RECURSIVE MESSAGE PASSING IS TOPOLOGY-ADAPTIVE

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph neural networks perform message passing through a fixed sequence of layers and thus couple parameterization, receptive field, and computation. We identify this *coupling problem* as a central structural limitation of message passing: no single depth L can simultaneously avoid over-smoothing dense, low-diameter graphs while reaching across elongated or weakly connected ones. We argue that input-adaptive depth is therefore a structural requirement of the message-passing paradigm, and introduce Tiny Recursive Graph Neural Networks (TRGNNs) to realize it. TRGNNs combine *recursive message passing*, which decouples effective depth from parameterization, with *iterative prediction refinement*, which refines node predictions at each recursive step so that effective depth emerges from a learned function of the downstream accuracy, where predictions stabilize as effective depth increases, rather than a predetermined layer count or other external halting signal. We show empirically that these mechanisms yield *topology-adaptive effective depth*. Moreover, we establish a restricted formal connection between the rate at which latent representations approach a fixed point and the spectral quantity governing over-smoothing. Empirically, TRGNNs match or exceed GNN performance while using far fewer parameters and support anytime prediction.

1 ON THE INELASTICITY OF MESSAGE PASSING

Message-passing–based graph neural networks (GNNs) (Gilmer et al., 2017) have long been a *de facto* approach to representation learning over relational data. GNNs enforce locality and permutation invariance, two inductive biases aligned with the relational structure of graphs, which in turn underpins their historically strong empirical performance across a wide range of graph learning tasks.

Despite these favorable biases, GNNs often exhibit limited performance gains under increased depth scaling (Pengmei et al., 2025; Sypetkowski et al., 2024; Zhao et al., 2021). Whereas transformers, particularly in the language domain, have shown predictable improvements under parameter scaling (Hoffmann et al., 2022; Kaplan et al., 2020) and, more recently, increased test-time computation (Snell et al., 2024; Muennighoff et al., 2025), GNNs show diminishing returns in such regimes.

This limitation has challenged the hegemony of message-passing architectures in graph learning, with recently proposed graph transformer-based models (Kim et al., 2021; 2022; Kreuzer et al., 2021; Mialon et al., 2021) achieving strong performance despite replacing topology-constrained aggregation with dense attention over the full node set, thereby relaxing the locality and sparsity biases that motivate message-passing. While this scaling difficulty is commonly attributed to over-smoothing (Li et al., 2018; Oono & Suzuki, 2020), we argue that it is indicative of a deeper structural limitation of the message-passing paradigm of which over-smoothing is but one symptom:

The Coupling Problem. Consider a standard L-layer message-passing GNN with layer-wise parameters $\{\theta^{(\ell)}\}_{\ell=1}^{L}$. Depth L simultaneously governs three quantities that are, in principle, independent:

- **Parameterization.** Each layer introduces a distinct set of learnable weights, and thus the total parameter count scales as $O(L)$. Increasing the receptive field necessarily enlarges model capacity.
- **Receptive field.** Latent representation $h_v^{(L)}$ of node v depends only on the subgraph induced by its L-hop neighborhood $\mathcal{N}^L(v)$. Capturing dependencies at distance k requires $L \geq k$.
- **Computation.** The forward pass requires L sequential aggregation steps; both training and inference cost are fixed at design time and applied uniformly to all inputs.

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.

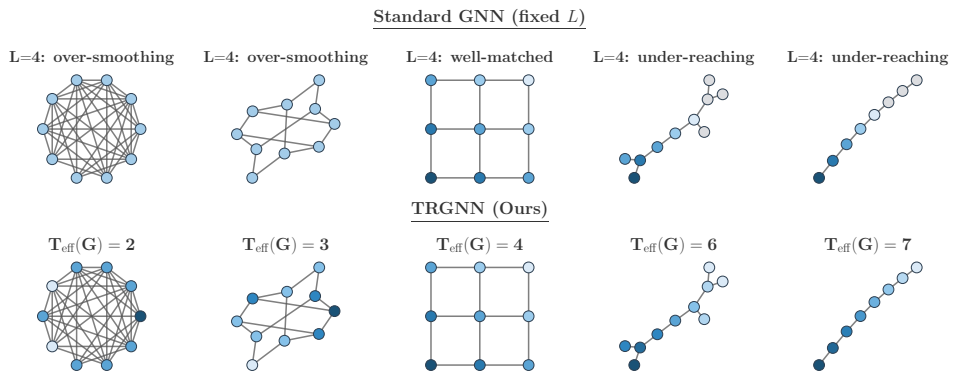

Figure 1: Topology-dependent propagation across graphs of increasing diameter (δ). **Top:** a fixed-depth GNN ($L = 4$) applied uniformly. Node shading encodes proximity to a source node (dark blue); grey denotes unreached nodes, uniform shading denotes over-smoothed representations. **Bottom:** a TRGNN adapts its effective depth $T(G)$, matching propagation to topology at fixed parameterization.

This coupling is not unique to GNNs; in any feedforward architecture, depth simultaneously governs capacity and computation. What makes this phenomenon uniquely problematic for graph learning is that the *communication structure varies across inputs*. In domains such as vision and language, inputs arise from fixed underlying topologies (e.g., grids or sequences), rendering a fixed depth a reasonable design choice. Graphs exhibit no such uniformity: within a single dataset, graph connectivity profiles may vary arbitrarily, inducing distinct propagation regimes (Figure 1). Consequently, any fixed depth L induces a mismatch: it over-smooths dense, low-diameter graphs while under-reaching in elongated or weakly connected ones, implying that no single choice of L can be uniformly appropriate across such structural heterogeneity. Section 2.1 formalizes this as the *elasticity window*, the range of depths in which a graph can be processed without over-smoothing or under-reaching; its location and width vary with graph topology, so no fixed L can lie in every graph's window simultaneously.

Graph transformer architectures sidestep this mismatch by discarding topology-constrained message passing altogether, permitting arbitrary node pairs to attend to each other regardless of geodesic distance. This removes the receptive field bottleneck, but at a cost: $O(N^2)$ complexity and the loss of the locality and sparsity biases that underlie the effectiveness of message passing. From this perspective, their empirical success stems not from resolving the coupling problem, but from bypassing it entirely. Consequently, we posit that if message passing is to remain a viable and competitive paradigm for graph learning, *the inelasticity of message passing* must be addressed directly by decoupling depth from propagation while preserving its structural inductive biases.

Decoupling via Recursion. Recursive architectures in the transformer setting have recently received considerable attention for reducing parameter count (Giannou et al., 2023; Jeddi et al., 2026; Jolicoeur-Martineau, 2025; Kohli et al., 2026; Merrill & Sabharwal, 2025). *The key insight of this manuscript is that on graphs, recursion does something more fundamental: it creates input-adaptive depth that matches the topological heterogeneity of graph data.* We therefore propose recursive message passing as a principled resolution to the coupling problem. This constitutes a precondition for topology-adaptivity: once depth is a free parameter, it can target each graph's elasticity window.

Adaptivity via Iterative Refinement. Decoupling is a necessary condition for topology-adaptivity. We now address sufficiency. Motivated by classical iterative algorithms on graphs (Pearl, 1982; 2014; Zhu & Ghahramani, 2002), we combine recursion with iterative refinement of predictions, so that adaptive behavior emerges from the learned dynamics without requiring an external halting signal.

We introduce *Tiny Recursive Graph Neural Networks (TRGNNs)*[1], which combine recursive message passing with iterative refinement of a persistent prediction. TRGNNs maintain a coupled state $(H^{(t)}, y^{(t)})$, a latent node representation and a prediction that are jointly updated across iterations.

[1]The name *Tiny Recursive GNN* is chosen both to emphasize that a small shared operator, applied recursively, suffices to match the representational reach of substantially larger models and *en hommage* to the Tiny Recursive Model of Jolicoeur-Martineau (2025), which studies analogous concepts in the reasoning model domain.

Because its prediction head receives both $H^{(t)}$ and $y^{(t-1)}$, it can learn to produce $y^{(t)} \approx y^{(t-1)}$ once $H^{(t)}$ over-smooths, so that additional iteration need not degrade the output.

As illustrated in Figure 1, this enables TRGNNs to adapt to the structural properties of each graph: dense, low-diameter graphs converge in few iterations, while elongated or weakly connected graphs induce deeper computation. In this way, *TRGNNs restore elasticity to message passing* while preserving its locality and sparsity biases. We demonstrate this adaptivity empirically, characterize the conditions under which it is well-behaved, and evaluate its consequences in the remainder of the manuscript. Our contributions are as follows:

- We identify the *coupling problem* in message passing—the mismatch between fixed-depth computation and input-dependent topology—and formalize it via the *elasticity window*, showing that no fixed depth is uniformly admissible across a topologically heterogeneous dataset.
- We show empirically that recursive message passing with iterative prediction refinement yields *topology-adaptive effective depth*: convergence rates track spectral properties of the input graph across controlled experiments, and a simple stopping rule allocates computation according to graph topology (Section 4, Appendix G).
- We complement this with a restricted formal result showing that, under normalized-adjacency and subspace-invariance assumptions, this topology dependence is derivable for the component of the recurrence governing over-smoothing (Section 4, Appendix E).
- We instantiate this in *Tiny Recursive Graph Neural Networks (TRGNNs)*, which decouple effective depth from parameterization and yield anytime inference as a direct consequence (Section 3).
- TRGNNs match or exceed standard GNNs on evaluated benchmarks while using far fewer parameters, with predictions improving monotonically under iteration (Section 5).

2 Background and Prior Work

Preliminaries. Let $G = (V, E)$ denote a graph with node set V, $|V| = N$, and edge set $E \subseteq V \times V$. We consider graphs that are (optionally) endowed with signals on V and E. That is, each node $v \in V$ carries a feature vector $x_v \in \mathbb{R}^d$, collected as rows of a feature matrix $X \in \mathbb{R}^{N \times d}$; edges $(u, v) \in E$ may carry attributes $e_{uv} \in \mathbb{R}^{d_e}$. We write $\mathrm{d}_G(u, v)$ for the geodesic (shortest-path) distance, $\mathcal{N}^k(v) = \{u \in V : \mathrm{d}_G(u, v) \leq k\}$ for the k-hop neighborhood of node v and $\mathcal{N}(v) := \mathcal{N}^1(v)$. We denote the adjacency matrix by $A \in \{0, 1\}^{N \times N}$ and the symmetric normalized adjacency by $\hat{A}$.

A message-passing neural network (MPNN) (Gilmer et al., 2017) computes node representations through L successive layers. At layer $\ell = 1, \ldots, L$, each node v aggregates information from its neighborhood and updates its representation:

$$h_v^{(\ell)} = \psi^{(\ell)}\Big(h_v^{(\ell-1)}, \bigoplus_{u \in \mathcal{N}(v)} \phi^{(\ell)}\big(h_u^{(\ell-1)}, h_v^{(\ell-1)}, e_{uv}\big)\Big), \tag{1}$$

where $h_v^{(0)} = x_v$, the message function $\phi^{(\ell)}$ and update function $\psi^{(\ell)}$ are parameterized by layer-specific weights $\theta^{(\ell)}$, and $\bigoplus$ denotes a permutation-invariant aggregation (e.g., sum, mean, or max). For graph-level tasks, a permutation-invariant readout ρ pools over the final node representations to yield $h_G = \rho\big(\{\{h_v^{(L)} : v \in V\}\}\big)$; for node-level tasks, $h_v^{(L)}$ is passed directly to a task-specific head. Particular choices of ϕ, ψ, and $\bigoplus$ recover standard MPNN architectures (Kipf & Welling, 2017; Hamilton et al., 2017; Xu et al., 2019; Veličković et al., 2018).

2.1 Over-Smoothing, Under-Reaching, the *Elasticity Window*

Receptive field and under-reaching. Consider the iterative structure of the message passing algorithm in Equation 1. The representation $h_v^{(L)}$ is determined entirely by the subgraph induced on the L-hop neighborhood $\mathcal{N}^L(v)$. Any dependency between nodes u, v with $\mathrm{d}_G(u, v) > L$ is therefore unreachable at depth L: the representations $h_u^{(L)}$ and $h_v^{(L)}$ depend on disjoint subgraphs and cannot influence each other. In the worst case, capturing all pairwise dependencies requires a

depth matching the graph diameter $\delta(G) := \max_{u,v \in V} \mathrm{d}_G(u, v)$. This yields the *reach requirement* $T_{\mathrm{reach}}(G) := \delta(G)$; architectures operating with $L < T_{\mathrm{reach}}(G)$ are said to *under-reach* on G[2].

