Shirzad et al., 2023), NodeFormer (Wu et al., 2022), SGFormer (Wu et al., 2023), and Polynormer (Deng et al., 2024).

The baseline entries in Tables 1 and 2 are compiled primarily from prior studies (Deng et al., 2024; Ma et al., 2024; Luo et al., 2024; Hevapathige et al., 2026); missing entries are reproduced from official implementations or the OpenGT benchmark codebase (Tang et al., 2025) using the same data split as TRN. Mechanism-level claims rely on the TRN ablations and the TRN-GRA comparison, for which the split, tree, gates, normalization, and tuning protocol are held fixed.

D Hardware Information

We conduct all experiments on a Linux server equipped with two Intel Xeon Gold 6330 CPUs (56 physical cores in total, base frequency 2.00 GHz), 1.0 TiB RAM, and NVIDIA GeForce RTX 4090 GPUs (each with about 24 GB of memory). Each run uses a single RTX 4090 GPU under CUDA 12.4.

E Hyperparameter Settings

We tune the hyperparameters of TRN on the validation set of each dataset. The searched hyperparameters include the learning rate, weight decay, hidden dimension, dropout rate, mixing coefficient α, residual correction order K, and tree-distance decay coefficient λ. For the residual correction order, we search $K \in \{0, 1, 2, 3\}$ for the main experiments. Although some datasets may further benefit from larger K, we restrict the search range to keep the model efficient and avoid excessive residual re-routing. For the mixing coefficient, we perform a full grid search over $\alpha \in \{0, 0.1, 0.2, \ldots, 1.0\}$. For the tree-distance decay coefficient, we search $\lambda \in \{0, 0.3, 0.6\}$. The remaining hyperparameters are selected from common validation grids: learning rate in $\{0.001, 0.005, 0.01\}$, weight decay in $\{5 \times 10^{-7}, 5 \times 10^{-6}, 5 \times 10^{-5}, 5 \times 10^{-4}, 5 \times 10^{-3}, 5 \times 10^{-2}\}$, hidden dimension in $\{64, 128, 256\}$, and dropout in $\{0.2, 0.4, 0.5, 0.6, 0.7, 0.8\}$. Here, α controls the balance between the corrected tree-routed representation and the local readout, K denotes the number of residual correction rounds, and λ controls the exponential decay along tree propagation. All results are reported as mean $\pm$ standard deviation over 10 runs. We train TRN with Adam and select the best checkpoint according to validation performance. The selected hyperparameters are reported in Table 4.

The endpoint settings in Table 4 delimit the evidence supplied by individual datasets. On Wisconsin and Actor, $K = 0$ and $\alpha = 1$, so the selected model is tree routing followed by classification; these datasets therefore support the backbone rather than residual correction. On Computer and Chameleon, $\alpha = 0$ removes the direct $\mathbf{F}_v^{(K)}$ branch, but the neighborhood mean still aggregates states that have passed through tree routing and residual correction ($K = 2$ and $K = 1$, respectively).

F Model Implementation

Algorithm 1 summarizes the forward propagation of TRN. Before training, we construct and cache the BFS tree backbone and residual edge set. For each connected component, the default root is chosen by a two-sweep heuristic that approximates a minimum-height root: it finds two approximate diameter endpoints and selects the node minimizing the maximum distance to them. A BFS traversal from this root then defines the tree parent of each node by the first-visit rule, and nodes are cached by depth for bottom-up and top-down propagation. This shallow-tree construction is mainly used to make training faster: a smaller tree height reduces the number of sequential bottom-up and top-down level-wise propagation steps, while preserving a connected tree backbone for global routing.

Table 4: Selected hyperparameters of TRN on all datasets.

