CAN GRAPH FOUNDATION MODELS GENERALIZE OVER ARCHITECTURE?

Benjamin Gutteridge[1,*], **Michael Bronstein**[1,2] **& Xiaowen Dong**[1]
[1]University of Oxford, [2]AITHYRA

ABSTRACT

Graph foundation models (GFMs) have recently attracted interest due to the promise of graph neural network (GNN) architectures that generalize zero-shot across graphs of arbitrary scales, feature dimensions, and domains. While existing work has demonstrated this ability empirically across diverse real-world benchmarks, these tasks share a crucial hidden limitation: they admit a narrow set of effective GNN architectures. In particular, current domain-agnostic GFMs rely on fixed architectural backbones, implicitly assuming that a single message-passing regime suffices across tasks. In this paper, we argue that *architecture adaptivity* is a necessary requirement for true GFMs. We show that existing approaches are non-robust to task-dependent architectural attributes and, as a case study, use *range* as a minimal and measurable axis along which this limitation becomes explicit. With theoretical analysis and controlled synthetic experiments, we demonstrate that fixed-backbone GFMs provably under-reach on tasks whose architectural requirements differ from those seen at training time. To address this issue, we introduce a framework that adapts effective GNN architecture at inference time by discovering and mixing task-specific linear graph operators, enabling zero-shot generalization across tasks with heterogeneous architectural requirements, without retraining. We validate our approach on arbitrary-range synthetic tasks and a suite of real-world benchmarks, demonstrating improved performance and robustness over existing domain-agnostic GFMs.[1]

1 INTRODUCTION

The remarkable success of foundation models in domains such as computer vision (Dosovitskiy et al., 2020; He et al., 2022; Radford et al., 2021; Wang et al., 2023c) and natural language processing (Devlin, 2018; Bubeck et al., 2023; Touvron et al., 2023) has sparked interest in graph foundation models (GFMs) as a means of achieving similar gains for generating and reasoning over graph-structured data. Several GFMs have been proposed towards this goal, but most of these are domain-specific, in that they are designed to handle multiple graph tasks on highly inter-related data (Wang et al., 2025), such as knowledge graphs, molecules or transport networks (Wu et al., 2025; Huang et al., 2024; Wang et al., 2023a; Galkin et al., 2023). Indeed, the lack of a shared 'vocabulary' among domains is one reason why effective *domain-agnostic* GFMs remain elusive. Nevertheless, several recent works have made progress in this direction; approaches include inference-time selection among domain experts (Xia & Huang, 2024), conversion of graphs to tabular format to leverage tabular foundation models (Eremeev et al., 2025; Hayler et al., 2025), and GNNs hybridized with large language models (LLMs) (Liu et al., 2025). The latter may use LLMs as feature encoders on backbone GNNs (Zhu et al., 2024), use GNNs as LLM 'tools' (Tang et al., 2024), or even use LLMs end-to-end on token or text representations of graphs (Perozzi et al., 2024).

In this paper, we focus our attention on works that tackle GFMs within the geometric learning paradigm, i.e., respecting the graph structure and symmetry principles offered by GNNs. Within this space, there have been two major contributions, both of which are focused on inductive node classification as a zero-shot 'inpainting' problem; i.e. given a graph and some reference node labels,

[*]Correspondence to beng@robots.ox.ac.uk
[1]Reproducible code implementation at https://github.com/BenGutteridge/GOBLIN.

classify the remaining nodes. The first, GraphAny (Zhao et al., 2024), can be interpreted as a node-wise mixture-of-experts across a fixed basis of linear GNNs. The second, Finkelshtein et al. (2025), builds on this, extending symmetry-aware architectures (Zaheer et al., 2017; Segol & Lipman, 2019) to formalize the 'triple symmetry' required for zero-shot GFM inference: node equivariance, label equivariance, and feature *in*variance.

Both GraphAny and triple-symmetry (TS-)GNNs are domain-agnostic, and both demonstrate their generalization on a set of real-world benchmarks that are diverse in graph size, topology, domain, task homophily/heterophily, feature dimension, and number of classes. However, in using these standard graph benchmarks to assess generalization capability, these methods end up strongly overfitting in a crucial area: backbone *GNN architecture*.

Existing domain-agnostic GFMs assume a fixed architecture; generalization to tasks beyond the capabilities of that architecture is impossible without retraining.

This implicit overfitting becomes clear when one considers the benchmarks used by these works. While they are diverse in certain ways (e.g., homophilic vs heterophilic), they are arguably all short-range tasks (Bamberger et al., 2025; Arnaiz-Rodriguez & Errica, 2025). Specifically, there is nothing to suggest that any of these tasks requires an approach more sophisticated than local message passing—and this is reflected in the underlying architectures: both GraphAny and TS-GNN are evaluated using MPNN backbones of only *two layers*.

In this paper, we argue that this limitation is not specific to the benchmarks considered, but is structural: fixed-backbone GFMs cannot, by construction, adapt to tasks whose architectural requirements differ from those implicitly encoded in their design. More broadly, we argue that *architecture adaptivity* is a necessary requirement for GFMs.

To make this limitation precise, we focus on *range* as a minimal and measurable architectural attribute. Range captures the distance (according to some metric on a graph, such as geodesic distance) over which node interactions must be aggregated in order to solve a task. While range is only one of many potential architectural axes along which GNNs may differ, it has the advantage of admitting formal analysis and clean counterexamples. Furthermore, range and long-range interactions are a topic of increasing interest for GNN researchers, with several long-range benchmarks and measurements emerging in recent years Bamberger et al. (2025); Liang et al. (2025); Miglior et al. (2025); Zhou et al. (2025); Attali et al. (2026).

We show that existing domain-agnostic GFMs are provably bounded in range by their fixed operator bases or fixed message-passing depth, and therefore may under-reach on tasks that exceed these bounds (Section 3). We demonstrate this failure mode empirically using a simple synthetic node classification task with controllable range, for which existing methods collapse even under generous operator expansions (Section 4). Crucially, we show that this failure cannot be resolved by additive architectural changes—such as additional layers or expanded operator bases—without degrading performance on standard benchmarks. This highlights a fundamental tension: *there is no single GNN architecture that performs well across tasks with heterogeneous architectural requirements*.

Motivated by this observation, we introduce **G**raph **O**perator **B**asis **L**earning and **IN**ference (**GOBLIN**), a GFM framework for inference-time architecture adaptation (Section 5). Rather than fixing a message-passing architecture in advance, GOBLIN discovers a task-appropriate basis of linear graph operators at inference time, using only the labeled subset of nodes in the target graph (the same node label 'inpainting' task of Zhao et al. (2024) and Finkelshtein et al. (2025)). These operators are then combined via a permutation-invariant mixture-of-experts, enabling zero-shot adaptation to task-specific architectural requirements without retraining or backpropagation on the target graph. While our analysis and experiments focus on range as a diagnostic axis, the proposed framework is not range-specific: the same inference-time basis learning mechanism applies to any parameterized family of graph operators encoding architectural choices. We validate the zero-shot generalization capabilities of GOBLIN on arbitrary-range synthetic tasks, as well as on a suite of real-world benchmarks, including long-range tasks (Liang et al., 2025), and demonstrate improved performance over existing domain-agnostic GFMs (Sections 4 & 6) and standard MPNNs trained end-to-end.