Over-smoothing as topology-dependent contraction. Counterbalancing this lower bound, an effective upper bound on depth arises from *over-smoothing* (Cai & Wang, 2020; Li et al., 2018; Oono & Suzuki, 2020; Rusch et al., 2023): as ℓ grows, the representations $\{h_v^{(\ell)}\}_{v \in V}$ converge toward a subspace in which they become indistinguishable, and thus erode the discriminative power of the model. A standard measure of this feature collapse is the Dirichlet energy,

$$\mathcal{E}(H) := \frac{1}{2} \sum_{(u,v) \in E} \|h_u - h_v\|^2, \tag{2}$$

where $H \in \mathbb{R}^{N \times d}$ stacks node features row-wise and rapid decrease in $\mathcal{E}(H)$ indicates oversmoothing. Under mild regularity assumptions on $\phi^{(\ell)}$ and $\psi^{(\ell)}$, prior analyses (Cai & Wang, 2020; Oono & Suzuki, 2020; Rusch et al., 2023) establish that

$$\mathcal{E}(H^{(\ell)}) \;\leq\; \kappa(G)^{\ell}\, \mathcal{E}(H^{(0)}), \tag{3}$$

for a contraction rate $\kappa(G) \in [0, 1)$ governed by topological quantities such as $|\lambda_2(\hat{A})|$, the second-largest eigenvalue magnitude of the normalized adjacency (Chung, 1997). Intuitively, comparatively densely connected or small diameter graphs mix information faster, yielding smaller $\kappa(G)$ and thus more rapid feature collapse (Figure 1). We summarize this phenomenon through the *smoothing time*

$$T_{\mathrm{smooth}}(G) := -\frac{1}{\log \kappa(G)}, \tag{4}$$

the characteristic depth at which $\mathcal{E}(H^{(\ell)})$ has decayed by a factor of $1/e$ relative to its initial value.

The elasticity window. $T_{\mathrm{reach}}(G)$ and $T_{\mathrm{smooth}}(G)$ bound the admissible depth from below and above: a fixed-depth MPNN avoids both under-reaching and over-smoothing only when

$$T_{\mathrm{reach}}(G) \;\leq\; L \;\lesssim\; T_{\mathrm{smooth}}(G), \tag{5}$$

a range we refer to as the *elasticity window* for G. Both its width and its location are functions of graph topology. For example, dense and low-diameter graphs admit small T_{reach} but likewise small T_{smooth}, producing a narrow window at shallow depth; elongated or weakly connected graphs push T_{reach} outward and typically shift T_{smooth} rightward as well. Consequently, when a dataset exhibits topological heterogeneity — that is, when the elasticity windows of its constituent graphs fail to overlap — no single choice of L is uniformly admissible. This formalizes, at the level of the message-passing operator, the coupling problem identified in the opening discussion of the manuscript: depth is an architectural knob that governs parameterization, reach, and computation in unison, yet its appropriate value is a graph-specific quantity that no fixed choice of L can accommodate.

2.2 Related Work

We survey directly relevant lines of work. A comprehensive treatment is provided in Appendix A.

Recursive transformers and adaptive computation. Universal Transformers (Dehghani et al., 2019), looped transformers (Giannou et al., 2023; Jeddi et al., 2026; Kohli et al., 2026), and *Tiny Recursive Models* (Jolicoeur-Martineau, 2025) demonstrate that weight-shared iteration can match or exceed deeper architectures in the sequence domain. These works operate on fixed topologies (sequences), where the coupling problem does not arise. TRGNNs bring recursion to the graph domain, where structural variance provides a topology-grounded motivation for adaptive depth.

Recurrent, weight-tied, and implicit-layer GNNs. Recurrent architectures for graph representation learning predate layer-stacked message passing. Such classical models repeatedly apply a single weight-tied transition to latent node states and then read out a final prediction. They differ chiefly in how the recursion is run (Appendix A.2): either to a contractive fixed point (Scarselli et al., 2009; Gallicchio & Micheli, 2010; Micheli & Tortorella, 2023), for a fixed (Li et al., 2016) or learned (Tang et al., 2020) number of steps, or as an implicit layer solved for its equilibrium (Gu et al., 2020; Baker

[2]A closely related and well-studied pathology, *over-squashing* (Alon & Yahav, 2021; Topping et al., 2022),captures signal compression as it propagates through graph bottlenecks, and is similarly governed by topology.

et al., 2023; Liu et al., 2021; Georgiev et al., 2024). TRGNNs instead make the *prediction* part of the recurrent state and feed it back into each latent update (Equation 6). Our ablation TRGNN^{-y} (Section 5) is exactly this classical recurrence with a GIN/GINE transition.

Graph transformers. These architectures (Kim et al., 2021; 2022; Kreuzer et al., 2021; Mialon et al., 2021) sidestep the coupling problem by replacing topology-constrained aggregation with dense attention over the full node set. This removes the receptive field bottleneck at the cost of $O(N^2)$ complexity and the locality and sparsity biases that underlie message passing. TRGNNs instead resolve the coupling while preserving these priors.

Mitigating Over-smoothing. Residual connections (Chen et al., 2020), propagation decoupling (Gasteiger et al., 2019), normalization (Zhao & Akoglu, 2020), and edge dropping (Rong et al., 2020) attempt to reduce over-smoothing by widening the elasticity window for a given graph. However, these approaches operate within the fixed-depth paradigm and do not make effective depth itself adaptive. They are thus complementary to the proposed recursive formulation.

3 Tiny Recursive Graph Neural Networks

TRGNNs replace the stacked-layer construction of standard message passing (Equation 1) with recursive application of a weight-shared operator, making effective depth a free parameter rather than a fixed architectural choice. Rather than relying on external halting signals to exploit this decoupling, TRGNNs couple the recurrence to a prediction state $y^{(t)}$ refined jointly with latent representation $H^{(t)}$. This design, shared with iterative refinement in neural reasoning (Jolicoeur-Martineau, 2025) and classical graph algorithms (Pearl, 1982; Zhu & Ghahramani, 2002; Page et al., 1999), allows the model to preserve its output once further propagation is uninformative, so that additional iteration need not degrade performance. Section 4 formalizes this mechanism.

3.1 Architecture

A TRGNN is parameterized by three weight-shared modules applied at every iteration: an encoder $\eta_\alpha : \mathbb{R}^d \to \mathbb{R}^{d_h}$ that lifts input features to a hidden representation, a message-passing block f_θ consisting of L_{block} layers of the form of Equation 1, and a prediction head g_ω. Inference maintains a coupled state $(H^{(t)}, y^{(t)})$ evolving across outer iterations $t = 1, \dots, T$:

- $H^{(t)} \in \mathbb{R}^{N \times d_h}$: a latent node representation, refined across iterations;
- $y^{(t)}$: a running prediction, either graph-level ($y^{(t)} \in \mathbb{R}^{d_y}$) or node-level ($y^{(t)} \in \mathbb{R}^{N \times d_y}$).

Algorithm 1 TRGNN forward pass.

Input: G, X; $T, n \in \mathbb{N}$
$X_{\text{enc}} \leftarrow \eta_\alpha(X)$
$H \leftarrow X_{\text{enc}}$, $y \leftarrow \mathbf{0}$
for $t = 1$ to T **do**
  for $i = 1$ to n **do**
    $H \leftarrow f_\theta\big(G,\ [X_{\text{enc}} \,\|\, H \,\|\, \bar{y}]\big)$
  end for
  $y \leftarrow g_\omega\big(\rho(H),\ y\big)$
end for
return y

The encoded input $X_{\text{enc}} := \eta_\alpha(X)$ is computed once and held fixed, and the coupled state is initialized as $H^{(0)} := X_{\text{enc}}$ and $y^{(0)} := \mathbf{0}$. For uniform notation across task types, we write $\bar{y} \in \mathbb{R}^{N \times d_y}$ for the row-wise broadcast of y to the node set, defined as $\bar{y} = \mathbf{1}_N y^\top$ for graph-level $y \in \mathbb{R}^{d_y}$ and $\bar{y} = y$ for node-level y.

Inner update. At each outer iteration t, the latent state $H^{(t-1)}$ is refined through n applications of the shared message-passing block, each conditioned on the encoded input and the current prediction. Writing $H^{(t,0)} := H^{(t-1)}$, for $i = 1, \dots, n$:

$$H^{(t,i)} \;=\; f_\theta\Big(G,\ \big[X_{\text{enc}} \,\|\, H^{(t,i-1)} \,\|\, \bar{y}^{(t-1)}\big]\Big), \tag{6}$$

where $\|$ denotes concatenation along the feature dimension.

Outer update. The prediction head g_ω refines y from the pooled latent state and previous prediction:

$$y^{(t)} \;=\; g_\omega\Big(\rho(H^{(t)}),\ y^{(t-1)}\Big), \tag{7}$$

where ρ is the invariant readout introduced in Section 2 (the identity for node-level tasks). Equations 6 and 7 define a outer-iteration map $\mathcal{R} : (H^{(t-1)}, y^{(t-1)}) \mapsto (H^{(t)}, y^{(t)})$; the forward pass arises from applying $\mathcal{R}$ a total of T times from the initial state. Algorithm 1 summarizes the full procedure.

Design choices. Three aspects of the architecture warrant discussion. First, feeding $y^{(t-1)}$ into the latent update distinguishes TRGNNs from purely latent recursive models of Section 2: subsequent iterations reason in light of the current output. Second, we retain a separate prediction head g_ω rather than folding prediction into the message-passing block, isolating node-level representation refinement (f_θ) from output refinement (g_ω) and allowing g_ω to implement non-trivial dynamics on y. Third, f_θ is shallow ($L_{\text{block}} \in \{1, 2\}$) so that expressiveness accrues through iteration rather than stacking.

Properties. The formulation has three immediate consequences.

- *Parameterization is decoupled from T.* All iterations share $\{\alpha, \theta, \omega\}$, so the parameter budget is determined entirely by f_θ, η_α, and g_ω, independent of the number of outer iterations applied.
- *Receptive field grows linearly with iterations.* f_θ applies L_{block} message-passing layers per inner step and there are n inner steps per outer iteration. Thus, $H_v^{(t)}$ depends on the subgraph induced by $\mathcal{N}^{t \cdot n \cdot L_{\text{block}}}(v)$. Increasing T extends the receptive field without modifying parameterization.
- *TRGNN generalizes standard MPNNs.* Setting $T = 1$ and removing prediction feedback $\bar{y}^{(t-1)}$ from Equation 6 yields a single application of f_θ and g_ω, i.e., a L_{block}-layer MPNN (Equation 1).

3.2 Training and Anytime Inference

Training. TRGNNs are trained by supervising the final prediction $y^{(T)}$ with a task loss $\mathcal{L}(y^{(T)}, y^\star)$. A naive implementation would backpropagate through all T outer iterations, incurring memory cost $O(T \cdot n \cdot L_{\text{block}})$ that grows linearly with T and tightly bounds the achievable effective depth in practice. To circumvent this, we unroll the first $T - 1$ outer iterations under stop-gradient,

$$(H^{(t)}, y^{(t)}) \;=\; \mathrm{sg}\Big[\mathcal{R}(H^{(t-1)}, y^{(t-1)})\Big], \qquad t = 1, \ldots, T-1, \tag{8}$$

where $\mathrm{sg}[\cdot]$ denotes the stop-gradient operator, acting as the identity on the forward pass and with zero Jacobian on the backward pass. A gradient update is then applied only through the final iteration $(H^{(T)}, y^{(T)}) = \mathcal{R}(H^{(T-1)}, y^{(T-1)})$, so that training memory is constant in T, scaling only as $O(n \cdot L_{\text{block}})$ in the layers through which gradients flow.

Beyond efficiency, this scheme admits a natural interpretation: the model learns a one-step refinement operator at the transition $(H^{(T-1)}, y^{(T-1)}) \to (H^{(T)}, y^{(T)})$, with the preceding warm-up supplying realistic input states from the refinement dynamics themselves. At test time, the same $\mathcal{R}$ is simply iterated from initialization. Section 4 formalizes when this iteration is well-behaved.