Dataset	Learning rate	Hidden dim.	Dropout	α	K	λ
Cora	0.001	256	0.8	0.5	1	0
CiteSeer	0.005	256	0.8	1	1	0.6
PubMed	0.005	256	0.7	0.6	1	0.3
Computer	0.01	128	0.6	0	2	0.3
Photo	0.005	128	0.4	0.7	2	0.3
CS	0.005	128	0.6	0.8	1	0.3
Physics	0.005	128	0.6	0.2	2	0
Texas	0.01	64	0.4	1	1	0.3
Wisconsin	0.01	128	0.6	1	0	0
Cornell	0.01	128	0.4	1	3	0
Actor	0.01	128	0.7	1	0	0.3
Chameleon	0.01	128	0.6	0	1	0
Tolokers	0.005	64	0.5	1	2	0.3
Questions	0.001	256	0.7	0.8	2	0

Algorithm 1 Forward propagation of TRN

Require: Precomputed rooted tree $T = (V, E_T)$, residual edge set E_R, node features X, correction order K, mixing coefficient α, decay coefficient λ
Ensure: Node logits Y
Project input features $H = \mathrm{LayerNorm}(\mathrm{SiLU}(XW_{\mathrm{in}}))$
Compute and cache $g_T(H_u, H_v)$ for every directed tree edge
Compute initial tree-routed representation $F^{(0)} = \textsc{TreeRoute}_\theta(H, T, g_T, \lambda)$
for $k = 1, \ldots, K$ **do**
    Compute signed residual correction field $C^{(k)}$ over E_R by Eq. (7)
    Re-route correction $\Delta^{(k)} = \mathrm{TreeRoute}_\theta^{\mathrm{fix}}(C^{(k)}, T, g_T, \lambda)$
    Update $F^{(k)} = F^{(k-1)} + \Delta^{(k)}$
end for
Compute local readout Z by Eq. (13)
Compute logits $Y = ZW_{\mathrm{out}}$
return Y

The model then performs one bidirectional tree aggregation to obtain the initial tree-routed representation. The subroutine $\mathrm{TreeRoute}_\theta$ denotes the bidirectional tree aggregation operator used in the main text. It performs a bottom-up pass from leaves to roots and a top-down pass from roots to leaves, using the directed transition in Eq. (5). Residual correction is applied for K rounds, where each round estimates signed discrepancies over residual edges and redistributes the correction field through the same tree backbone with fixed routing gates. The fixed version $\mathrm{TreeRoute}_\theta^{\mathrm{fix}}$ reuses the tree gates computed from the projected input states H before the initial route and does not recompute gates from the transient correction field or corrected states. Finally, a lightweight local readout over the original graph is used for prediction.

G Memory Usage

We further report the GPU memory usage of TRN and representative baselines on Cora, CiteSeer, and Pubmed. The memory consumption is measured by the peak GPU memory allocated during training under the same experimental setting as the main experiments. The results are summarized in Table 5.

As shown in Table 5, TRN maintains moderate memory usage across the three datasets. Compared with GraphGPS, TRN requires substantially less memory, especially on Pubmed, where GraphGPS consumes 5.79 GiB while TRN uses 1171.3 MiB. This is consistent with the sparse propagation design of TRN: messages are mainly propagated over a tree backbone, while omitted graph edges are handled through lightweight residual correction. Therefore, TRN avoids the expensive dense attention operations and structural encodings commonly used in graph transformer models.

Table 5: GPU memory usage on Cora, CiteSeer, and Pubmed.

Model	Cora	CiteSeer	Pubmed
TRN	178.9 MiB	217.4 MiB	1171.3 MiB
GCNII	198.9 MiB	289.7 MiB	1075.1 MiB
GraphGPS	578.7 MiB	852.4 MiB	5.79 GiB

Compared with GCNII, TRN remains in the same memory range. It uses less memory on Cora and CiteSeer, while being slightly higher on Pubmed. This increase is reasonable because TRN maintains additional tree propagation states and residual correction signals. Overall, these results suggest that TRN achieves competitive memory efficiency among standard GNN baselines and is much more memory-efficient than transformer-style graph models.

H Runtime and Large-Graph Evaluation

Table 6 reports single-epoch training and full-graph inference time under a common hidden dimension of 128; TRN uses $K = 1$. Timings were measured on the hardware in Appendix D after warm-up. They characterize these implementations, not hardware-independent algorithm rankings.

Table 6: Wall-clock time in seconds (training/inference). OOM denotes out of memory under the evaluated standard configuration.