2 PROBLEM FORMULATION AND BACKGROUND

Problem setting. We consider *zero-shot, transductive node classification* on a single graph, following the setting of existing domain-agnostic GFMs (Zhao et al., 2024; Finkelshtein et al., 2025). Let $G = (V, E)$ be a graph with $|V| = N$ nodes, adjacency matrix $A \in \mathbb{R}^{N \times N}$, and node features $X \in \mathbb{R}^{N \times d}$. Each node $u \in V$ is associated with a (possibly unknown) label $Y_u \in \{1, \ldots, C\}$. A subset of nodes $V_L \subset V$ is labeled, with corresponding labels $Y_L \in \mathbb{R}^{N_L \times C}$ (represented as one-hot vectors), while the remaining nodes $V_U = V \setminus V_L$ are unlabeled. The task is to predict labels for all nodes in V_U given (A, X, Y_L), without retraining or fine-tuning on the target graph.

2.1 GRAPHANY

GraphAny can be interpreted as a per-node mixture of experts selector with a fixed operator basis. With GOBLIN, we will extend this pipeline to include adaptive operator selection.

Let us define a linear GNN, $\hat{Y} = SXW$, with an analytic solution given by $W^* = (SX)^+_L Y_L$ (where $^+$ denotes the pseudo-inverse). We can think of $\hat{Y}$as a cheap (i.e. training-free) GNN 'expert', defined entirely by the choice of graph operator $S \in \mathbb{R}^{N \times N}$, where S likely encodes some function of the adjacency A and likely represents some form of message passing. For example, $S = I$ is a linear layer with no mixing, and $S = A^k$ is a k-layer linear GNN.

By choosing a fixed operator basis $\hat{\boldsymbol{Y}} = \{\hat{Y}^{(i)}\}_{i=1}^t$, we can define a GraphAny-style per-node mixture-of-experts as a summed attention weighting $\bar{Y}_u = \sum_{i=1}^t \alpha_u^{(i)} \hat{Y}_u^{(i)}$ over a given basis, where $\alpha_u = [\alpha_u^{(1)}, \alpha_u^{(2)}, \ldots, \alpha_u^{(t)}] = f_\theta(\boldsymbol{P}_u)$, and $\boldsymbol{P}_u$ denotes node-level distance features derived from $\hat{\boldsymbol{Y}}$. To ensure feature-permutation invariance, GraphAny uses node-wise linear GNN difference functions $||\hat{Y}_u^{(i)} - \hat{Y}_u^{(j)}||^2$, i.e., $t(t-1)$ features in total.[2] f_θ is an MLP that outputs per-expert logits and is trained on one or more graphs to learn to weight experts from similarity features.

Once trained, this model can be applied at inference time, zero-shot, to new graphs of arbitrary size and feature/class dimension, with no retraining and only t analytic linear GNN solves. However, a trained GraphAny model is *locked to the operator basis it was trained on*, and therefore must share any shortcomings imposed by that basis. GraphAny uses $\mathcal{S} = \{S^{(1)}, \ldots S^{(t)}\} = \{I, A, A^2, (I - A), (I - A)^2\}$ (degree-normalized adjacency), i.e., a linear term plus two 'low-pass' and two 'high-pass' terms. The effective architecture, therefore, is a mixture of 0-, 1- and 2-hop message passing.

2.2 TRIPLE-SYMMETRY GNNS

TS-GNNs extend the GraphAny framework by formalizing GFM symmetry requirements. Specifically, TS-GNNs enforce *node equivariance*, *label equivariance*, and *feature invariance*, and implement these constraints using Deep Set constructions over nodes, features, and labels. In contrast to GraphAny's explicit mixture-of-experts formulation, TS-GNNs are trained end-to-end with stochastic gradient descent.

Despite these architectural differences, TS-GNNs share a key limitation with GraphAny: they rely on a *fixed message-passing backbone*. In practice, all reported TS-GNN results use shallow architectures with at most two local message-passing layers. As a result, TS-GNNs are subject to the same under-reaching constraints as other fixed-depth GNNs: information cannot propagate beyond the receptive field determined by the chosen depth and aggregation operators.

In the following sections, we show that architectural rigidity can be explicitly demonstrated through the lens of task range, and that overcoming it requires inference-time architectural adaptation.

2.3 LIMITATIONS OF FIXED-BACKBONE GFMS

Both GraphAny and TS-GNN are based on message-passing architectures with a fixed depth of, at most, two layers, and must collapse via under-reaching when evaluated on tasks requiring anything

[2] Zhao et al. (2024) entropy-normalize features to ensure robustness to class size; for simplicity, we use squared distance features.

beyond this. One could argue that there are trivial solutions to this: use a deep GNN or graph Transformer (GT; Dwivedi & Bresson (2020)) backbone for TS-GNN, or use an alternative or more comprehensive basis for GraphAny. But this highlights the crucial shortcoming of both methods: any alternative architecture is still *fixed*, and there is no 'one-size-fits-all' architecture. As we see in Section 4, expanding the basis of GraphAny can help for some tasks, but introduces noise and hurts performance for others. Similarly, using a deeper GNN, or a GT, often introduces needless computational complexity for many tasks (Tönshoff et al., 2023), and can force performance trade-off against different types of tasks, often due to over-smoothing/-squashing and vanishing gradient effects (Oono & Suzuki, 2019; Alon & Yahav, 2020; Di Giovanni et al., 2023; Arroyo et al., 2025).

Put simply, a true GFM must be robust across tasks that necessitate architectural differences. Indeed, the entire motivation for developing GFMs is to avoid expensive and lengthy hyperparameter and architecture searches.[3] Existing methods do not address this; *they can only generalize to tasks that benefit from a narrow range of GNN architecture parameterizations.* Our method addresses it via adaptive basis (and therefore adaptive effective architecture) selection at inference time.

3 Range as a diagnostic for architectural limitations

In this section we formally demonstrate the rigidity of fixed-architecture GFMs, using range as a proxy. We focus on GraphAny, showing that the overall range of GraphAny can be studied by deriving the ranges of basis operators. We use the node-level range measure ρ_u from Bamberger et al. (2025): given some distance metric on a graph, such as geodesic distance, range, informally, is distance-weighted pairwise node sensitivity, averaged per node. Formally,

$$\rho_u = \frac{\sum_{v \in V} J_{uv} \cdot d_G(u, v)}{\sum_{v \in V} J_{uv}} \quad \text{where } J_{uv} = \left\| \frac{\partial \phi(X)_u}{\partial X_v} \right\|_1, \tag{1}$$

and ϕ is some operator mapping node input features to output features, such as a GNN.

Range of GraphAny. As GraphAny is a weighting of linear GNNs, we can study its overall range by computing the range of individual linear GNN experts and comparing their relative weights. The Jacobian of a linear GNN decomposes as $\frac{\partial \hat{Y}}{\partial X} = \frac{\partial SX}{\partial X} W + SX \frac{\partial W}{\partial X}$. The second term requires computational means to obtain due to the pseudo-inverse, and only encodes label-fitting, not node feature mixing; we therefore focus on only the first term. To put it another way, we assume that the weights of a solved linear GNN are fixed, so $\frac{\partial W}{\partial X} = 0$ (see Appendix C for a detailed discussion of this decision). Given this, we can derive the sensitivity of an output node logit (u, c) to an input node feature (v, d'): $\frac{\partial \hat{Y}_{uc}}{\partial X_{vd'}} = \frac{\partial}{\partial X_{vd'}} \left(\sum_{w=1}^{N} \sum_{k=1}^{d} S_{uw} X_{wk} W_{kc} \right) = S_{uv} W_{d'c}$. With reference to Eq. 1, here $J_{uv} = |S_{uv} W_{d'c}|$, and we can thus obtain the node-level range:

$$\tilde{\rho}_u = \left(\sum_{d'=1}^{d} \sum_{c=1}^{C} |W_{d'c}| \right) \left(\sum_{v \in V} |S_{uv}| \, d_G(u, v) \right) = \|W\|_1 \cdot \sum_{v \in V} |S_{uv}| \, d_G(u, v),$$
$$\text{normalizing to} \quad \rho_u = \frac{\sum_{v \in V} |S_{uv}| \, d_G(u, v)}{\sum_{v \in V} |S_{uv}|}. \tag{2}$$

Therefore, the range of a linear GNN depends only on the effective architecture, S, and the graph topology A (via d_G, and possibly S), *not* on model weights.