Anytime inference. Because every outer iteration produces a valid prediction $y^{(t)}$, inference is anytime (Huang et al., 2017; McGill & Perona, 2017; Zilberstein, 1996) by construction: the sequence $y^{(t)}{}_{t \geq 1}$ is available at arbitrary stopping points, enabling flexible accuracy–compute tradeoffs from a single trained model. In Section 4, we characterize conditions under which this sequence converges, while the architecture permits predictions to stabilize under continued iteration.

4 TRGNNs are Topology-Adaptive

The architecture of Section 3 decouples effective depth from parameterization. We now establish, under a restricted but standard setting, that recursive message passing with iterative prediction refinement induces topology-adaptivity. We establish two results under progressively stronger assumptions: the capacity for prediction preservation at each iteration (Proposition 4.1) and, under a contraction condition, convergence at a topology-dependent rate (Proposition 4.2)[3].

Definition 4.1 (Effective Depth and Topology-Adaptivity)**.** Let $\{y^{(t)}\}_{t \geq 0}$ denote the prediction sequence of a recursive model on graph G, from initialization $y^{(0)} = \mathbf{0}$. If the sequence converges to a limit $y^*(G)$, the *effective depth* at tolerance $\varepsilon > 0$ is

$$T_{\text{eff}}(G, \varepsilon) \;:=\; \inf\big\{t \in \mathbb{N} : \|y^{(t)} - y^*(G)\| < \varepsilon\big\}. \tag{9}$$

A recursive model is *topology-adaptive* if, under fixed parameters, $\kappa(G_1) \leq \kappa(G_2)$ implies $T_{\text{eff}}(G_1, \varepsilon) \leq T_{\text{eff}}(G_2, \varepsilon)$ for all graphs G_1, G_2 and sufficiently small ϵ.

Definition 4.1 ties adaptivity to $\kappa(G)$, which sets the elasticity window (Equation 5): adaptive models allocate more effective depth to graphs tolerating more propagation before over-smoothing.

[3] N.B.: We do not impose contractivity in the trained models of Section 5, yet observe adaptivity empirically.

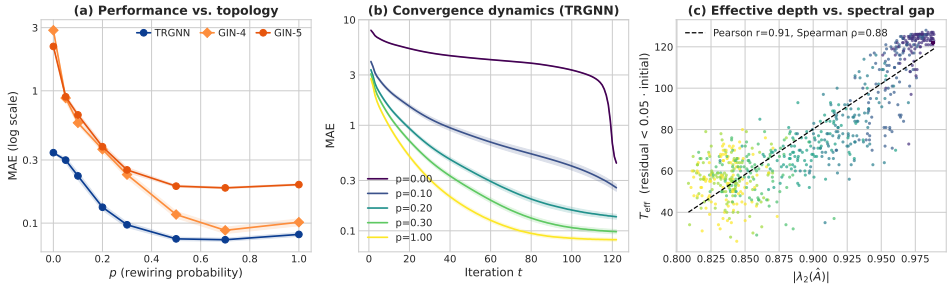

Figure 2: **Controlled topology-adaptivity experiments.** Watts-Strogatz graphs at varying rewiring probability p. **(a)** Node-level test MAE vs. p. Fixed-depth GINs exhibit the under-reach/over-smoothing tradeoff (Equation 5) while TRGNN adapts uniformly. **(b)** TRGNN test MAE at each iteration t, stratified by p. Convergence is monotone and topology-dependent. **(c)** Measured effective depth T_{eff} vs. $|\lambda_2(\hat{A})|$ across test graphs with linear fit, validating spectral dependence of Section 4.1

.

4.1 Prediction Preservation and Convergence

Proposition 4.1 (Capacity for Prediction Preservation)**.** *Let $g_\omega : \mathbb{R}^{d_z} \times \mathbb{R}^{d_y} \to \mathbb{R}^{d_y}$ be the prediction head (Equation 7), parameterized as a universal approximator. Then for any compact $K \subset \mathbb{R}^{d_z} \times \mathbb{R}^{d_y}$ and any $\varepsilon > 0$, there exist parameters ω^* such that*

$$\sup_{(z,\, y) \in K} \|g_{\omega^*}(z, y) - y\| \;<\; \varepsilon.$$

Thus, for any graph G and iteration t such that $(\rho(H^{(t)}),\, y^{(t-1)}) \in K$, we have $\|y^{(t)} - y^{(t-1)}\| < \varepsilon$.

The universal approximation assumption is satisfied by standard MLP parameterizations of sufficient width. Prediction heads implementing a residual update $g_\omega(z, y) = y + \Delta_\omega(z, y)$ achieve exact prediction preservation ($\Delta_\omega \equiv 0$) as a special case. Crucially, this property arises from the architectural coupling of g_ω to $y^{(t-1)}$, not from universal approximation alone: fixed-depth models lack a persistent prediction state and thus have no mechanism to condition their output on a previous prediction. Because g_ω receives (z, y) jointly, it can more generally represent any continuous refinement strategy, including one that selectively preserves y on converged inputs while continuing to refine on others.

Proposition 4.1 establishes that the architecture *permits* prediction preservation but does not guarantee that learned parameters achieve it. The following result provides the stronger guarantee: under a contraction condition, iteration actively *converges*, and does so at a rate governed by the input graph.

Proposition 4.2 (Topology-Dependent Convergence)**.** *Let $\mathcal{R}_G$ denote the outer-iteration map (Equations 6–7), and let $\mathcal{S}_G := \mathbb{R}^{N \times d_h} \times \mathcal{Y}$ denote the coupled state space, with $\mathcal{Y} = \mathbb{R}^{d_y}$ (graph-level) or $\mathcal{Y} = \mathbb{R}^{N \times d_y}$ (node-level). Suppose $\mathcal{R}_G$ contracts $\mathcal{S}_G$ with rate $\gamma(G) := \mathrm{Lip}(\mathcal{R}_G) < 1$. Then:*

- *(i) $\mathcal{R}_G$ admits a unique fixed point $(H^*(G), y^*(G)) \in \mathcal{S}_G$.*
- *(ii) The iterates converge geometrically: $\|(H^{(t)}, y^{(t)}) - (H^*, y^*)\| \;\le\; \gamma(G)^t \cdot D_0$, where $D_0 := \|(H^{(0)}, y^{(0)}) - (H^*, y^*)\|$.*
- *(iii) The effective depth satisfies $T_{\text{eff}}(G, \varepsilon) \le \left\lceil \frac{\log(D_0/\varepsilon)}{\log \gamma^{-1}(G)} \right\rceil$.*

Proposition 4.2 follows from the Banach fixed-point theorem. Contractivity ($\gamma(G) < 1$) can be encouraged by enforcing $\mathrm{Lip}(f_\theta)\,\mathrm{Lip}(g_\omega) < 1$, e.g., via spectral normalization (Miyato et al., 2018) and we empirically probe convergence in Sections 4.2 and 5.2. Topological heterogeneity precludes any fixed depth L from lying in every graph's elasticity window, whereas, under contraction, TRGNNs admit a bounded, graph-dependent effective depth $T_{\text{eff}}(G)$. Proposition 4.2 leaves open the relation between $\gamma(G)$ and $\kappa(G)$ as for arbitrary learned non-linear dynamics, $\gamma(G)$ depends on both the learned operator and topology. In the standard normalized-adjacency setting, however, contraction of the message-passing component on the disagreement subspace is governed by $|\lambda_2(\hat{A})|$, and Proposition E.1 shows the topology-dependent component is nondecreasing in $\kappa(G)$.

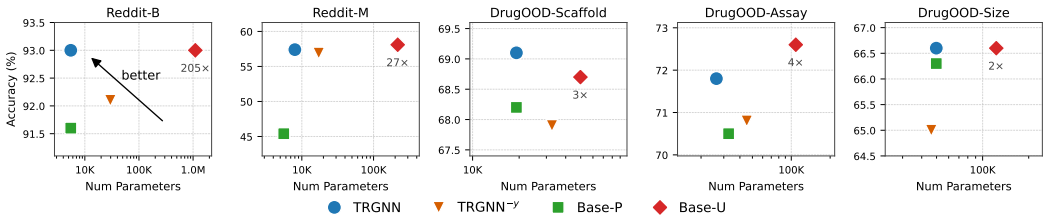

Figure 3: **Parameter efficiency on standard benchmarks.** Validation accuracy versus parameter count (log scale) for TRGNN, ablation TRGNN^{-y}, a parameter-matched MPNN (Base-P, $|\theta| \pm 15\%$ of TRGNN), and an unconstrained MPNN (Base-U). Reddit reports 10-fold accuracy (%); DrugOOD reports OOD AUROC (%). Multipliers indicate ratio of parameters between best performing TRGNN and Base-U. TRGNN sits to the left of Base-U at comparable accuracy on tested tasks.

4.2 Controlled Experiments

We validate the preceding analysis on Watts-Strogatz graphs (Watts & Strogatz, 1998), where rewiring probability $p \in [0, 1]$ interpolates between ring lattices ($p = 0$) and random graphs ($p = 1$). Since $|\lambda_2(\hat{A})|$ governs $\kappa(G)$, varying p continuously modulates both $T_{\text{reach}}(G)$ and $T_{\text{smooth}}(G)$. The task is node-level shortest-path regression from a designated source node. We compare TRGNN against parameter-matched GIN baselines at depths $L \in \{4, 5\}$. Experimental details are in Appendix C.

Figure 2(a) demonstrates the coupling problem: GIN-5 outperforms GIN-4 at low p, where its greater depth meets the reach requirement, but is overtaken at high p as over-smoothing erodes its advantage. Neither baseline dominates across the full topological range. TRGNN, operating at matched parameter count, achieves uniformly low MAE (0.17 overall vs. 0.62 for the best baseline).

Panel (b) shows TRGNN prediction trajectories $\{y^{(t)}\}$, stratified by topology. Task error decreases monotonically across each graph family, consistent with Proposition 4.1. Convergence varies markedly: random graphs ($p = 1.0$) saturate within ≈ 20 iterations, while ring lattices ($p = 0.0$) improve past 100, reflecting the topology dependence of $\gamma(G)$ in Proposition 4.2. Panel (c) shows that $T_{\text{eff}}(G)$ correlates strongly with $|\lambda_2(\hat{A}(G))|$ (Pearson $r = 0.91$), consistent with Section 4.1.

5 Experiments

Having established topology-adaptivity (Section 4), we now turn to real-world benchmarks.

Datasets. We consider common graph benchmarks (Hosseini et al., 2025; Kanatsoulis et al., 2025): Reddit-B and Reddit-M (Yanardag & Vishwanathan, 2015) (social network classification), DrugOOD (Ji et al., 2023) with assay, scaffold, and size distribution shifts (molecular property prediction under domain shift), and Peptides-struct (Dwivedi et al., 2022) (long-range structure prediction).

Models. Sections 5.1–5.2 compare three variants: a standard GNN, recursive GNN without prediction refinement (**TRGNN^{-y}**), and **TRGNN**. All use GIN (Xu et al., 2019) without edge features and GINE (Hu* et al., 2020) otherwise. Appendix D reports alternative backbones, Appendix F additional baselines, and Appendix C experimental details. Section 5.3 compares against PEARL (Kanatsoulis et al., 2025), GPS (Rampášek et al., 2022), SAN (Kreuzer et al., 2021), GNN-AK (Zhao et al., 2022), SUN (Frasca et al., 2022), GViT (Dwivedi & Bresson, 2020), and SRF (Hosseini et al., 2025).

5.1 TRGNNs are Parameter Efficient Learners

As shown in the sequence domain (Section 2), recursive architectures trade parameters for iterations. We test this tradeoff in the graph setting against two GNN baselines: Base-P, matched to TRGNN's parameter count, and independently tuned Base-U. Figure 3 reports five benchmarks. TRGNN stays within 0.8 points of Base-U on each dataset with far fewer parameters. The gap is largest on social-network tasks, where graphs lack node features and performance depends on structure. On molecular tasks, TRGNN uses 2–4× fewer parameters while matching or exceeding Base-U across distribution shifts. TRGNN^{-y} underperforms overall despite using more parameters than TRGNN.

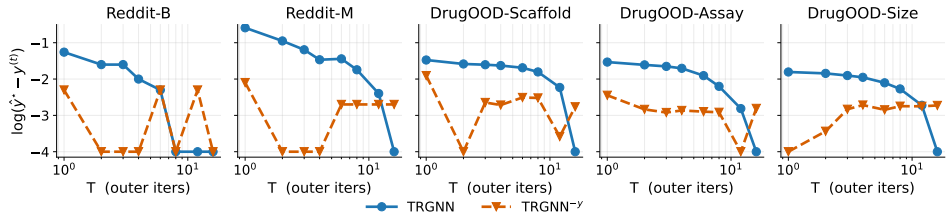

Figure 4: **Anytime convergence.** For each model, $\log(\hat{y}^\star - y^{(t)})$ plots the log-gap to its own best performance $\hat{y}^\star$ across iterations. TRGNN exhibits approximately linear decrease, consistent with geometric convergence (Proposition 4.2(ii)). Conversely, TRGNN^{-y} exhibits non-monotonicity.