Model	Cora	PubMed	ogbn-arxiv
TRN	0.0154/0.0047	0.0196/0.0046	0.1230/0.0233
GCNII	0.0296/0.0113	0.0385/0.0177	0.5183/0.2406
GraphGPS	0.0316/0.0106	0.4786/0.1374	OOM
Exphormer	0.0132/0.0046	0.0727/0.0298	OOM

On ogbn-arxiv, TRN obtains 72.31 ± 0.19 test accuracy, compared with 71.72 ± 0.32 for a reproduced GCN and 71.49 ± 0.27 for the standardized GraphSAGE reference result. This experiment establishes that the implementation runs at this scale; specialized deep GNN systems report higher accuracy. The level-wise bottom-up and top-down schedule also remains less parallel than conventional sparse message passing, despite depth-wise batching and a custom autograd path.

As an additional attention baseline under the standard citation-network protocol, GATv2 (Brody et al., 2022) obtains 87.43 ± 1.53, 75.62 ± 1.19, and 88.21 ± 0.42 on Cora, CiteSeer, and PubMed, respectively, while TRN obtains 88.89 ± 0.87, 78.40 ± 0.95, and 89.86 ± 0.36. We report the full uncertainty rather than interpreting the differences in means alone as significance tests.

I Sensitivity under Heterophily

Figure 5 reports the sensitivity of TRN to the routing order K and the mixing coefficient α on Texas, Cornell, and Chameleon. Compared with the homophilic datasets in the main text, the heterophilic datasets exhibit more dataset-dependent patterns. On Texas and Cornell, performance generally improves when α becomes large, indicating that preserving the corrected tree-routed representation is important on these small heterophilic graphs. This suggests that excessive reliance on local readout may reintroduce noisy one-hop neighborhood information, while the tree-residual representation provides a more useful structural signal.

The effect of the correction order K also varies across datasets. Texas achieves strong performance with a small correction order, while Cornell benefits from larger K in some regions, suggesting that repeated residual correction can be useful when heterophilic relations require higher-order interactions between residual edges and tree paths. Chameleon shows a different pattern: larger K and larger α tend to give better performance, which indicates that deeper residual re-routing can help exploit non-local structural information when local neighborhoods are less label-consistent. Overall, these

results show that residual correction remains important on heterophilic graphs, but the optimal routing depth and mixing coefficient depend on the dataset.

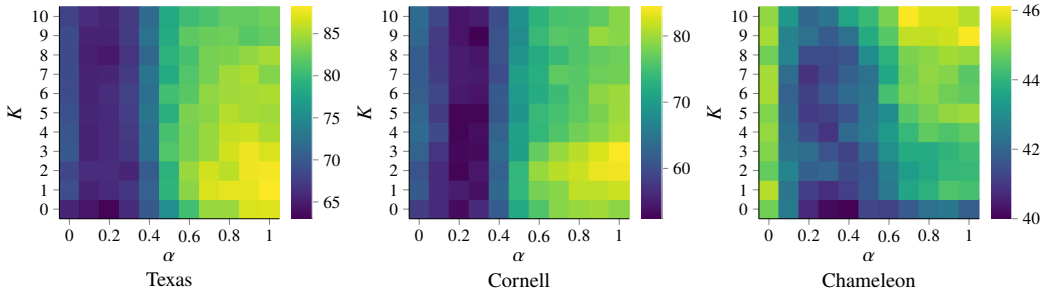

Figure 5: Sensitivity to K and α on heterophilic graphs. Each panel uses its own labeled accuracy color scale; colors are not numerically comparable across panels.

J Performance under Different Tree Samples

To evaluate the robustness of TRN to different tree backbones, we compare the default *Shallow-BFS* construction with three alternative tree sampling strategies. *Shallow-BFS* is the default strategy used in the main experiments, where an approximate minimum-height root is selected before BFS construction. *Random-BFS* constructs a BFS tree from a randomly selected root. *Random-Wilson* samples a spanning tree using Wilson's algorithm (Wilson, 1996) from a random starting node. *Similarity-Wilson* further uses similarity-weighted transition probabilities during Wilson's random walk. For a fair robustness test, all alternative tree sampling strategies use the optimal hyperparameters selected under *Shallow-BFS*, without additional hyperparameter tuning.