GraphAny uses the degree-normalized adjacency for its operators, and as this A has rows which sum to 1 (2, for the Laplacian), we obtain interpretable ranges/range upper bounds for given operators: $S = I \rightarrow \rho_u = 0$, $S = A \rightarrow \rho_u = 1$; $S = A_k$ where $(A_k)_{ij} = 1[d_G(i, j) = k] \rightarrow \rho_u = k$; $S = (I - A) \rightarrow \rho_u = 0.5$, and $S = A^k \rightarrow \rho_u \leq k$. To summarize, of the five operators used by GraphAny, three of them have range that is invariant to topology $(0, 1, 0.5)$ and the other two are upper bounded by 2 hops. This illustrates a key shortcoming in generalization. GraphAny

[3] Neural architecture search (Nunes & Pappa, 2020; Wang et al., 2023b; Dong et al., 2025) is another research area of note that is concerned with addressing this problem.

with a basis comprised of $S = A^k$ operators can have a maximum range of k, and in practice, this is likely to be much lower, especially given that attention weight is shared between experts with ranges guaranteed to be ≤ 1. With its standard basis, GraphAny *cannot solve tasks with range* > 2.

TS-GNNs. We cannot derive range analytically here, but the same conclusion holds; assuming an ℓ-layer message passing backbone, $\frac{\partial \hat{Y}_{uc}}{\partial X_{vd'}} = 0$ for all $\{u, v \in V : d_G(u, v) > \ell\}$, therefore $\rho_u \leq \ell$.

Motivation for using range as a diagnostic. While range is not the only task property that requires architectural adaptation, it is a useful and elegant proxy here. Unlike task properties such as label homophily, range has an intuitive meaning as a property of both tasks and architectures (and, indeed, architectures *applied to* tasks), depending on whether ϕ is a task operator or a model. This measurable relationship between task and architectural properties makes range, as well as *known-range synthetic tasks*, useful diagnostic tools, complementary to merely evaluating downstream performance, which is predominant in the literature.

Another benefit is that GFMs that are mixtures of linear GNN experts can be decomposed into components with specific ranges using Eq. 2. As linear GNNs are entirely defined by their operator S, and the range of each linear GNN on a task is entirely defined by its operator and the task graph topology, range provides a geometric interpretation with which to investigate an extended, or adaptive, operator basis; for example, operators can be low- or high-pass, but they can also be short-, medium- or long-*range*.

4 kHOPSIGN: FAILURE OF A FIXED BASIS ON A KNOWN-RANGE TASK

In this section we demonstrate empirically that a fixed operator basis cannot arbitrarily adapt to tasks that require a certain minimum architecture, even when expanding the GraphAny basis.

Synthetic tasks with fixed ranges, such as RingTransfer (Bodnar et al., 2021) and k-Dirac (Bamberger et al., 2025) have been used in prior works to study range, but these are typically in the multi-graph setting. We adapt the core idea—nodes retrieving information at a k-hop distance—to the transductive setting, with kHopSign. Given some graph topology with diameter $> k_{\max}$ (we use a random geometric graph with 1000 nodes and radius 0.1), let each node have a scalar feature x_u drawn from a standard Gaussian and binary labels given by $y_u^{(k)} = \text{sign}\left(\sum_{v \in V} \exp\left(-\frac{(d_G(u,v)-k)^2}{2\sigma_{\text{noise}}^2}\right) x_v\right)$; i.e., a 'soft' version of a fixed k-hop retrieval task, which collapses to $y_u^{(k)} = \text{sign}\left(\sum_{v \in \mathcal{N}_k(u)} x_v\right)$ when $\sigma_{\text{noise}} = 0$. This task has a range of $\approx k$ (exactly k in the hard case). We randomly sample a 50:50 train-test split. We train (and evaluate) GraphAny on Wisconsin and, in Table 1, evaluate zero-shot on 25 other benchmarks, as well as kHopSign for $k = 1$ to 8. We use several different operator bases and evaluate on a single seed. The first two bases additionally use $(I - A), (I - A)^2$ as operators at feature generation but not at prediction time, as in Zhao et al. (2024). All instances employ the hyperparameter grid search from Zhao et al. (2024).

Additional operator bases. We know that GraphAny with standard operators must fail for $k > 2$, so to demonstrate that simply *expanding* a fixed basis does not solve the problem, we include several additional bases: further powers of the adjacency, precise-hop adjacencies A_k (close to oracle operators for given k), and two operator families that attempt to compress varying ranges into a handful of operators; $(A_{3:d^*})_{ij} = 1[3 \leq d(i,j) \leq d^*]$ where d^* is the median geodesic distance between all node pairs, i.e., a 'medium-range' binned hop term that covers up to most node pairs, and an $A_{>d^*}$ term that covers the remaining 'long-range' pairs. To capture notions of short-, medium- and long-range *resistance* distance, we use heat kernel terms where $\tau_{\text{S}} = 1, \tau_{\text{M}} = \bar{d}^2, \tau_{\text{L}} = (2\bar{d})^2$ where $\bar{d}$ is the mean geodesic node pair distance.

Discussion. As we expect standard GraphAny to have $\rho < 2$, it is unsurprising that it collapses to random guessing for $k > 1$. Even the basis including third and fourth adjacency powers collapses beyond $k > 1$, suggesting that pure smoothing is insufficient for such a task, even if its range is within the upper bound of the basis. As expected, precise-hop operators A_k yield stronger performance for tasks with corresponding ranges, but this also collapses to random guessing beyond the

Table 1: Performance of GraphAny with various operator bases on the kHopSign task, as well as averaged across 26 benchmark datasets. See Section 5 for implementation details of GOBLIN. For clarity, cells with higher accuracy are shaded in increasing intensities ($\geq 65, 75, 85\%$).

Operators $\mathcal{S} \in \{I, A, \dots\}$	kHopSign (%)									26 graphs Avg. (%)
	1	2	3	4	5	6	7	8	Avg.	
A^2	91	64	49	59	58	54	62	58	62.0	**67.4**
A^2, A^3, A^4	88	60	50	62	59	52	64	58	61.8	*66.1*
A_2, A_3, A_4	79	81	79	76	57	55	57	60	67.9	65.7
$A_2, A_{3:d^*}, A_{>d^*}$	72	87	55	65	70	57	60	65	66.3	65.4
$\{\exp(-\tau L)\}_\tau^{\text{S,M,L}}$	89	68	60	64	68	72	63	62	*68.2*	65.2
GOBLIN	88	87	68	83	89	93	65	89	**82.8**	—[4]

basis (i.e. $k > 4$). Our 'binned' multi-range bases have more success, but inconsistently, and they rarely surpass 70% accuracy. Furthermore, we can see that when we expand the basis, while performance on kHopSign slightly improves, performance *drops* on the benchmarks. This demonstrates that *there is no one-size fits all architecture*/basis; widening the basis to address one architectural requirement introduces noise elsewhere. We require an adaptive basis—and indeed, we see that GOBLIN achieves strong performance across arbitrary ranges. See Sections 5 & 6 for details.