5.2 TRGNNs are Principled Anytime Predictors

Section 3.2 establishes that TRGNNs produce valid predictions at every iteration, enabling anytime inference. Its practical value depends on whether predictions improve with iteration. Figure 5 measures this trend via Spearman correlation between iteration t and performance. TRGNN achieves near-perfect rank correlation ($\rho \geq 0.95$) across each benchmark. TRGNN^{-y} generally shows weak or negative correlation, including $\rho \approx -0.6$ on DrugOOD-Size, where iteration degrades performance. Figure 4 shows the log-gap to each model's best achieved performance against T. TRGNN shows an approximately linear decline, consistent with Proposition 4.2(ii); TRGNN^{-y} is non-monotonic.

5.3 TRGNNs are Competitive Beyond Message Passing Baselines

The preceding experiments isolate recursion and prediction refinement against conventional MPNNs. We now compare TRGNN with message-passing, hybrid, and transformer architectures on Peptides-struct (Dwivedi et al., 2022), a long-range task challenging for conventional MPNNs. TRGNN matches or outperforms all but two baselines without attention or auxiliary structural features (Table 1). GViT uses $O(N^2)$ attention and random walk encodings unavailable to other methods (Kanatsoulis et al., 2025), while B-PEARL+SRF relies on PEARL, which incurs $\sim$32$\times$ runtime and $\sim$45$\times$ peak memory of TRGNN (Appendix D). These results support our thesis: recursive depth addresses the coupling problem while preserving the priors of message-passing.

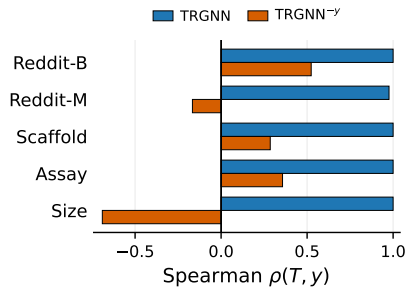

Figure 5: **Anytime monotonicity.** Spearman $\rho(T, y)$ across iteration trajectory of each method (higher is better). Scaffold/Assay/Size are DrugOOD splits.

Table 1: **Long-range benchmark.** MAE (↓) on Peptides-struct. Error bars do not exceed ≤ 0.004. TRGNN's pure message-passing model reaches parity with the strongest attention and structural-feature-based baselines on a benchmark designed to expose message passing's reach limitations.

Baselines								Ours
R-PEARL	B-PEARL	GPS	SAN_{RWSE}	GNN-AK+	SUN	GViT	$\text{B-PEARL}_{\text{SRF}}$	TRGNN
0.247	0.248	0.252	0.255	0.274	0.250	0.245	0.243	0.247

6 Conclusion

We have presented Tiny Recursive Graph Neural Networks, which decouple effective depth from network depth and adapt computation to graph topology. One limitation of this method, as with recursive architectures more broadly, is that parameter savings come at the cost of sequential inference steps. TRGNNs are thus best suited to memory constrained or latency tolerant settings (Appendix C).

AI USE STATEMENT

We used generative AI tools to assist with limited implementation and writing tasks, including generating boilerplate experimental code, editing figures for presentation and formatting, and revising text for clarity or concision. We did not use generative AI to develop the paper's theoretical framework, formulate mathematical claims, design the core methodology, or interpret experimental results. All AI-assisted work was carefully reviewed by the authors, who take full responsibility for the final content of the paper.

ETHICS STATEMENT

This work studies architectures for graph learning on public benchmark datasets used in accordance with their licenses. It involves no human subjects, personal data, or applications that we consider to raise ethical concerns beyond those common to machine learning research.

REPRODUCIBILITY STATEMENT

Architectures, training procedures, and hyperparameter search spaces are given in Section 3 and Appendix C. All theoretical statements list their assumptions and are proved in Appendices B and E. Dataset processing follows the public releases cited in Appendix C.

REFERENCES

Yassine Abbahaddou, Fragkiskos D. Malliaros, Johannes F. Lutzeyer, and Michalis Vazirgiannis. ADMP-GNN: Adaptive depth message passing GNN. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM)*, 2025. doi: 10.1145/3746252.

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

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in neural information processing systems*, 32, 2019.

Justin Baker, Qingsong Wang, Cory D Hauck, and Bao Wang. Implicit graph neural networks: A monotone operator viewpoint. In *International conference on machine learning*, pp. 1521–1548. PMLR, 2023.

Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder. In *8th ICML Workshop on Automated Machine Learning (AutoML)*, 2021. URL https://openreview.net/forum?id=1EuxRTe0WN.

Arpit Bansal, Avi Schwarzschild, Eitan Borgnia, Zeyad Emam, Furong Huang, Micah Goldblum, and Tom Goldstein. End-to-end algorithm synthesis with recurrent networks: Extrapolation without overthinking. *Advances in Neural Information Processing Systems*, 35:20232–20242, 2022.

Lukas Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=F72ximsx7C1.

Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020.

Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. Grand: Graph neural diffusion. In *International conference on machine learning*, pp. 1407–1418. PMLR, 2021.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International conference on machine learning*, pp. 1725–1735. PMLR, 2020.

Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

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 *International conference on machine learning*, pp. 7865–7885. PMLR, 2023.

Andrew J Dudzik and Petar Veličković. Graph neural networks are dynamic programmers. *Advances in neural information processing systems*, 35:20635–20647, 2022.

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

Vijay Prakash Dwivedi, Ladislav Rampášek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. *Advances in Neural Information Processing Systems*, 35:22326–22340, 2022.

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=2edigk8yoU.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Fabrizio Frasca, Beatrice Bevilacqua, Michael Bronstein, and Haggai Maron. Understanding and extending subgraph gnns by rethinking their symmetries. *Advances in Neural Information Processing Systems*, 35:31376–31390, 2022.

Claudio Gallicchio and Alessio Micheli. Graph echo state networks. In *International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2010. doi: 10.1109/IJCNN.2010.5596796.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Combining neural networks with personalized pagerank for classification on graphs. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1gL-2A9Ym.

Jonas Geiping, Sean Michael McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=S3GhJooWIC.

Dobrik Georgiev, JJ Wilson, Davide Buffelli, and Pietro Liò. Deep equilibrium algorithmic reasoning. *Advances in Neural Information Processing Systems*, 37:33638–33667, 2024.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning*, pp. 11398–11442. PMLR, 2023.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. Pmlr, 2017.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Fangda Gu, Heng Chang, Wenwu Zhu, Somayeh Sojoudi, and Laurent El Ghaoui. Implicit graph neural networks. *Advances in neural information processing systems*, 33:11984–11995, 2020.

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

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, DDL Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10, 2022.

Ryien Hosseini, Filippo Simini, Venkatram Vishwanath, Rebecca Willett, and Henry Hoffmann. Sketch-augmented features improve learning long-range dependencies in graph neural networks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

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

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens Van Der Maaten, and Kilian Q Weinberger. Multi-scale dense convolutional networks for efficient prediction. *arXiv preprint arXiv:1703.09844*, 2(2), 2017.

Ahmadreza Jeddi, Marco Ciccone, and Babak Taati. Loopformer: Elastic-depth looped transformers for latent reasoning via shortcut modulation. *arXiv preprint arXiv:2602.11451*, 2026.

Yuanfeng Ji, Lu Zhang, Jiaxiang Wu, Bingzhe Wu, Lanqing Li, Long-Kai Huang, Tingyang Xu, Yu Rong, Jie Ren, Ding Xue, et al. Drugood: Out-of-distribution dataset curator and benchmark for ai-aided drug discovery–a focus on affinity prediction problems with noise annotations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 8023–8031, 2023.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025.

Charilaos Kanatsoulis, Evelyn Choi, Stefanie Jegelka, Jure Leskovec, and Alejandro Ribeiro. Learning efficient positional encodings with graph neural networks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=AWg2tkbydO.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Jinwoo Kim, Saeyoon Oh, and Seunghoon Hong. Transformers generalize deepsets and can be extended to graphs & hypergraphs. *Advances in Neural Information Processing Systems*, 34: 28016–28028, 2021.

Jinwoo Kim, Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. *Advances in Neural Information Processing Systems*, 35:14582–14595, 2022.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Harsh Kohli, Srinivasan Parthasarathy, Huan Sun, and Yuekun Yao. Loop, think, & generalize: Implicit reasoning in recurrent-depth transformers. *arXiv preprint arXiv:2604.07822*, 2026.

Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in neural information processing systems*, 34:21618–21629, 2021.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Hankook Lee and Jinwoo Shin. Anytime neural prediction via slicing networks vertically. *arXiv preprint arXiv:1807.02609*, 2018.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *International Conference on Learning Representations (ICLR)*, 2016.

Juncheng Liu, Kenji Kawaguchi, Bryan Hooi, Yiwei Wang, and Xiaokui Xiao. Eignn: Efficient infinite-depth graph neural networks. *Advances in Neural Information Processing Systems*, 34: 18762–18773, 2021.

Mason McGill and Pietro Perona. Deciding how to decide: Dynamic routing in artificial neural networks. In *International Conference on Machine Learning*, pp. 2363–2372. PMLR, 2017.

William Merrill and Ashish Sabharwal. A little depth goes a long way: The expressive power of log-depth transformers. *arXiv preprint arXiv:2503.03961*, 2025.

Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667*, 2021.

Alessio Micheli and Domenico Tortorella. Addressing heterophily in node classification with graph echo state networks. *Neurocomputing*, 550:126506, 2023. doi: 10.1016/j.neucom.2023.126506.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1QRgziT-.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1ldO2EFPr.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford infolab, 1999.

Judea Pearl. 11 reverend bayes on inference engines: A distributed hierarchical. *Probabilistic and Causal Inference: The Works of Judea Pearl*, pp. 129, 1982.

Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

Zihan Pengmei, Zhengyuan Shen, Zichen Wang, Marcus D. Collins, and Huzefa Rangwala. Pushing the limits of all-atom geometric graph neural networks: Pre-training, scaling, and zero-shot transfer. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4S2L519nIX.

Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Hkx1qkrKPr.

T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. doi: 10.1109/TNN.2008.2005605.

Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Furong Huang, Uzi Vishkin, Micah Goldblum, and Tom Goldstein. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. *Advances in Neural Information Processing Systems*, 34:6695–6706, 2021.

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. PMLR, 2023.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Maciej Sypetkowski, Frederik Wenkel, Farimah Poursafaei, Nia Dickson, Karush Suri, Philip Fradkin, and Dominique Beaini. On the scalability of gnns for molecular graphs. *Advances in Neural Information Processing Systems*, 37:19870–19906, 2024.

Hao Tang, Zhiao Huang, Jiayuan Gu, Bao-Liang Lu, and Hao Su. Towards scale-invariant graph-related problem solving by iterative homogeneous gnns. *Advances in Neural Information Processing Systems*, 33:15811–15822, 2020.

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pp. 2464–2469. IEEE, 2016.

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.

Petar Veličković and Charles Blundell. Neural algorithmic reasoning. *Patterns*, 2(7), 2021.

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

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025.

Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393 (6684):440–442, 1998.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pp. 5453–5462. pmlr, 2018.

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

Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S. Du, Ken ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rJxbJeHFPS.

Pinar Yanardag and Svn VN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1365–1374, 2015.

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.

Jianan Zhao, Chaozhuo Li, Qian Wen, Yiqi Wang, Yuming Liu, Hao Sun, Xing Xie, and Yanfang Ye. Gophormer: Ego-graph transformer for node classification. *ArXiv*, abs/2110.13094, 2021. URL https://api.semanticscholar.org/CorpusId:239768298.

Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkecl1rtwB.

Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. From stars to subgraphs: Uplifting any GNN with local structure awareness. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Mspk_WYKoEH.

Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. *ProQuest number: information to all users*, 2002.

Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI magazine*, 17(3):73–73, 1996.