Table 7: Performance of TRN under different tree construction strategies. All variants use the hyperparameters selected under the default *Shallow-BFS* setting, without additional tuning.

Tree strategy	Cora	CiteSeer	PubMed	Texas	Chameleon
Shallow-BFS	$88.89_{\pm 0.87}$	$78.40_{\pm 0.95}$	$89.86_{\pm 0.36}$	$88.11_{\pm 5.43}$	$45.43_{\pm 2.73}$
Random-BFS	$88.33_{\pm 1.02}$	$78.34_{\pm 1.01}$	$89.84_{\pm 0.35}$	$87.22_{\pm 8.50}$	$45.55_{\pm 3.41}$
Random-Wilson	$88.21_{\pm 0.85}$	$78.16_{\pm 0.75}$	$89.95_{\pm 0.38}$	$87.03_{\pm 4.80}$	$45.98_{\pm 2.20}$
Similarity-Wilson	$88.17_{\pm 0.87}$	$78.28_{\pm 1.02}$	$89.92_{\pm 0.41}$	$87.30_{\pm 5.41}$	$45.70_{\pm 2.76}$

Table 7 reports the results on Cora, CiteSeer, PubMed, Texas, and Chameleon. Overall, TRN remains stable under different tree construction strategies. On homophilic datasets, the performance of alternative trees is close to the default *Shallow-BFS* result; for example, Random-BFS achieves 88.33% on Cora and 78.34% on CiteSeer, both close to the default results. On PubMed, Wilson-based trees even slightly outperform the default tree, suggesting that TRN does not rely on one specific deterministic tree sample. On heterophilic datasets, the performance variation is larger but still moderate. For Texas, Wilson-based trees remain competitive, while Random-BFS has higher variance, indicating that root choice can matter more on small heterophilic graphs. On Chameleon, all alternative strategies are close to or slightly better than the default result. These results are consistent with residual correction compensating for tree-specific omissions by restoring non-tree connectivity from the original graph. Thus, while the sampled tree affects the global routing path, TRN is not overly dependent on a single tree backbone.

K Comparison with Forest-Based Graph Learning

FGL (Li et al., 2026) is closely related to TRN because both methods exploit tree or forest structures for graph representation learning. We include the Texas, Wisconsin, and Cornell results reported in the original FGL paper.

Table 8: Comparison with FGL on heterophilic datasets.

Model	Texas	Wisconsin	Cornell
FGL†	$91.89_{\pm 0.00}$	$86.27_{\pm 0.00}$	$83.24_{\pm 2.02}$
TRN	$88.11_{\pm 5.43}$	$91.76_{\pm 2.88}$	$84.32_{\pm 3.15}$

† Results are taken from the original FGL paper.

As shown in Table 8, TRN achieves competitive reported means compared with FGL on the three heterophilic datasets. TRN has a higher reported mean on Wisconsin and Cornell, while FGL reports a higher result on Texas. It is worth noting that FGL and TRN differ in how they recover the information lost after tree extraction. FGL reduces this loss by sampling and aggregating multiple trees, and further introduces a homophily estimator and a local encoder. TRN instead uses a single BFS tree as the propagation backbone and, when $K > 0$, applies residual correction over omitted non-tree edges. Thus, TRN does not rely on multi-tree aggregation, homophily estimation, or an additional local encoder. Because the entries follow different experimental pipelines, this comparison is descriptive rather than controlled. Moreover, the selected Wisconsin configuration uses $K = 0$, so that result supports the single-tree backbone rather than residual correction. The table nevertheless situates the reported performance of the single-tree TRN design relative to forest aggregation on these heterophilic benchmarks. The tree-sampling analysis in Appendix J additionally indicates moderate robustness across the evaluated tree structures.