5 Method: Graph Operator Basis Learning & Inference

We introduce GOBLIN, a framework for inference-time architecture adaptation in GFMs. GOBLIN decomposes zero-shot inference into two stages: (i) discovery of a task-adapted basis of linear graph operators from a parameterized operator family, and (ii) permutation-invariant mixture-of-experts weighting over the resulting operators at the node level. Crucially, both stages operate entirely at inference time on the target graph, without retraining or backpropagation.

5.1 Parameterized Operator Families

GOBLIN assumes access to a parameterized family of linear graph operators $S(\theta) \in \mathbb{R}^{N \times N}$, where the parameters θ control architectural properties such as receptive field, smoothing, and aggregation scale. In this work, we consider a family composed of two complementary operator classes:

$$S(\theta) = \lambda\, S^{\text{Gauss}}(\mu, \sigma) + (1 - \lambda)\, S^{\text{Heat}}(\tau), \quad \text{where} \tag{3}$$

$$S^{\text{Gauss}}_{uv}(\mu, \sigma) = \exp\left(-\frac{(\mu - d_G(u, v))^2}{2\sigma^2}\right) \quad \text{and} \quad S^{\text{Heat}}(\tau) = \exp(-\tau L_{\text{sym}}), \tag{4}$$

where $\theta = (\lambda, \mu, \sigma, \tau)$, $\lambda \in \{0, 1\}$, and $\mu, \sigma, \tau \in \mathbb{R}_+$. This approach bears similarity to existing, non-GFM works that use adaptive operators over graph scales, such as SIGN (Frasca et al., 2020) or spectral graph wavelets (Hammond et al., 2011; Opolka et al., 2022). S^{Gauss} corresponds to (soft) distance-localized aggregation around a target hop distance μ, collapsing to exact A_k when $\sigma = 0$, and S^{Heat} corresponds to diffusion-based smoothing controlled by τ, capturing multi-layer message passing with increasing effective range. Such an operator family allows us to span a broad spectrum of architectural behaviors, including low-, high-, and band-pass filtering, short- and long-range aggregation, and local, multi-hop (Gutteridge et al., 2023) and multi-layer message passing, with two modes, each of which encode range via a particular distance metric: geodesic and resistance.

While we focus on this family for concreteness, GOBLIN is agnostic to the specific choice of operator family, and applies to any parameterization that encodes architectural variation.

5.2 Inference-Time Operator Basis Learning

Given an operator family $S(\theta)$, the goal of the first stage is to identify a small set of operator parameters $\{\theta_i\}_{i=1}^t$ that are well-suited to the task on the target graph. For each candidate θ, we

[4] GOBLIN is trained on Cora, so we don't compare against GraphAny trained on Wisconsin.

define a linear GNN expert $\hat{Y}^{\theta} = S(\theta)XW^{\theta}, \quad W^{\theta} = (S(\theta)X)_L^{+}Y_L$, obtained via an analytic least-squares solution on a subset of labeled nodes, as in Zhao et al. (2024).

To evaluate operator suitability without overfitting, we partition the labeled nodes as $Y_L = Y_{L_{\text{fit}}} \cup Y_{L_{\text{eval}}}$, and define a scalar score $f(\theta) = \ell\left(\hat{Y}^{\theta}_{L_{\text{eval}}}, Y_{L_{\text{eval}}}\right)$, where $\hat{Y}^{\theta}_{L_{\text{eval}}}$ is a linear GNN fit to $Y_{L_{\text{fit}}}$ and evaluated on $Y_{L_{\text{eval}}}$, and $\ell(\cdot)$ denotes some score metric. The function $f(\theta)$ is treated as a black-box objective over operator parameters. In practice, $f(\theta)$ may be optimized or approximated using any suitable strategy for sample-efficient black-box optimization. In our experiments, we employ Bayesian optimization (BO), but this choice is not essential to the framework. After a fixed budget of evaluations mixing exploration and exploitation, we select a small set of operators $\{\theta_i\}_{i=1}^{t}$ forming an adaptive basis, and refit each linear GNN on the full labeled set Y_L.

5.3 Permutation-Invariant Mixture of Experts

The second stage combines the adaptive operator basis via a per-node mixture-of-experts. Unlike GraphAny, where the operator basis is fixed and ordered, GOBLIN must remain invariant to both operator ordering and basis size. We therefore employ a Deep Set architecture (Zaheer et al., 2017) over operator-wise features. Like GraphAny, we employ linear-GNN-derived distance features; as we require t set elements rather than a $t(t-1)$ feature vector, we define per-node-pair 'disagreement' as $D_u^{(i,j)} = \|\hat{Y}_u^{(i)} - \hat{Y}_u^{(j)}\|_2^2$, and generate features that are summary statistics of this: $H_u^{(i)} = \left[\frac{1}{t-1}\sum_{j\neq i} D_u^{(i,j)}, \quad \mathrm{Var}_{j\neq i}\left[D_u^{(i,j)}\right], \quad \min_{j\neq i} D_u^{(i,j)}, \quad \max_{j\neq i} D_u^{(i,j)}\right]$. These features capture agreement and diversity among experts without dependence on basis size or ordering.

Note that these are simple, sensible preliminary features to illustrate the method; further experimentation would likely suggest richer expanded features. Each $H_u^{(i)}$ is embedded via a learned function (in practice, an MLP) ϕ, producing operator-wise embeddings $E_u^{(i)} = \phi(H_u^{(i)})$. A second learned function ψ maps the concatenation of $E_u^{(i)}$ and the pooled embedding $\sum_j E_u^{(j)}$ to a scalar logit $\tilde{\alpha}_u^{(i)}$, yielding normalized weights $\alpha_u^{(i)} = \exp(\tilde{\alpha}_u^{(i)}) / \sum_{j=1}^{t} \exp(\tilde{\alpha}_u^{(j)})$ for a final summation over operators. To avoid information leakage during training of ϕ, ψ, features are computed from experts fitted on $Y_{L_{\text{fit}}}$, while supervision is provided by $Y_{L_{\text{eval}}}$. At inference, experts are refit on the full labeled set Y_L before feature extraction and prediction.

6 Experiments

In this section we evaluate, zero-shot, on real-world benchmarks, comparing GOBLIN against models trained end-to-end per task, as well as against GraphAny and TS-GNNs.

Our BO objective is a trimmed mean: accuracy on $Y_{L_{\text{eval}}}$ without the top and bottom 20% of nodes by margin order. This objective ensures a degree of operator robustness and helps identify operators that perform well on a *subset* of nodes, as well as overall. For simplicity we fix $\sigma = 0.5$ and use 9 exploration steps randomly split between the two 1D parameter spaces, μ and τ; candidate samples are chosen based on maximum separation in parameter space. Then we have six exploitation steps, sampling each Gaussian process posterior argmax and ensuring a minimum separation between samples in parameter space. Two objective-maximizing sampled operators are then chosen as a basis; we also include A^2 as a sensible fixed baseline operator, and to diversify the feature space. For fair comparison with other models, we train on Cora, the same training graph used by Finkelshtein et al. (2025), and evaluate zero-shot, following the experimental procedure from that work.

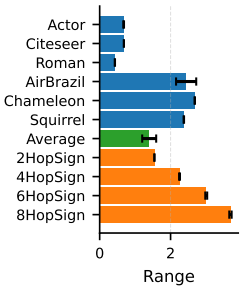

Figure 1: Range of GOBLIN on several benchmarks, $\pm\sigma$ over seeds. 'Average' is over benchmarks in Table 2.