A Prior Work in Detail

A.1 Recursive and Weight-Tied Architectures in the Sequence Domain

Cross-layer weight tying in transformer architectures has been studied both as a parameter compression mechanism (Lan et al., 2020) and, more relevant to this manuscript, as a means of allocating computation at test time: Universal Transformers (Dehghani et al., 2019) apply a single shared block to a fixed input sequence for a learned number of steps. More recent looped transformers can implement arbitrary programs whose runtime scales with the iteration count (Giannou et al., 2023), a view that has been exploited to obtain length generalization on algorithmic tasks by adapting the iteration count at inference (Fan et al., 2025). A complementary capacity result establishes that log-depth transformers express problems out of reach of fixed shallow ones (Merrill & Sabharwal, 2025). At scale, recurrent-depth language models allocate reasoning compute by unrolling a shared block rather than by emitting additional tokens (Geiping et al., 2026; Jeddi et al., 2026; Kohli et al., 2026).

Per-input adaptive depth has been previously studied in the sequence domain in two major ways. The first augments a recurrent computation with a learned halting distribution, casting depth selection as an explicit optimization (Graves, 2016; Banino et al., 2021). The second is more implicit: weight-tied networks trained on short instances of an algorithmic task can continue to improve as the recurrence is unrolled longer at test time, with no halting signal supplied (Schwarzschild et al., 2021; Bansal et al., 2022). The Hierarchical Reasoning Model (Wang et al., 2025) and the Tiny Recursive Model (Jolicoeur-Martineau, 2025) make this dynamic explicit: a small recurrent block iteratively refines a persistent answer state, so that adaptive computation arises from the convergence dynamics of the refinement process itself. TRGNNs adopt this design — the persistent prediction state $y^{(t)}$ and the prediction head g_ω that conditions on it (Section 3) are direct analogues of the corresponding constructs in (Jolicoeur-Martineau, 2025).

All of the above operate on sequence inputs, where the underlying topology (generally a chain of sequences) is fixed across the dataset. In this regime, adaptive depth is a per-input convenience that targets variation in instance difficulty rather than a structural mismatch with the data. On graphs, by contrast, the mismatch is intrinsic to the domain: as established in Section 2.1, the elasticity window varies with topology across inputs of the same dataset, and no fixed depth lies in every such window simultaneously. TRGNNs bring recursion into a setting where input-adaptive depth is a structural requirement rather than an efficiency consideration.

A.2 Recurrent, Weight-Tied, and Implicit-Layer GNNs

Explicit recurrent GNNs. Recursion on graphs predates layer-stacked message passing. The original graph neural network of Scarselli et al. (2009) iterates a contractive latent transition $H^{(t+1)} = F_\theta(G, X, H^{(t)})$ to its unique fixed point H^*, reads out $\hat{Y} = G_\omega(H^*, X)$, and computes gradients with the Almeida–Pineda recurrence rather than by unrolling. Gated graph neural networks (Li et al., 2016) drop the contractivity constraint and unroll a gated recurrent unit for a fixed number of steps, trained by backpropagation through time. Graph echo state networks (Gallicchio & Micheli, 2010) keep the transition fixed and random, scaled to be contractive, and train only a linear readout of the resulting fixed point; the approach remains competitive on heterophilous node classification (Micheli & Tortorella, 2023). IterGNN (Tang et al., 2020) applies a shared message-passing operator for a learned number of steps and halts via a sigmoidal termination signal trained jointly with the model. In each of these models the prediction is read out of the latent state and does not enter the transition. Written in our notation, this explicit recurrence is exactly the ablation TRGNN^{-y} of Section 5, $H^{(t+1)} = f_\theta(G, [X_{\text{enc}} \,\|\, H^{(t)}])$, $y^{(t+1)} = g_\omega(\rho(H^{(t+1)}))$, instantiated with a modern GIN/GINE transition; Appendix F compares TRGNNs directly against the Scarselli model and graph echo state networks.

Implicit-layer GNNs. The implicit-layer view of deep networks, introduced by Bai et al. (2019), replaces a finite stack of distinct layers with the fixed point of a single weight-tied operator, solved via root-finding and differentiated through by the implicit function theorem. This construction provides effectively infinite depth at $O(1)$ training memory and underlies the implicit-GNN literature discussed below.

Implicit Graph Neural Networks (Gu et al., 2020) specialize this framework to message passing, solving for the equilibrium $H^* = \sigma(WH^*A + bX)$ of a shared-weight propagation. Subsequent work strengthens this construction along several axes: a monotone-operator reformulation (Baker et al., 2023) establishes well-posedness and contractivity guarantees that the original Perron–Frobenius parameterization does not, while EIGNN (Liu et al., 2021) obtains the equilibrium in closed form via spectral decomposition of the propagation operator, avoiding the iterative solver entirely. A parallel line specializes the implicit-layer framework to neural algorithmic reasoning (Georgiev et al., 2024), solving for the equilibrium of a problem-specific GNN rather than supervising per-step iteration counts. A differential-equation perspective on message passing, for example that of graph neural diffusion (Chamberlain et al., 2021), reaches related models from a continuous-depth angle, framing propagation as a learned ODE in which the integration horizon plays a role analogous to discrete depth.

TRGNNs share the weight-tying construction with all of the above but differ in two structural ways on which the analysis of Section 4 hinges. First, unlike the implicit-layer models, the iteration is trained explicitly via stop-gradient unrolling rather than implicitly via fixed-point differentiation, so the learned object is a one-step refinement operator iterated at test time, not a root-finding objective. Second, and more centrally, a persistent prediction state $y^{(t)}$ evolves jointly with the latent state $H^{(t)}$ and feeds back into the message-passing input (Equation 6), so that the head g_ω can preserve the current prediction once further propagation is uninformative (Proposition 4.1). The TRGNN^{-y} ablation of Section 5 retains weight tying and recursive iteration but removes this prediction feedback, isolating its empirical contribution; the ablated model keeps the topology-dependent convergence of the recursion (Section 4.2) but loses the anytime monotonicity reported in Figure 5 despite matching the recursive architecture otherwise.

Adaptive depth without weight tying. ADMP-GNN (Abbahaddou et al., 2025) attaches an exit to every layer of a fixed, untied stack and halts each node adaptively. Depth then varies per node, but parameter count and maximal receptive field remain fixed by the stack depth L. In TRGNNs, parameter count is independent of the iteration count and the receptive field grows with it (Section 3.1).

A.3 Over-smoothing, Over-squashing, and Elasticity-Window Mitigations

The two pathologies that bound the elasticity window of Equation 5 have been subject of significant study. Over-smoothing was first observed empirically as a depth-induced collapse of node-level representations (Li et al., 2018) and later formalized as exponential loss of expressive power (Oono & Suzuki, 2020) and as Dirichlet-energy contraction (Cai & Wang, 2020). We refer the interested reader to a recent survey (Rusch et al., 2023) for a comprehensive treatment. The dual pathology of over-squashing (the compression of long-range signals through structural bottlenecks) was identified in (Alon & Yahav, 2021) and later analyzed analytically, e.g. via discrete graph curvature (Topping et al., 2022). Notably, subsequent work clarifies oversmoothing's joint dependence on layer width, network depth, and graph topology (Di Giovanni et al., 2023).

A substantial methodological literature addresses these pathologies by widening the elasticity window for a given graph at fixed depth L. Skip and residual constructions (Xu et al., 2018; Chen et al., 2020) aggregate or directly inject earlier representations to combat depth-induced contraction. Propagation decoupling (Gasteiger et al., 2019) factors the model into a learnable transformation and a separate parameter-free personalized-PageRank propagation, allowing the receptive field to expand without compounding parameter depth. Normalization methods such as PairNorm (Zhao & Akoglu, 2020) explicitly counteract the collapse of inter-node distances induced by repeated aggregation. Finally, edge-perturbation methods such as DropEdge (Rong et al., 2020) regularize learned propagation by removing edges stochastically during training. A separate line targets the topology directly: curvature-based (Topping et al., 2022) and related spectral rewiring methods modify the graph before message passing is applied, alleviating bottlenecks at the source.

These approaches operate within the fixed-depth paradigm of standard message passing. That is, they enlarge the range of L at which a given dataset can be processed without over-smoothing or under-reaching, but the chosen L remains uniform across inputs. The mismatch identified in Section 2.1, that the elasticity window itself varies with topology across graphs in the same dataset, is therefore unaffected. TRGNNs take a complementary approach: rather than widening the window for a fixed depth, they decouple effective depth from architectural design and let it vary per input. Its

important to note that these two approaches are not in conflict. A wider per-graph window expands the range of admissible T for any given input, and both classes of construction can in principle be combined in a single architecture, a direction we discuss as future work in Section 6.

A.4 Graph Transformers and Topology-Free Attention

Graph transformer architectures relax the topological constraints of message passing by allowing attention over node sets larger than (and ultimately disjoint from) the immediate neighborhood. The first works in this direction restricted attention to local neighborhoods while augmenting node tokens with structural positional encodings (Dwivedi & Bresson, 2020). Subsequent designs adopted dense attention over the full node set, with graph structure reintroduced through learned attention biases (Ying et al., 2021). These design choices, that is, what to attend over and how to inject graph structure into a permutation-invariant operator, define the space of subsequent architectures.

Because attention itself carries no topological information, graph structure must be reintroduced through positional or structural encodings. Existing approaches differ chiefly in the choice of encoding: spectral encodings derived from the graph Laplacian (Kreuzer et al., 2021), random-walk and diffusion-based features (Mialon et al., 2021), and token-level constructions that treat both nodes and edges as inputs to a standard transformer (Kim et al., 2022). A complementary line studies expressivity of the resulting models in connection with deep-set and hypergraph generalizations (Kim et al., 2021). Recent work on learnable positional encodings (Kanatsoulis et al., 2025) makes the encoding itself trainable rather than a fixed preprocessing step.

A practical limitation of dense attention is its $O(N^2)$ cost in the number of nodes, prohibitive for large graphs. Thus, GNN-transformer hybrid approaches (Rampášek et al., 2022) interleaves message-passing layers with global attention. Sparse-attention variants (Shirzad et al., 2023) go further by replacing dense attention with subquadratic constructions based on simplifying signals from expander graphs and virtual nodes.

A.5 Iterative Algorithms on Graphs and Neural Algorithmic Reasoning

Classical algorithms on graphs are predominantly iterative: Belief propagation (Pearl, 1982; 2014) computes marginal posteriors by repeatedly exchanging message updates between adjacent variables until convergence. Label propagation (Zhu & Ghahramani, 2002) diffuses class assignments along edges by averaging over neighbors at each step. PageRank (Page et al., 1999) ranks nodes by iterating a weighted random-walk update to its stationary distribution. Each method maintains a state distributed over the graph, refines it through a local update rule applied uniformly across the graph, and reads the output off the (approximate) fixed point. The algorithm is, in a structural sense, the iteration.

A growing body of work studies the extent to which graph neural networks can serve as differentiable analogs of such algorithms. The neural algorithmic reasoning program (Veličković & Blundell, 2021) articulates this view explicitly by framing GNNs as parameterized executors of classical algorithms over learned input encodings. Algorithmic alignment (Xu et al., 2020) formalizes the conditions under which a neural network architecture matches the computational structure of a target algorithm and demonstrates that aligned architectures generalize substantially better. Subsequent work has made the structural parallel between message passing and dynamic programming more precise (Dudzik & Veličković, 2022).

TRGNNs inherit the structural template common to these works, i.e. a state distributed over the graph and refined through a shared local update rule applied recursively, but differ in the purpose of the iteration. Classical iterative algorithms target a known fixed point and the neural algorithmic reasoning literature trains a network to imitate the trajectory of a specific algorithm. TRGNNs instead lets the refinement dynamics be learned end-to-end against the task loss, with no reference algorithm and no explicit fixed-point objective. The iteration itself produces the prediction, rather than computing one for a separately defined output, and the convergence behavior characterized in Section 4 arises from the learned dynamics themselves.

A.6 Anytime Prediction and Test-Time Compute

The notion of an anytime algorithm, that is, one that produces a valid output at any stopping point with quality improving over time, was originally formulated for resource-bounded reasoning in the classical artificial intelligence literatuare (Zilberstein, 1996). In deep learning, the same idea is operationalized by multi-exit and early-exit architectures, which attach intermediate prediction heads at varying network depths and supervise them jointly during training (Huang et al., 2017; Lee & Shin, 2018; McGill & Perona, 2017; Teerapittayanon et al., 2016). Inference can then terminate early when a compute budget is exhausted or a confidence threshold is met. A more recent line of work in language modeling pushes the same compute–quality axis in the opposite direction: predictable scaling laws have been established for parameters and training compute (Kaplan et al., 2020; Hoffmann et al., 2022), and subsequent work demonstrates analogous predictability for test-time compute via mechanisms such as extended sampling, search, and inference-time reasoning (Snell et al., 2024; Muennighoff et al., 2025).