L Residual Correction versus Gated Residual-Edge Aggregation

To further examine whether the proposed residual correction can be reduced to ordinary aggregation over the residual graph, we compare TRN with a gated residual-edge aggregation variant. We denote this variant as TRN-GRA. The tree backbone is kept unchanged, while the residual correction module is replaced by a direct gated aggregation operator over the omitted non-tree edges:

$$\mathbf{C}_u^{(k)} = \sum_{v \in \mathcal{N}_R(u)} g_R(\mathbf{F}_v^{(k-1)}, \mathbf{F}_u^{(k-1)}) \odot \frac{\mathbf{F}_v^{(k-1)} \mathbf{W}_T}{\sqrt{d_R(u) d_R(v)}}. \tag{21}$$

Here, g_R is the same endpoint-conditioned residual gate used in TRN, and $\mathbf{W}_T$ is the shared message transformation. Thus, TRN-GRA has access to the same residual graph and uses the same gating and normalization design. However, unlike the proposed residual correction, it directly aggregates transformed neighbor representations from residual edges and does not compute signed discrepancy signals between residual-edge endpoints. This comparison controls the tree backbone, residual graph, gates, normalization, and shared transformation while changing the complete residual channel. It therefore tests whether omitted edges are more useful as discrepancy-and-rerouting channels than as direct residual messages. It is not a factorial ablation of the sign and the re-routing operator separately; the analytical separation is given in Appendix A.2, and the architectural factorization is given in Appendix A.3. If the benefit of TRN came only from adding omitted edges back into the computation, the gated aggregation variant should perform similarly. A higher TRN result instead supports using omitted edges as correction channels for tree-routed representations.

Table 9 reports a higher mean for TRN than TRN-GRA on every evaluated dataset, including both homophilic and heterophilic settings. The gap is small on PubMed and CS, showing that direct gated aggregation can recover part of the omitted local connectivity; the larger separation on Cornell is consistent with the value of discrepancy-based correction. In the symmetric reference specialization, a balanced signed pair is selectively attenuated along shared tree routes, whereas direct aggregation uniformly imports neighbor representations. Learned directional gates and transformations need not preserve this cancellation exactly, so this reference behavior motivates, but does not by itself prove, the observed gains. Overall, the controlled comparison supports using residual edges as signed correction channels rather than ordinary message-passing edges.

Table 9: Comparison between the proposed residual correction and gated residual-edge aggregation. TRN-GRA replaces the signed residual correction with direct gated aggregation over residual edges.

Dataset	TRN-GRA	TRN
Cora	$88.36_{\pm 0.94}$	$88.89_{\pm 0.87}$
CiteSeer	$77.86_{\pm 0.73}$	$78.40_{\pm 0.95}$
Pubmed	$89.85_{\pm 0.32}$	$89.86_{\pm 0.36}$
Computer	$93.31_{\pm 0.21}$	$93.56_{\pm 0.13}$
Photo	$96.47_{\pm 0.17}$	$97.00_{\pm 0.12}$
CS	$95.88_{\pm 0.13}$	$96.01_{\pm 0.08}$
Physics	$97.59_{\pm 0.08}$	$97.70_{\pm 0.05}$
Texas	$86.76_{\pm 7.40}$	$88.11_{\pm 5.43}$
Cornell	$70.54_{\pm 5.33}$	$84.32_{\pm 3.15}$
Chameleon	$44.78_{\pm 2.93}$	$45.43_{\pm 2.73}$
Questions	$77.42_{\pm 1.21}$	$78.62_{\pm 0.82}$
Tolokers	$83.05_{\pm 0.84}$	$83.82_{\pm 0.69}$

M Limitations and Future Work

The theory analyzes symmetric or fixed-gate reference settings and a separate quadratic surrogate, whereas TRN uses learned gates, nonlinear transformations, normalization, and finite depth. These analyses are therefore operator-level perspectives, not convergence or exact-recovery theorems for nonlinear TRN. Wisconsin and Actor use $K = 0$ and support the tree backbone rather than correction; Computer and Chameleon use a local-mean decoder. The controlled ablations and TRN-GRA comparison provide the direct component evidence where correction is active, while main-table baselines compile published sources and reproduced runs.

TRN uses a fixed shallow-BFS tree. The sampling study indicates moderate robustness but does not identify a task-aware tree, and the effective-resistance quantity is not validated as a predictor of accuracy or correction depth. Finally, for fixed hidden dimension the method is linear in graph size, but level-wise routing has an $O((K+1)h_T)$ critical path and less parallelism than optimized local message passing. Parallel routing, adaptive correction depth, and task-aware tree construction remain open directions.