We evaluate on 25 standard benchmark tasks (Table 2), as well as 4 real-world long-range tasks (Table 3) and our synthetic kHopSign tasks (Table 1). The long-range tasks are from CityNetworks (Liang et al., 2025), a benchmark that requires

Table 2: Mean $\pm$ std % accuracy (4 seeds) across 25 benchmark graph datasets. End-to-end models are trained directly on each task, so are easier than the zero-shot setting. Non-GOBLIN columns are reproduced from Finkelshtein et al. (2025). Heterophilic graphs are marked with an asterisk. First-/second-best results per dataset are color-coded.

	End-to-end		Zero-shot		
	MeanGNN	GAT	GraphAny	TS-Mean	GOBLIN
Actor*	32.03 $\pm$ 0.29	32.59 $\pm$ 0.83	27.54 $\pm$ 0.20	28.09 $\pm$ 0.93	27.48 $\pm$ 0.10
AirBrazil	32.31 $\pm$ 7.50	35.38 $\pm$ 4.21	33.84 $\pm$ 15.65	39.23 $\pm$ 5.70	50.00 $\pm$ 11.32
AirEU	39.12 $\pm$ 6.44	39.00 $\pm$ 4.30	41.25 $\pm$ 7.25	35.88 $\pm$ 6.91	39.84 $\pm$ 2.81
AirUS	43.93 $\pm$ 1.16	43.03 $\pm$ 2.08	43.35 $\pm$ 1.62	42.34 $\pm$ 2.12	49.19 $\pm$ 2.12
AmzComp	73.88 $\pm$ 0.88	70.94 $\pm$ 3.40	82.79 $\pm$ 1.13	81.37 $\pm$ 1.25	82.18 $\pm$ 1.57
AmzPhoto	88.95 $\pm$ 1.08	80.78 $\pm$ 3.59	89.91 $\pm$ 0.88	90.18 $\pm$ 1.30	90.06 $\pm$ 1.07
AmzRatings*	40.74 $\pm$ 0.13	40.63 $\pm$ 0.66	42.80 $\pm$ 0.09	42.27 $\pm$ 1.40	43.90 $\pm$ 0.03
BlogCatalog	84.48 $\pm$ 0.74	78.20 $\pm$ 7.23	71.54 $\pm$ 3.04	76.30 $\pm$ 2.92	77.55 $\pm$ 3.47
Chameleon*	61.75 $\pm$ 0.94	58.46 $\pm$ 6.23	63.64 $\pm$ 1.48	60.83 $\pm$ 5.41	73.03 $\pm$ 0.00
Citeseer	65.06 $\pm$ 1.30	63.92 $\pm$ 0.84	68.88 $\pm$ 0.10	68.66 $\pm$ 0.19	67.75 $\pm$ 0.06
CoCS	80.87 $\pm$ 0.69	82.28 $\pm$ 0.86	90.06 $\pm$ 0.80	90.92 $\pm$ 0.47	91.13 $\pm$ 0.45
CoPhysics	79.05 $\pm$ 1.13	85.92 $\pm$ 1.10	91.85 $\pm$ 0.34	92.61 $\pm$ 0.61	89.74 $\pm$ 2.05
Cornell*	63.78 $\pm$ 1.48	69.73 $\pm$ 2.26	63.24 $\pm$ 1.32	68.65 $\pm$ 2.42	62.16 $\pm$ 0.00
DBLP	65.01 $\pm$ 2.40	67.34 $\pm$ 2.75	71.48 $\pm$ 1.44	66.42 $\pm$ 3.65	70.81 $\pm$ 1.67
FCora	55.85 $\pm$ 1.04	59.95 $\pm$ 0.88	51.18 $\pm$ 0.78	53.58 $\pm$ 0.73	56.61 $\pm$ 0.09
Minesweeper*	84.06 $\pm$ 0.18	84.15 $\pm$ 0.24	80.46 $\pm$ 0.11	80.68 $\pm$ 0.38	80.98 $\pm$ 0.02
Pubmed	74.56 $\pm$ 0.13	75.12 $\pm$ 0.89	76.46 $\pm$ 0.08	74.98 $\pm$ 0.56	76.97 $\pm$ 0.05
Questions*	97.16 $\pm$ 0.06	97.13 $\pm$ 0.05	97.07 $\pm$ 0.03	97.02 $\pm$ 0.01	95.09 $\pm$ 0.15
Roman*	69.37 $\pm$ 0.66	69.80 $\pm$ 4.18	63.34 $\pm$ 0.58	66.36 $\pm$ 1.02	63.05 $\pm$ 0.03
Squirrel*	43.32 $\pm$ 0.66	38.16 $\pm$ 1.04	49.74 $\pm$ 0.47	41.81 $\pm$ 0.80	64.10 $\pm$ 0.14
Texas*	76.76 $\pm$ 1.48	81.62 $\pm$ 6.45	71.35 $\pm$ 2.16	73.51 $\pm$ 4.01	77.70 $\pm$ 1.35
Tolokers*	78.59 $\pm$ 0.66	78.22 $\pm$ 0.37	78.20 $\pm$ 0.03	78.12 $\pm$ 0.09	78.24 $\pm$ 0.02
Wiki	74.23 $\pm$ 0.89	68.91 $\pm$ 9.50	60.27 $\pm$ 3.06	69.89 $\pm$ 1.31	58.79 $\pm$ 2.17
Wisconsin*	74.12 $\pm$ 12.20	73.33 $\pm$ 8.27	59.61 $\pm$ 5.77	61.18 $\pm$ 11.38	65.20 $\pm$ 1.88
WkCS	71.97 $\pm$ 1.70	74.99 $\pm$ 0.59	74.11 $\pm$ 0.60	74.16 $\pm$ 2.07	70.81 $\pm$ 0.23
Average	66.04 $\pm$ 1.83	65.98 $\pm$ 2.91	65.76 $\pm$ 1.96	66.20 $\pm$ 2.31	**68.09 $\pm$ 1.31**

the classification of road junction accessibility from large-diameter graphs ($\sim$1-600k nodes), requiring information up to 16 hops away.

Discussion. Table 2 shows that, while other methods are, on average, within 0.5% of each other GOBLIN is about 2% better than the next best model, and beats standard MPNNs (Velickovic, 2017) trained end-to-end on the tasks. We note that GOBLIN's overall higher performance is dominated by a few datasets (AirBrazil $>$10%, Chamelon $>$10%, Squirrel $>$20%); indeed, for some tasks, GOBLIN under-performs compared to baselines. This is expected performance for a more general model: we want robustness across tasks, not SOTA, and these results suggest that existing GFMs are not necessarily robust to these tasks, which may be longer-range than expected. Figure 1, in which we report the ranges of GOBLIN on selected benchmarks, validates this idea. As expected, the kHopSign tasks generally increase in range with increasing k. On average, the 25 standard benchmark tasks have low range, around that of a 2-hop synthetic task. Our three outlier tasks, however, are all much higher range than the average; we can conclude that GOBLIN's architectural adaptivity compared to fixed-short-range baselines is the main contributor to the performance boost.[5]

Long-range. In Table 3 we see that GOBLIN out-performs baseline GFMs, a result we expect given their strictly low ranges. Indeed, even GOBLIN's operators saturate at the maximum available

[5]We note that strong performance on these three particular tasks is *not* a simple consequence of heterophily; indeed, AirBrazil is homophilic. Contrast this with Roman, a large-diameter heterophilic graph that nevertheless has very low range—a finding previously demonstrated by Bamberger et al. (2025) and discussed by Arnaiz-Rodriguez & Errica (2025).