TRGNNs are anytime by construction: every outer iteration produces a valid prediction $y^{(t)}$, and theoretical analysis in Section 4.1 provides a principled basis for stopping at any iteration. The mechanism differs structurally from those above on three points. First, no architectural commitment is made to a finite set of exit depths: the prediction sequence is dense in iteration count T, supporting termination at arbitrary granularity. Second, no per-exit supervision is required: the architecture supports prediction preservation under standard single-loss training (Section 3.2). This is in contrast to the joint supervision of all intermediate heads required by deep multi-exit networks. Third, the decision of when to halt is not delegated to an external mechanism but emerges from the convergence dynamics of $\{y^{(t)}\}$ itself: once the latent state $H^{(t)}$ over-smooths, the prediction head can preserve its current output, so additional iteration neither helps nor hurts.

B Proofs

Proposition 4.1 (Non-Degradation). *Let $g_\omega : \mathbb{R}^{d_z} \times \mathbb{R}^{d_y} \to \mathbb{R}^{d_y}$ be the prediction head (Equation 7), parameterized as a universal approximator. Then for any compact $K \subset \mathbb{R}^{d_z} \times \mathbb{R}^{d_y}$ and any $\varepsilon > 0$, there exist parameters ω^* such that*

$$\sup_{(z,\, y) \in K} \|g_{\omega^*}(z, y) - y\| \;<\; \varepsilon.$$

Consequently, for any graph G and iteration t with $(\rho(H^{(t)}),\, y^{(t-1)}) \in K$, the prediction satisfies $\|y^{(t)} - y^{(t-1)}\| < \varepsilon$.

Proof. Define the projection $\pi_y : \mathbb{R}^{d_z} \times \mathbb{R}^{d_y} \to \mathbb{R}^{d_y}$ by $\pi_y(z, y) = y$. This map is continuous (in fact, linear) and K is compact. Since g_ω is a universal approximator, there exist parameters ω^* such that

$$\sup_{(z,\, y) \in K} \|g_{\omega^*}(z, y) - \pi_y(z, y)\| \;=\; \sup_{(z,\, y) \in K} \|g_{\omega^*}(z, y) - y\| \;<\; \varepsilon.$$

For any graph G and iteration t, the outer update (Equation 7) gives $y^{(t)} = g_{\omega^*}(\rho(H^{(t)}), y^{(t-1)})$. If $(\rho(H^{(t)}), y^{(t-1)}) \in K$, the bound above yields $\|y^{(t)} - y^{(t-1)}\| < \varepsilon$ directly. □

Proposition 4.2 (Topology-Dependent Convergence). *Let $\mathcal{S}_G := \mathbb{R}^{N \times d_h} \times \mathcal{Y}$ denote the coupled state space, with $\mathcal{Y} = \mathbb{R}^{d_y}$ (graph-level) or $\mathcal{Y} = \mathbb{R}^{N \times d_y}$ (node-level). Suppose $\mathcal{R}_G$ is a contraction on $\mathcal{S}_G$ with rate $\gamma(G) := \mathrm{Lip}(\mathcal{R}_G) < 1$. Then:*

- *(i) $\mathcal{R}_G$ admits a unique fixed point $(H^*(G), y^*(G)) \in \mathcal{S}_G$.*
- *(ii) The iterates converge geometrically: $\|(H^{(t)}, y^{(t)}) - (H^*, y^*)\| \;\le\; \gamma(G)^t \cdot D_0$, where $D_0 := \|(H^{(0)}, y^{(0)}) - (H^*, y^*)\|$.*
- *(iii) The effective depth satisfies $T_{\text{eff}}(G, \varepsilon) \le \left\lceil \frac{\log(D_0/\varepsilon)}{\log \gamma^{-1}(G)} \right\rceil$.*

Proof. Write $s^{(t)} := (H^{(t)}, y^{(t)}) \in \mathcal{S}_G$ for the coupled state at iteration t. Equip $\mathcal{S}_G$ with a product norm $\|\cdot\|$; as a finite-dimensional normed space, $\mathcal{S}_G$ is complete.

(i). $\mathcal{R}_G : \mathcal{S}_G \to \mathcal{S}_G$ is a contraction with rate $\gamma(G) < 1$ on a complete metric space. By the Banach fixed-point theorem, $\mathcal{R}_G$ admits a unique fixed point $s^* := (H^*(G), y^*(G)) \in \mathcal{S}_G$ satisfying $\mathcal{R}_G(s^*) = s^*$.

(ii). The contraction property and the fixed-point identity $\mathcal{R}_G(s^*) = s^*$ give

$$\|s^{(t)} - s^*\| \;=\; \|\mathcal{R}_G(s^{(t-1)}) - \mathcal{R}_G(s^*)\| \;\le\; \gamma(G)\,\|s^{(t-1)} - s^*\|.$$

Applying this bound by induction on t yields $\|s^{(t)} - s^*\| \le \gamma(G)^t\,\|s^{(0)} - s^*\| = \gamma(G)^t \cdot D_0$.

(iii). The canonical projection $(H, y) \mapsto y$ is non-expansive under the product norm, so

$$\|y^{(t)} - y^*(G)\| \;\le\; \|s^{(t)} - s^*\| \;\le\; \gamma(G)^t\, D_0.$$

The condition $\|y^{(t)} - y^*(G)\| < \varepsilon$ of Definition 4.1 is therefore satisfied whenever $\gamma(G)^t D_0 < \varepsilon$. Since $0 < \gamma(G) < 1$, taking logarithms and dividing by $\log \gamma(G) < 0$ reverses the inequality to $t > \log(\varepsilon/D_0)/\log \gamma(G)$. The smallest integer satisfying this bound gives $T_{\text{eff}}(G, \varepsilon) \le \left\lceil \frac{\log(D_0/\varepsilon)}{\log \gamma^{-1}(G)} \right\rceil$. □

C Experimental Details

Here, we provide dataset descriptions, detail our experimental procedure, and discuss necessary hardware and software to reproduce results from our empirical study.

C.1 Dataset Descriptions

Sections 4 and 5 of the main manuscript reports results on 7 different datasets/splits. Datasets are used in accordance to their respective terms of use. We provide dataset descriptions for each below.

Watts-Strogatz Graphs are sampled from the Watts-Strogatz model with $N = 64$ nodes and initial neighbor count $k = 4$, varying the rewiring probability across $p \in \{0.00, 0.05, 0.10, 0.20, 0.30, 0.50, 0.70, 1.00\}$. This interpolates continuously between ring lattices ($p = 0$; diameter $\delta \approx 16$, $|\lambda_2(\hat{A})| \approx 0.99$) and Erdos-Renyi–like random graphs ($p = 1$; $\delta \approx 4$, $|\lambda_2(\hat{A})| \approx 0.82$) and thus provides a controlled variation in both $T_{\text{reach}}(G)$ and $T_{\text{smooth}}(G)$. Disconnected draws are rejected and resampled to ensure all graphs are connected.

The task is node-level shortest-path regression from a single designated source. Each graph is equipped with a one-dimensional node feature $x_v \in 0, 1$ indicating whether v is the source, and the target y_v is the BFS hop distance from the source. There are no edge features. For each value of p, we independently generate 500 training, 100 validation, and 100 test graphs (4,000 / 800 / 800 total). All splits are shuffled across p values so that each mini-batch during training contains a mix of topologies. We note that $|\lambda_2(\hat{A})|$ denotes the second-largest eigenvalue of the symmetric normalized adjacency $D^{-1/2}AD^{-1/2}$ in absolute value; this distinction matters at low p, where the most negative eigenvalue can approach -1 in magnitude.

Reddit. The Reddit-Binary (Reddit-B) and Reddit-Multi-5K (Reddit-M) datasets (Yanardag & Vishwanathan, 2015) are graph classification benchmarks where each graph represents an online discussion thread: nodes correspond to users and edges to response relationships between comments. Reddit-B contains 2,000 graphs across 2 classes (question/answer-based vs. discussion-based communities); Reddit-M contains 5,000 graphs across 5 classes (community prediction). We evaluate using 10-fold cross-validation accuracy following standard protocol.

DrugOOD. DrugOOD (Ji et al., 2023) is an out-of-distribution benchmark for molecular binding affinity prediction. Each sample is a molecular graph with 39-dimensional node features (atomic properties) and 10-dimensional edge features (bond properties), labeled by binarized binding activity. We evaluate on three domain shifts that partition molecules by experimental assay (Assay), chemical scaffold (Scaffold), and molecular size (Size), testing generalization to unseen experimental conditions, structural motifs, and compound scales respectively. Each shift induces a five-way split (train, in-distribution validation/test, OOD validation/test). We report OOD AUROC.

Peptides-struct. Peptides-struct (Dwivedi et al., 2022) is a regression task from the Long Range Graph Benchmark. The dataset contains 15,535 molecular graphs (mean 150.94 nodes, mean diameter

Table 2: **Full numerical results for Section 5.1.** Notation follows Figure 3: Base-P is the baseline within $\pm 15\%$ of TRGNN's parameter count, Base-U is unconstrained. Reddit reports 10-fold mean accuracy $\pm$ std (%); DrugOOD reports OOD validation AUROC (%) across 5 runs. Reduction is $|\theta_{\text{Base-U}}|/|\theta_{\text{TRGNN}}|$. Scaffold, Assay, and Size refer to DrugOOD splits.

	TRGNN		**TRGNN^{-y}**		**Base-P**		**Base-U**	
Dataset	Perf.	$\lvert\theta\rvert$	Perf.	$\lvert\theta\rvert$	Perf.	$\lvert\theta\rvert$	Perf.	$\lvert\theta\rvert$
Reddit-B	93.0$\pm$1.5	5.5K	92.1$\pm$1.3	29.6K	91.6$\pm$2.5	5.5K	93.0$\pm$2.5	1.1M (205$\times$)
Reddit-M	57.4$\pm$2.8	8.0K	57.3$\pm$2.0	17.2K	46.4$\pm$2.4	5.6K	58.1$\pm$2.1	217.1K (27$\times$)
Scaffold	69.1$\pm$0.4	19.2K	67.9$\pm$0.5	32.6K	68.0$\pm$0.6	19.2K	68.7$\pm$0.5	49.8K (3$\times$)
Assay	71.8$\pm$0.8	26.3K	70.8$\pm$1.2	45.2K	70.5$\pm$1.1	32.7K	72.6$\pm$1.1	108.4K (4$\times$)
Size	66.6$\pm$0.8	49.8K	65.0$\pm$0.7	26.2K	66.3$\pm$0.8	49.8K	66.6$\pm$0.8	122.8K (2$\times$)

56.99) where nodes represent atoms and edges represent chemical bonds, featurized via 9- and 3-dimensional encoders, respectively. We use standard 70/15/15 train/validation/test split and report MAE.

C.2 Experimental Details

Controlled experiments. The TRGNN uses a GIN backbone with hidden dimension $d_h = 64$, block depth $L_{\text{block}} = 1$, inner steps $n = 1$, and $T = 128$ outer iterations, operating in node-level readout mode. The encoder is a two-layer MLP (Linear–ReLU–Linear) and the prediction head is a two-layer MLP with residual output $y^{(t)} = y^{(t-1)} + \Delta_\omega$. All GIN layers use `train_eps` (Fey & Lenssen, 2019) and are followed by LayerNorm and ReLU. Baselines are standard GINs at depths $L \in 4, 5$ with untied weights.Baseline hidden dimensions are chosen analytically to match the TRGNN's message-passing parameter count within $\sim 3\%$.

Training uses Adam (lr $= 10^{-3}$, no weight decay), batch size 32, MSE loss, gradient clipping at $|\cdot|2 = 1.0$, up to 300 epochs with early stopping (patience 30) on validation MAE. Stop-gradient unrolling follows Section 3.2: the first $T-1$ iterations are detached and only the final step carries gradients. At evaluation, $T_{\text{test}} = T_{\text{train}} = 128$ and each test graph is processed individually so that per-graph trajectories can be recorded. Effective depth $T_{\text{eff}}(G, \varepsilon)$ is measured at $\varepsilon = 0.05$ as the smallest iteration t at which the residual $|y^{(t)} - y^{(T)}|$ drops below ε times its initial value.