Table 3: Mean $\pm$ std % accuracy (4 seeds) on CityNetworks. We report TS-GAT rather than TS-Mean as it performed better on CityNetworks.

	MeanGNN	GAT	GraphAny	TS-GAT	GOBLIN
Paris	20.79 ± 0.25	20.73 ± 0.28	20.09 ± 0.10	16.36 ± 1.93	24.30 ± 0.00
Shanghai	22.86 ± 0.13	20.80 ± 0.18	19.72 ± 0.02	19.93 ± 1.62	22.34 ± 0.00
LA	19.34 ± 0.08	18.74 ± 0.25	17.19 ± 0.02	15.85 ± 1.42	21.13 ± 0.00
London	21.58 ± 0.12	21.13 ± 0.15	17.13 ± 0.01	OOM[6]	21.51 ± 0.00
Average	21.14 ± 0.15	20.35 ± 0.22	18.53 ± 0.04	17.38 ± 1.66	22.32 ± 0.00

range (this is what causes zero σ; see Appendix D.4.2). GOBLIN out-performs end-to-end trained MPNNs overall, but we note that (for fair cross-benchmark comparison) these use the same parameter search described above, with a fixed depth of 2 layers. In practice, CityNetworks is extremely depth-sensitive, and all of these results can be significantly improved upon by increasing model depth. We discuss this in detail in AppendixD.4.

7 Conclusion

This work aims to highlight that a prerequisite for an effective GFM (fixed architecture *and* model weights) is a versatile *architecture* and, for GNNs, the problem of a foundation architecture is arguably not yet solved. Range is one axis along which architecture is significant; we highlight the weakness of established GFMs in generalization along this axis due to their static architectures. We encourage the community to include synthetic and real-world long-range tasks in the evaluation of GFMs in the future.

Limitations. GOBLIN is not a SOTA method; as a mixture-of-experts architecture composed of weak linear GNN learners, it is insufficiently expressive for many complex tasks (see Table 3; Appendix D.4.2). Our aim is to highlight that in GFM development and evaluation, architectural trade-offs should be explicitly considered, and pure-performance homogeneous-task benchmarking is insufficient.

GOBLIN incurs higher inference-time cost than fixed-basis methods like GraphAny. In particular, S^{Gauss} operators require computation of shortest-path distances up to an effective radius of approximately $\mu + 3\sigma$, which can be expensive on large graphs, although these distances can be computed once-per-graph and cached. S^{Heat} operators are also more costly than simple adjacency powers, though they can be efficiently approximated using truncated Taylor expansions. Overall, while GOBLIN typically requires more linear GNN solves at inference time (e.g., ~ 10 versus 5 in GraphAny[7]), we view this as a reasonable trade-off for improved robustness.

Future work. Our current instantiation of GOBLIN uses simple, heuristically chosen sampling rules for operator selection and a small set of summary features for mixture-of-experts weighting. These choices are not fundamental to the framework. More expressive operator features would likely improve performance, and adaptive sampling strategies could balance search budget against task difficulty. An investigation into training across *multiple graphs* would likely lead to improvements in both efficiency and generalization.

Acknowledgments

B.G. is supported by the EPSRC Centre for Doctoral Training in AIMS (EP/S024050/1). M.B. is supported by the EPSRC Turing AI World-Leading Research Fellowship No. EP/X040062/1 and EPSRC AI Hub No. EP/Y028872/1.

[6] TS-GNNs suffered GPU out-of-memory errors on London due to its size; average is over remaining tasks.
[7] 15 samples, each solving on $N_L/2$ labels; computational complexity roughly halved if $N_L \gg d$.

REFERENCES

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020.

Adrian Arnaiz-Rodriguez and Federico Errica. Oversmoothing, oversquashing, heterophily, long-range, and more: Demystifying common beliefs in graph machine learning. *arXiv preprint arXiv:2505.15547*, 2025.

Álvaro Arroyo, Alessio Gravina, Benjamin Gutteridge, Federico Barbero, Claudio Gallicchio, Xiaowen Dong, Michael Bronstein, and Pierre Vandergheynst. On vanishing gradients, over-smoothing, and over-squashing in gnns: Bridging recurrent and graph learning. *arXiv preprint arXiv:2502.10818*, 2025.

Hugo Attali, Nadi Tomeh, Thomas Papastergiou, and Fragkiskos D Malliaros. When do distant dependencies matter? diagnostics for long-range propagation in GNNs, 2026. URL `https://openreview.net/forum?id=5RbpF0U3aQ`.

Jacob Bamberger, Benjamin Gutteridge, Scott le Roux, Michael M Bronstein, and Xiaowen Dong. On measuring long-range interactions in graph neural networks. *arXiv preprint arXiv:2506.05971*, 2025.

Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Lio, Guido F Montufar, and Michael Bronstein. Weisfeiler and lehman go cellular: Cw networks. *Advances in neural information processing systems*, 34:2625–2640, 2021.

Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*, 2017.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Paper review:'sparks of artificial general intelligence: Early experiments with gpt-4'. 2023.

Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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.

Yukang Dong, Fanxing Pan, Yi Gui, Wenbin Jiang, Yao Wan, Ran Zheng, and Hai Jin. Comprehensive architecture search for deep graph neural networks. *IEEE Transactions on Big Data*, 2025.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

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

Dmitry Eremeev, Oleg Platonov, Gleb Bazhenov, Artem Babenko, and Liudmila Prokhorenkova. Graphpfn: A prior-data fitted graph foundation model. *arXiv preprint arXiv:2509.21489*, 2025.

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

Ben Finkelshtein, Ismail Ilkan Ceylan, Michael Bronstein, and Ron Levie. Equivariance everywhere all at once: A recipe for graph foundation models. *arXiv preprint arXiv:2506.14291*, 2025.

Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. Sign: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198*, 2020.

Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. Towards foundation models for knowledge graph reasoning. *arXiv preprint arXiv:2310.04562*, 2023.

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

David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and computational harmonic analysis*, 30(2):129–150, 2011.

Adrian Hayler, Xingyue Huang, Ismail Ilkan Ceylan, Michael Bronstein, and Ben Finkelshtein. Bringing graphs to the table: Zero-shot node classification via tabular foundation models. *arXiv preprint arXiv:2509.07143*, 2025.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

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.

Kexin Huang, Payal Chandak, Qianwen Wang, Shreyas Havaldar, Akhil Vaid, Jure Leskovec, Girish N Nadkarni, Benjamin S Glicksberg, Nils Gehlenborg, and Marinka Zitnik. A foundation model for clinician-centered drug repurposing. *Nature Medicine*, 30(12):3601–3613, 2024.

Huidong Liang, Haitz Sáez de Ocáriz Borde, Baskaran Sripathmanathan, Michael Bronstein, and Xiaowen Dong. Towards quantifying long-range interactions in graph machine learning: a large graph dataset and a measurement. *arXiv preprint arXiv:2503.09008*, 2025.

Jiawei Liu, Cheng Yang, Zhiyuan Lu, Junze Chen, Yibo Li, Mengmei Zhang, Ting Bai, Yuan Fang, Lichao Sun, Philip S Yu, et al. Graph foundation models: Concepts, opportunities and challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

Péter Mernyei and Cătălina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*, 2020.

Luca Miglior, Matteo Tolloso, Alessio Gravina, and Davide Bacciu. Can you hear me now? a benchmark for long-range graph propagation. *arXiv preprint arXiv:2512.17762*, 2025.

Matheus Nunes and Gisele L Pappa. Neural architecture search in graph neural networks. In *Brazilian conference on intelligent systems*, pp. 302–317. Springer, 2020.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947*, 2019.

Felix Opolka, Yin-Cong Zhi, Pietro Lio, and Xiaowen Dong. Adaptive gaussian processes on graphs via spectral graph wavelets. In *International Conference on Artificial Intelligence and Statistics*, pp. 4818–4834. PMLR, 2022.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Mehran Kazemi, Rami Al-Rfou, and Jonathan Halcrow. Let your graph do the talking: Encoding structured data for llms. *arXiv preprint arXiv:2402.05862*, 2024.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of gnns under heterophily: Are we really making progress? *arXiv preprint arXiv:2302.11640*, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 385–394, 2017.

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

Nimrod Segol and Yaron Lipman. On universal equivariant set networks. *arXiv preprint arXiv:1910.02421*, 2019.

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

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 491–500, 2024.

Jan Tönshoff, Martin Ritzert, Eran Rosenbluth, and Martin Grohe. Where did the gap go? reassessing the long-range graph benchmark. *arXiv preprint arXiv:2309.00367*, 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

P Velickovic. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Ding Wang, Xuhong Wang, Liang Chen, Shengyue Yao, Ming Jing, Honghai Li, Shiqiang Bao, Li Li, Fei-Yue Wang, and Yilun Lin. Transworldng: Traffic simulation via foundation model. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 6007–6012. IEEE, 2023a.

Haishuai Wang, Yang Gao, Xin Zheng, Peng Zhang, Hongyang Chen, Jiajun Bu, and Philip S Yu. Graph neural architecture search with gpt-4. *arXiv preprint arXiv:2310.01436*, 2023b.

Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, et al. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. *Advances in Neural Information Processing Systems*, 36:61501–61513, 2023c.

Yuxiang Wang, Wenqi Fan, Suhang Wang, and Yao Ma. Towards graph foundation models: A transferability perspective. *arXiv preprint arXiv:2503.09363*, 2025.

Bin Wu, Yihang Wang, Yuanhao Zeng, Jiawei Liu, Jiashu Zhao, Cheng Yang, Yawen Li, Long Xia, Dawei Yin, and Chuan Shi. Graph foundation models for recommendation: A comprehensive survey. *arXiv preprint arXiv:2502.08346*, 2025.

Lianghao Xia and Chao Huang. Anygraph: Graph foundation model in the wild. *arXiv preprint arXiv:2408.10700*, 2024.

Renchi Yang, Jieming Shi, Xiaokui Xiao, Yin Yang, Sourav S Bhowmick, and Juncheng Liu. Pane: scalable and effective attributed network embedding. *The VLDB Journal*, 32(6):1237–1262, 2023.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48. PMLR, 2016.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Jianan Zhao, Hesham Mostafa, Michael Galkin, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. Graphany: A foundation model for node classification on any graph. *arXiv preprint arXiv:2405.20445*, 29, 2024.

Dongzhuoran Zhou, Evgeny Kharlamov, and Egor V Kostylev. Glora: A benchmark to evaluate the ability to learn long-range dependencies in graphs. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yun Zhu, Yaoke Wang, Haizhou Shi, and Siliang Tang. Efficient tuning and inference for large language models on textual graphs. *arXiv preprint arXiv:2401.15569*, 2024.

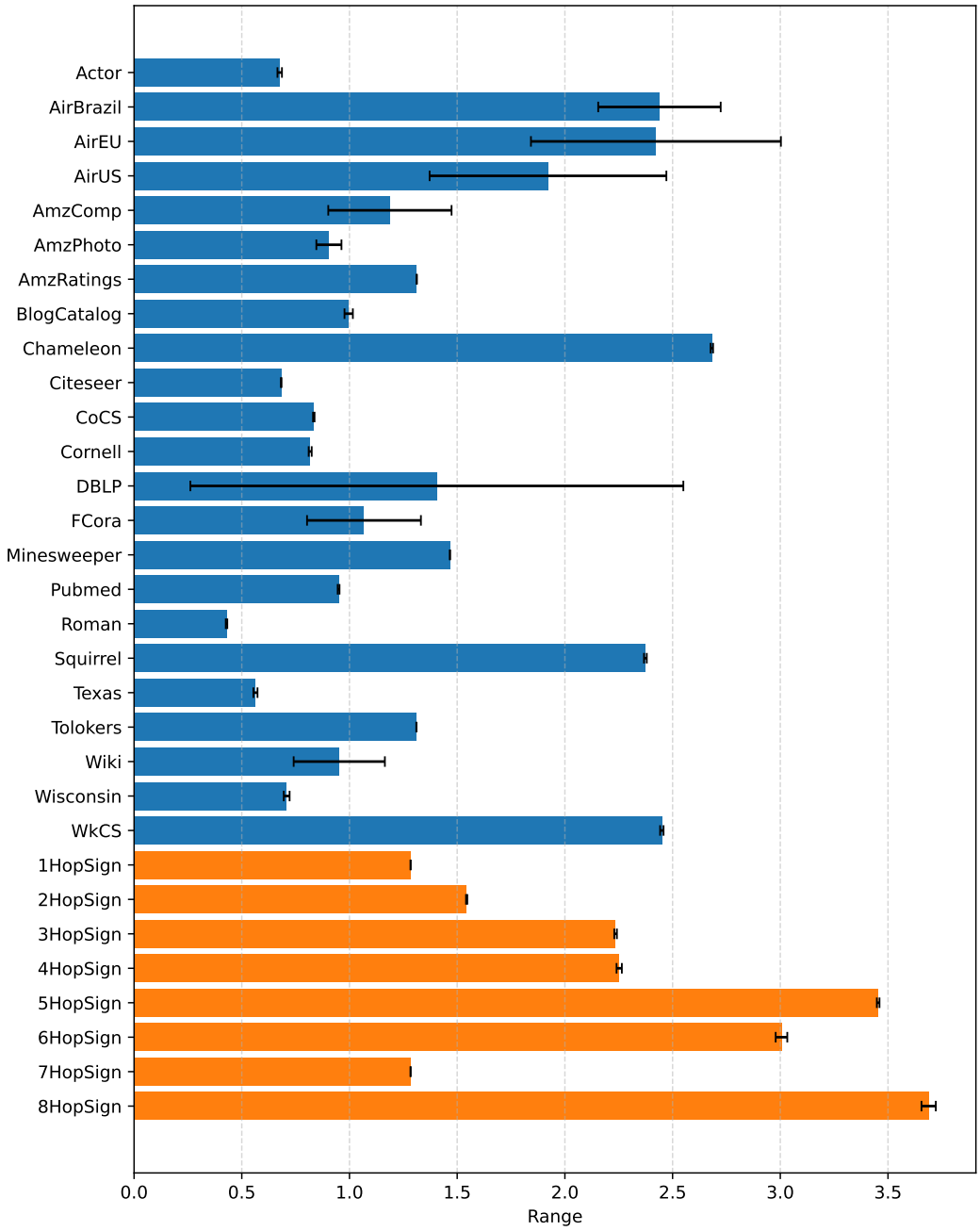

Figure 2: Range of GOBLIN on synthetic and real-world benchmark tasks ($\pm\sigma$ over 4 seeds).

A RANGE OF GOBLIN ON BENCHMARKS

Figure 2 shows overall GOBLIN model ranges for each real-world and synthetic benchmark.