Architecture. All models in Sections 5.1–5.3 share the same model classes. TRGNNs use a two-layer encoder η_α (Linear–ReLU–Linear), a shared message-passing block f_θ consisting of L_{block} GIN or GINE layers, and a two-layer prediction head g_ω with residual output. Each GIN/GINE layer uses `train_eps`, and is followed by LayerNorm and ReLU. The backbone is GIN for Reddit (no edge features) and GINE for DrugOOD and Peptides-struct (with edge features). Reddit graphs carry no native node features; we use scalar node degree as the sole input ($d = 1$). DrugOOD edge features are encoded by a learned linear projection; Peptides-struct uses the OGB AtomEncoder and BondEncoder (Hu et al., 2020).

Standard GNN baselines use a single linear input projection, L GIN/GINE layers with untied weights, LayerNorm after each layer, and a two-layer prediction head. Readout pooling is sum-pool for Reddit and mean-pool for DrugOOD and Peptides-struct.

Hyperparameter search. Sections 5.1 and 5.2 tune all models via Bayesian hyperparameter sweeps(Weights & Biases (Biewald, 2020), method: `bayes`). Each (dataset, model type) pair has a separate sweep. For TRGNN and TRGNN^{-y} on Reddit, the search space is $d_h \in \{32, 48, 64\}$, $n \in \{2, 4, 6\}$, $T \in \{2, 3, 4, 5, 6, 7, 8\}$, $L_{\text{block}} \in \{1, 2\}$, learning rate $\in \{10^{-3}, 5 \times 10^{-4}, 10^{-4}\}$, dropout $\in \{0, 0.1, 0.2\}$, weight decay $\in \{0, 10^{-5}\}$; for DrugOOD, $d_h \in \{32, 48, 64, 80\}$ and weight decay $\in \{10^{-5}, 10^{-4}\}$, with the remaining axes identical. Baseline sweeps search $d_h \in \{32, 48, 64, 128, 256\}$ (Reddit) or $\{32, 48, 64, 80, 128\}$ (DrugOOD) and depth $L \in \{2, 4, 6, 8\}$. TRGNN^{-y} shares the TRGNN search space. Each sweep samples 30 configurations from the Bayesian surrogate.

For Peptides-struct (Section 5.3), we perform a grid search over 48 cells: $d_h \in \{96, 128, 160\}$, two depth schemes (effective depth 6: n=2, T=3, L_{block}=1; effective depth 16: n=2, T=4, L_{block}=2),

learning rate $\in \{10^{-3}, 3 \times 10^{-3}\}$, and dropout $\in \{0, 0.1\}$. All configurations fall under the LRGB (Dwivedi et al., 2022) 500K parameter budget.

Training protocol. On Reddit, each sweep configuration is evaluated via 10-fold stratified cross-validation over 330 epochs per fold. The optimizer is Adam with a StepLR schedule (factor 0.5 every 50 epochs) and gradient clipping ($|\cdot|_2 \leq 1.0$). Loss is binary cross-entropy (Reddit-B) or cross-entropy (Reddit-M). On DrugOOD, training runs for 100 epochs with Adam (no learning rate schedule) and gradient clipping ($|\cdot|_2 \leq 1.0$); loss is binary cross-entropy. Batch sizes are 32 (Reddit) and 16 (DrugOOD). All recursive models use the stop-gradient scheme of Section 3.2.

For the anytime evaluation of Section 5.2, the sweep-winning configuration for each (dataset, model type) pair is retrained with $T_{\text{train}} = 16$ outer iterations (the sole architectural override); all other hyperparameters are inherited from the sweep winner. Reddit retrains use a single fold (fold 0); DrugOOD retrains use a single seed.

On Peptides-struct, training runs for 200 epochs with Adam (weight decay 3×10^{-6} on non-bias parameters) and a linear warmup-then-linear-decay learning rate schedule (100 warmup steps). Batch size is 64 and loss is MAE on standardized targets.

Model selection and reporting. For Sections 5.1 and 5.2, the selection metric is 10-fold mean validation accuracy (Reddit) and OOD validation AUROC (DrugOOD). Four configurations are reported per dataset: **Base-U**, the unconstrained baseline sweep winner; **TRGNN**, the smallest-parameter recursive model whose selection metric equals or exceeds Base-U's (strict, no slack; falls back to argmax if no configuration dominates); **Base-P**, the best baseline whose parameter count lies within $\pm 15\%$ of TRGNN's; and **TRGNN**$^{-y}$, the argmax of its own sweep. On DrugOOD, hyperparameters and best-epoch checkpoints are both selected on OOD validation AUROC, and the same metric is reported; Reddit hyperparameters are selected on validation accuracy within each training fold.

For the anytime evaluation, each retrained model is evaluated at test-time iteration counts $T \in 1, 2, 3, 4, 6, 8, 12, 16$. For Peptides-struct, model selection is by best validation MAE, and test MAE at that epoch is reported. Section 5.3 baselines. Baseline results in Table 1 for R-PEARL, B-PEARL, GPS, SAN+RWSE, GNN-AK+, SUN, Graph ViT, and B-PEARL+SRF are taken from Kanatsoulis et al. (2025) and Hosseini et al. (2025).

C.3 Software and Compute Resources

All experiments were conducted on an AMD EPYC 7313P 16-Core Processor with 4× NVIDIA A40 GPUs (48GB memory each). While no single experiment requires multi-GPU training, multiple GPUs were used to run independent experiments in parallel. We estimate the empirical study support this manuscript required 500 GPU A40 GPU hours.

D Additional Experimental Results

D.1 Full Numerical Results Parameter Efficiency Experiments

Figure 3 presents parameter efficiency results as scatter plots. Table 2 reports the corresponding numerical values with error bars where applicable. Reddit error bars are standard deviations across the 10 cross-validation folds. DrugOOD error bars are standard deviations across five seeds.

Table 3: **Alternative backbones on Reddit.** 10-fold mean accuracy ± std (%). Each backbone uses the sweep-winning TRGNN configuration from Section 5.1 with no additional tuning.

Backbone	Reddit-B (%)	Reddit-M (%)
GIN (default)	$93.0_{\pm 1.5}$	$57.4_{\pm 2.8}$
GCN	$91.2_{\pm 1.7}$	$54.0_{\pm 2.1}$
GAT	$90.8_{\pm 2.8}$	$53.3_{\pm 2.4}$
GATv2	$90.0_{\pm 2.6}$	$53.9_{\pm 2.5}$

Table 4: **Runtime on Section 5.1 benchmarks.** Per-epoch training time and per-pass inference time (seconds, mean ± std over 10 timed epochs) on a single NVIDIA A40. Each method uses the sweep-winning configuration from Section 5.1. Scaffold, Assay, and Size refer to DrugOOD splits.

Dataset	Method	Train (s/epoch)	Inference (s/pass)
Reddit-B	TRGNN	$0.47_{\pm 0.04}$	$0.040_{\pm 0.000}$
	TRGNN^{-y}	$0.81_{\pm 0.05}$	$0.064_{\pm 0.003}$
	Base-P	$0.29_{\pm 0.00}$	$0.020_{\pm 0.000}$
	Base-U	$1.20_{\pm 0.01}$	$0.064_{\pm 0.000}$
Reddit-M	TRGNN	$1.67_{\pm 0.15}$	$0.133_{\pm 0.002}$
	TRGNN^{-y}	$2.62_{\pm 0.17}$	$0.186_{\pm 0.002}$
	Base-P	$0.71_{\pm 0.00}$	$0.049_{\pm 0.000}$
	Base-U	$1.55_{\pm 0.00}$	$0.091_{\pm 0.000}$
Scaffold	TRGNN	$17.9_{\pm 0.5}$	$10.4_{\pm 0.2}$
	TRGNN^{-y}	$9.0_{\pm 1.4}$	$3.7_{\pm 0.0}$
	Base-P	$6.6_{\pm 0.2}$	$2.5_{\pm 0.0}$
	Base-U	$6.6_{\pm 0.4}$	$2.5_{\pm 0.0}$
Assay	TRGNN	$72.0_{\pm 2.4}$	$22.5_{\pm 0.5}$
	TRGNN^{-y}	$22.9_{\pm 3.4}$	$5.1_{\pm 0.1}$
	Base-P	$10.2_{\pm 0.4}$	$2.5_{\pm 0.0}$
	Base-U	$33.7_{\pm 5.9}$	$4.7_{\pm 0.1}$
Size	TRGNN	$22.8_{\pm 1.1}$	$5.1_{\pm 0.1}$
	TRGNN^{-y}	$48.4_{\pm 4.8}$	$7.3_{\pm 0.2}$
	Base-P	$11.8_{\pm 0.8}$	$2.2_{\pm 0.0}$
	Base-U	$14.2_{\pm 2.7}$	$2.2_{\pm 0.0}$

D.2 Alternative Backbones

The experiments of Section 5.1 use GIN (Xu et al., 2019) as the message-passing backbone throughout. To verify that the recursive formulation is not specific to this choice, we evaluate TRGNN with GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), and GATv2 (Brody et al., 2022) backbones on the Reddit benchmarks. For each alternative backbone, we retain the hyperparameters of the GIN-based sweep winner (Section 5.1) and substitute only the message-passing operator; no additional tuning is performed.

Table 3 reports results. All three alternative backbones perform competitively despite the absence of backbone-specific hyperparameter tuning, with GCN within 2 points of GIN on both datasets. GIN remains the strongest choice, consistent with its design for maximizing discriminative power on graphs without edge features (Xu et al., 2019).

D.3 Runtime and Memory

As noted in Section 6, recursive architectures trade parameter count for sequential computation. Table 4 quantifies this tradeoff on the benchmarks of Section 5.1. TRGNN incurs higher per-epoch training time and per-pass inference time than Base-P due to its T sequential outer iterations, each requiring a forward pass through the shared message-passing block. The overhead varies with the sweep-selected iteration count: on DrugOOD-Assay, where TRGNN's winning configuration uses comparatively many iterations, training time is comparatively longer. On Reddit-B, where fewer iterations suffice, the ratio is $\sim 1.6\times$. Relative to Base-U, TRGNN's runtime is comparable or lower on Reddit (where Base-U's large parameter count dominates) and higher on DrugOOD (where TRGNN's iteration cost exceeds Base-U's wider but shallower forward pass). The stop-gradient scheme of Section 3.2 ensures that training memory is constant in T; accordingly, peak GPU memory remains below 0.1 GB for TRGNN across all datasets.

Table 5 compares TRGNN against R-PEARL and B-PEARL (Kanatsoulis et al., 2025) on Peptides-struct using the open-source implementation of Kanatsoulis et al. (2025). TRGNN is $\sim 20\times$ faster per epoch than R-PEARL and $\sim 32\times$ faster than B-PEARL, while requiring $\sim 50\times$ less GPU memory. The disparity arises because PEARL computes learnable positional encodings via eigendecomposition

Table 5: **Runtime and memory on Peptides-struct.** Per-epoch wall time (seconds, mean ± std over 10 timed epochs) and peak GPU memory on a single NVIDIA A40. All methods use batch size 25. R-PEARL and B-PEARL use the open-source implementation of Kanatsoulis et al. (2025).

Method	Time/epoch (s)	Peak GPU memory (GB)
TRGNN	$19.3_{\pm 0.2}$	0.55
R-PEARL	$399.3_{\pm 36.1}$	28.56
B-PEARL	$623.1_{\pm 0.9}$	24.73

of the graph Laplacian at each forward pass, a cost that dominates training time on the moderately sized graphs of Peptides-struct. TRGNN's recursive iterations, while sequential, operate entirely within the message-passing framework and incur no such auxiliary computation.

E Topology Dependence of the Contraction Rate

Proposition 4.2 bounds effective depth in terms of the contraction rate $\gamma(G)$ of the coupled map, but does not relate $\gamma(G)$ to the over-smoothing rate $\kappa(G)$ of Equation 3. For arbitrary learned non-linear dynamics no universal relation exists, since $\gamma(G)$ depends on the learned operator as well as on the graph. This appendix gives a restricted result: for normalized-adjacency message passing on the subspace in which over-smoothing takes place, the topology-dependent component of the contraction rate is nondecreasing in $\kappa(G)$.

Setting. Let $\hat{A} = \hat{A}_G$ be the symmetric normalized adjacency of a connected graph, with eigenvalues $1 = \lambda_1 > |\lambda_2| \geq \cdots$, and let $\mu(G) := \max_{i \geq 2} |\lambda_i(\hat{A})|$. Write $\mathcal{U}_G^{\perp}$ for the orthogonal complement of the principal eigenvector of $\hat{A}$, i.e., the subspace in which node features are not constant. We use the normalized Dirichlet energy $\mathcal{E}_N(H) := \mathrm{Tr}\big(H^\top (I - \hat{A})H\big)$, which coincides with Equation 2 up to degree normalization.

Proposition E.1 (κ–γ certificate)**.**

(i) ***(Linear propagation.)*** *For H with columns in $\mathcal{U}_G^{\perp}$, $\mathcal{E}_N(\hat{A}H) \leq \mu(G)^2\, \mathcal{E}_N(H)$. Thus Equation 3 holds for $H' = \hat{A}H$ with $\kappa(G) = \mu(G)^2$.*

(ii) ***(Message-passing component.)*** *Let the block f_θ consist of $m := n \cdot L_{\mathrm{block}}$ steps of the form $H \mapsto \sigma(\hat{A}HW + B)$ with σ 1-Lipschitz, $\|W\|_2 \leq s$ for a graph-independent s, and B collecting the contributions of X_{enc} and $\bar{y}$. Suppose the iteration leaves $\mathcal{U}_G^{\perp}$ invariant. Then the hidden-to-hidden partial Lipschitz constant of one outer iteration restricted to $\mathcal{U}_G^{\perp}$ satisfies*

$$a_G \;\leq\; \big(s\,\mu(G)\big)^m \;=\; s^m\, \kappa(G)^{m/2}.$$

(iii) ***(Coupled rate.)*** *Let b, c, d be graph-uniform bounds on the partial Lipschitz constants of $\bar{y} \mapsto H$ (through f_θ), $H \mapsto y$ and $y \mapsto y$ (through g_ω and ρ). Then the errors $e_H := \|H - H^*\|$, $e_y := \|y - y^*\|$ on $\mathcal{U}_G^{\perp}$ satisfy, componentwise,*

$$\begin{pmatrix} e'_H \\ e'_y \end{pmatrix} \;\preceq\; M_G \begin{pmatrix} e_H \\ e_y \end{pmatrix}, \qquad M_G := \begin{pmatrix} s^m \kappa(G)^{m/2} & b \\ c\, s^m \kappa(G)^{m/2} & c\, b + d \end{pmatrix},$$

and the topology-dependent component of the coupled contraction rate, $\bar{\gamma}^{\perp}(G) := \|M_G\|_2$, is nondecreasing in $\kappa(G)$.

Proof. (i). Diagonalize $\hat{A} = \sum_i \lambda_i u_i u_i^\top$. For a column $h \in \mathcal{U}_G^{\perp}$ write $h = \sum_{i \geq 2} \alpha_i u_i$; then $\mathcal{E}_N(h) = \sum_{i \geq 2}(1 - \lambda_i)\alpha_i^2$ and $\mathcal{E}_N(\hat{A}h) = \sum_{i \geq 2}(1 - \lambda_i)\lambda_i^2 \alpha_i^2 \leq \mu(G)^2 \sum_{i \geq 2}(1 - \lambda_i)\alpha_i^2$. Sum over columns.

(ii). On $\mathcal{U}_G^{\perp}$, $\|\hat{A}\|_2 = \mu(G)$. For a single step, $\|\sigma(\hat{A}H_1W + B) - \sigma(\hat{A}H_2W + B)\|_F \leq \|\hat{A}(H_1 - H_2)W\|_F \leq \mu(G)\, s\, \|H_1 - H_2\|_F$, using that σ is 1-Lipschitz, that B is common to both arguments,

and invariance of $\mathcal{U}_G^{\perp}$. Composing m steps gives $a_G \le (s\mu(G))^m$; substitute $\mu(G) = \kappa(G)^{1/2}$ from (i).

(iii). One outer iteration maps $(H, y) \mapsto (H', y')$ with $H' = f_\theta(G, [X_{\text{enc}} \| H \| \bar{y}])$ and $y' = g_\omega(\rho(H'), y)$. By the partial Lipschitz bounds, $e'_H \le a_G e_H + b\, e_y$ and $e'_y \le c\, e'_H + d\, e_y \le c\, a_G e_H + (c\, b + d)\, e_y$, which is the stated componentwise inequality with $a_G \le s^m \kappa(G)^{m/2}$ substituted. M_G is entrywise nonnegative, and each entry is nondecreasing in $\kappa(G)$; the spectral norm of a nonnegative matrix is nondecreasing in each entry (by Perron–Frobenius applied to $M_G^\top M_G$), so $\bar{\gamma}^{\perp}(G)$ is nondecreasing in $\kappa(G)$. □

Remark E.1 (Scope)**.** Proposition E.1 bounds the recurrence on $\mathcal{U}_G^{\perp}$, the subspace in which oversmoothing takes place; it does not bound the principal (constant) mode or its coupling to $\mathcal{U}_G^{\perp}$, and therefore does not by itself bound the full-state Lipschitz constant $\mathrm{Lip}(\mathcal{R}_G)$ of Proposition 4.2. If the bound on the principal mode and the coupling also yield a full-state rate below one, the hypothesis of Proposition 4.2 holds and effective depth is ordered by $\kappa(G)$ as in Definition 4.1. The subspace-invariance assumption holds exactly for linear propagation and for propagation with degree-normalized biases, and approximately otherwise. The result concerns the architecture, not the trained parameters: we neither impose the weight-norm bound s nor verify contractivity in the models of Section 5. It establishes that the empirical spectral dependence of Section 4.2 is not incidental, but it does not constitute a guarantee for the learned non-linear model.

F Comparison with Classical Recurrent GNNs

Section 2.2 positions TRGNNs against the recurrent GNN lineage. Here we compare directly against the two graph-level members of that lineage, the contractive recurrent GNN of Scarselli et al. (2009) and the graph echo state network (Gallicchio & Micheli, 2010), on Reddit-B and Reddit-M. ADMP-GNN (Abbahaddou et al., 2025) is a node-classification method whose exit and aggregation mechanism has no graph-level counterpart, so it is not included.

Baselines. The *Scarselli* model iterates a weight-shared MLP transition with sum aggregation for n_{iter} steps under enforced contraction (the transition is rescaled so that $\|A\|_2 \cdot \max \deg \cdot \|W\|_2 \le 0.9$), since without contractivity the fixed point of the original model need not exist; $n_{\text{iter}} \in \{5, 10, 20\}$ is selected on validation, and validation accuracy saturates over this range, so the fixed count operates in the converged regime. Gradients are taken by backpropagation through the unroll rather than by the Almeida–Pineda recurrence, which does not handicap the baseline. *GESN* uses a fixed random reservoir of $\{256, 512, 1024\}$ units (selected on validation per fold) and a ridge-regression readout trained in closed form. TRGNN and TRGNN^{-y} use the configurations of Section 5.2.

Protocol. Stratified 10-fold cross-validation with 10% of each training fold held out for selection, so that test accuracy is read only at the selected configuration and epoch. Accuracy is mean ± std over folds; *vs. TRGNN* is TRGNN's accuracy gain, significant for every baseline under a paired t-test on matched folds ($p \le 0.02$). Cost is one warm-up plus 10 timed passes on a single NVIDIA H100 (the timings of Table 4 were measured on an A40 and are not directly comparable).

Tradeoffs. TRGNN attains the highest mean accuracy on both datasets, with every margin significant under a paired test on matched folds, and the lowest peak memory. The baselines are cheaper per inference pass: the Scarselli transition is a single weight-shared map with no prediction head in the loop, and the GESN readout is linear and untrained. GESN buys this with a 256–1024-unit reservoir that must be materialized at every iteration, hence 8–50× more stored weights than TRGNN; it also fits in seconds, since only the readout is trained. The Scarselli model illustrates the cost of constitutive contractivity: relaxing the contraction constraint in our implementation (the design point of gated graph neural networks (Li et al., 2016)) raises its accuracy to 85.0 and 46.0, still below TRGNN^{-y}, which is the same latent-only recurrence with a modern transition and no contraction constraint. TRGNN does not require the latent recurrence to be contractive; Proposition 4.2 gives contractivity of the coupled map as a sufficient condition for convergence, not a constraint imposed in training, which lets the prediction stabilize before the latent state has converged. Overall, TRGNN trades sequential computation for higher accuracy at substantially lower memory and storage.

Table 6: **Classical recurrent GNN baselines on Reddit.** 10-fold test accuracy (%), accuracy gain of TRGNN, trainable parameters, parameters stored at inference, hidden state width, inference time per pass over the test fold, and peak GPU memory. TRGNN leads on accuracy and peak memory on both datasets; the baselines are cheaper per inference pass, so we claim comparable rather than lower inference cost.

Dataset	Model	Test acc. (%)	vs. TRGNN	Trained $\lvert\theta\rvert$	Stored $\lvert\theta\rvert$	d_h	Inf. (s)	Peak (GB)
Reddit-B	TRGNN	**92.5**$\pm$2.4	—	21,186	21,186	64	0.121	**0.315**
	TRGNN^{-y}	91.9$\pm$1.7	—	21,122	21,122	64	0.056	0.190
	GESN	88.7$\pm$2.4	+3.8	2,050	1,051,650	1024	0.153	1.312
	Scarselli	75.5$\pm$4.1	+17.0	16,834	16,834	64	0.033	0.548
Reddit-M	TRGNN	**56.6**$\pm$2.2	—	8,039	8,039	32	0.261	**0.163**
	TRGNN^{-y}	51.5$\pm$9.0	—	7,879	7,879	32	0.576	0.332
	GESN	55.0$\pm$2.2	+1.6	1,285	67,077	256	0.087	0.532
	Scarselli	45.5$\pm$3.1	+11.0	66,821	66,821	128	0.115	0.920

G Anytime Inference: Variability and Adaptive Stopping

Monotonicity across folds and seeds. Table 7 repeats the monotonicity measurement of Figure 5 over all 10 Reddit folds and 3 DrugOOD seeds. TRGNN's Spearman correlation between iteration index and performance is near one with small spread on every dataset; the ablation's mean is lower everywhere and its spread is large, reaching 0.37 on Reddit-M.

Table 7: **Anytime monotonicity across folds/seeds.** Spearman ρ between iteration index t and performance $y^{(t)}$, mean $\pm$ std over n folds (Reddit) or seeds (DrugOOD).

Dataset	n	TRGNN	TRGNN^{-y}	$\Delta\rho$
Reddit-B	10	0.983 ± 0.024	0.820 ± 0.109	+0.163
Reddit-M	10	0.946 ± 0.089	0.126 ± 0.374	+0.820
DrugOOD-Assay	3	0.984 ± 0.014	0.563 ± 0.340	+0.421
DrugOOD-Scaffold	3	1.000 ± 0.000	0.841 ± 0.131	+0.159
DrugOOD-Size	3	1.000 ± 0.000	0.730 ± 0.099	+0.270

Adaptive stopping. Our claim is topology-adaptive effective depth, measured per graph as in Figure 2(c), not that the implementation automatically saves computation: all inputs in the main experiments run for a fixed maximum T, and anytime inference permits earlier stopping. To make the benefit of an adaptive schedule concrete, we evaluate the simplest prediction-residual rule, halting graph G at the first t with $\|y^{(t)} - y^{(t-1)}\| < \tau$, on the trained model and test graphs of Section 4.2. Table 8 reports the resulting accuracy–compute tradeoff. Even under this heuristic, computation is reallocated across inputs from a single trained model with no per-exit heads and no retraining: at $\tau = 0.01$ per-graph iteration counts span 28–128 and the halting time correlates with $|\lambda_2(\hat{A})|$ ($r = 0.78$), and at matched accuracy the rule uses 36% fewer iterations than the best fixed-T schedule. Better stopping rules are a direction for future work.

Table 8: **Prediction-residual stopping on the controlled task.** For each threshold τ: mean and range of per-graph iterations, test MAE, the number of fixed-T iterations needed for equal MAE, the resulting saving, and the correlation of halting time with $|\lambda_2(\hat{A})|$.

τ	Mean iters	Range	MAE	Fixed-T for equal MAE	Saving	r(halt, $\lvert\lambda_2\rvert$)
0.0032	112.1	46–128	0.170	126	1.12×	0.52
0.0056	97.7	34–128	0.185	124	1.27×	0.66
0.0100	76.3	28–128	0.246	120	1.57×	0.78