Given node level weights $\boldsymbol{\alpha} \in \mathbb{R}^{N \times t}$ we compute per-expert average weights $\bar{\alpha} = \frac{1}{N}\mathbf{1}^\top \boldsymbol{\alpha} \in \mathbb{R}^t$.

Similarly, we obtain graph-level per-expert ranges by averaging over node-level ranges $\boldsymbol{\rho}_u \in \mathbb{R}^t$: $\boldsymbol{\rho}_G = \frac{1}{N}\sum_{u \in V} \boldsymbol{\rho}_u$, where each element ρ_u is as defined in Equation 2.

Figure 2 plots $\bar{\alpha}^\top \boldsymbol{\rho}_G$, averaged over seeds, $\pm$ standard deviation.

As expected, the range of GOBLIN on kHopSign tasks generally increases with k, and varies quite a lot with benchmarks, depending on the task. It is worth explaining why the range of the HopSign tasks is not approximately k: since GOBLIN uses 3 different operators with an enforced minimum separation in parameter space, and kHopSign is optimally solved by a S^{Gauss} operator with $\mu = k$, 2 operators will have non-$\sim k$ range. As these terms are often noisy, with small weight norms, the most correct expert can still dominate the overall prediction without the weights of other terms going to zero. As a result, the average GOBLIN range is not necessarily $\approx k$, but range and k do correlate strongly.

It is interesting to note that some of the tasks with the largest ranges, around 2-2.5, such as Chameleon, Squirrel and AirBrazil, are also the tasks with the most significant performance boost over existing methods; as much as 10 or 20% accuracy (see Table 2). This suggests that GOBLIN is indeed learning a higher-range adaptive architecture to improve performance on tasks that are more challenging to standard MPNN-based GFMs with low ranges.

B RANGE OF GOBLIN OPERATOR FAMILIES

Gaussian. By definition, the range of S^{Gauss} operators is centered on μ. Depending on topology and σ, the range may be higher or lower than μ: $\rho > \mu$ when $|\mathcal{N}_{k+\delta}(u)| > |\mathcal{N}_{k-\delta}(u)|$, and vice versa; exactly μ for, for example, regular lattices.

Heat. The heat kernel operator $S^{\text{Heat}}(\tau)$spreads information from a node via a diffusion process rather than targeting a specific distance. Its effective range corresponds to the typical displacement of a random walk after time τ. For small τ, aggregation remains localized; as τ increases, the range grows gradually due to the diffusive nature of random walks, and eventually saturates once the walk mixes across the graph. As a result, the heat operator provides smooth, multi-scale aggregation but does not offer direct control over a precise interaction range. To put it another way, it represents a range-driven operator using random-walk/resistance distance as the distance metric on the graph, rather than geodesic distance.

C MEASURING RANGE FOR LINEAR GNN MIXTURE-OF-EXPERTS MODELS

C.1 WHY WE USE FIXED-W RANGE

A linear GNN expert has the form

$$\hat{Y} = SXW, \qquad W = (SX)_L^+ Y_L,$$

so its Jacobian decomposes as

$$\frac{\partial \hat{Y}}{\partial X} = \frac{\partial (SX)}{\partial X} W \;+\; (SX)\,\frac{\partial W}{\partial X}.$$

The first term captures *explicit architectural mixing*: how information is propagated across the graph by the operator S. The second term arises solely from refitting W to the labeled nodes and reflects sensitivity of the *regression solution*, not additional message passing.

Crucially, this second term does not introduce new graph-dependent paths by which information can propagate between nodes. Any dependence of $\hat{Y}_u$ on X_v induced by $\partial W / \partial X$ must already be mediated through SX, and therefore cannot extend the receptive field beyond that encoded by S. As a result, it cannot increase the effective range of the model.

From an architectural perspective, S fully specifies the computation graph through which node features interact; W merely selects a linear readout within that fixed interaction structure. Since our goal is to diagnose *architectural under-reaching*, isolating the contribution of S is sufficient and appropriate.

This mirrors standard practice in range and receptive-field analyses of trained GNNs, where weights are treated as fixed and range is understood as a property of the message-passing structure (Bamberger et al., 2025).

We also find this black-box range to be uninformative in practice; below we compute both types of range for GraphAny's linear GNN operators on several benchmark tasks and detail the relative weights for checkpointed GraphAny instances.

C.2 Fixed-W and black box ranges for linear GNNs for benchmark graphs

This section contains tables for 15 datasets; all *available, connected datasets* used by Zhao et al. (2024), except for the three largest ones, for which it is computationally infeasible to obtain the all-pairs geodesic distance. Let $\rho_{\bar{W}}$ denote fixed-weight linear GNN range as it is derived in the paper, and ρ_{B} denote black-box range computed with `torch.autograd.functional.jacobian` for the end-to-end linear GNN solve.

Discussion. $\rho_{\bar{W}}$ is around 0.6–1.0 for all datasets, usually about 0.9. The split between linear GNNs is relatively even everywhere; almost every weight is close to 0.33, and none are below 0.22 or above 0.48.

For most graphs variation in ρ_{B} between different linear GNNs (except for X, which is typically much lower) is minimal; some amount of mixing appears to be significant, but there is little difference between 1- and 2-hop. The second power is usually a little higher though. Overall ρ_{B} values do not vary much between *training* datasets on the same *evaluation* dataset, possibly because all of the training datasets are essentially short-range. There is some variation across evaluation datasets in ρ_{B}. In general we see that the range skews fairly close to the `mean_dist`, i.e. the average SPD between node pairs. Given that the linear GNN features are analytic solutions that use the subset of labeled nodes, it is likely that the solutions depend heavily on labeled nodes. If we assume that some labeled node w is approximately as close to some node u as any other non-labeled node v, it makes sense that the range would track this value. Roman is particularly unusual — because it is close to a chain graph, with low degree and a very large diameter, its range ends up being extremely high. Minesweeper is similar.

D Experimental details

D.1 Datasets

We use 29 benchmark tasks in this paper: Cornell, Texas, Wisconsin, Chameleon, Squirrel, Actor (Pei et al., 2020), AirBrazil, AirUS, AirEU (Ribeiro et al., 2017), Wiki, BlogCatalog (Yang et al., 2023), Cora, Citeseer, Pubmed Yang et al. (2016), AmzPhoto, AmzComp, CoCS, CoPhysics (Shchur et al., 2018), WikiCS (Mernyei & Cangea, 2020), FullCora, DBLP Bojchevski & Günnemann (2017), Roman, AmzRatings, Minesweeper, Tolokers and Questions (Platonov et al., 2023), and finally the four CityNetworks tasks: London, Paris, LA and Shanghai Liang et al. (2025).

With the exception of CityNetworks, we draw from the same set of benchmarks as Finkelshtein et al. (2025), but drop LastFMAsia and Deezer (Rozemberczki et al., 2021) as they can no longer be downloaded from Pytorch Geometric (Fey & Lenssen, 2019), as well as Arxiv (Hu et al., 2020), as its size makes it impractical to work with. Full dataset statistics are in Table 25.

D.2 GOBLIN hyperparameters

We perform a random hyperparameter search for GOBLIN, about 400 samples, each taking approximately 9 minutes on an A10 GPU for training and evaluation on val split. The search space and final hyperparameters are given in Table 7.

Table 4: Overall ranges (and for reference, validation accuracies), both fixed-W and black box, for 17 connected benchmark datasets. Computation of ρ_{B} uses 500 sampled nodes, or all of them, whichever is higher. In most cases, ρ_B tracks mean geodesic distance fairly closely; $\rho_{\bar{W}}$ provides a more accurate version of range—due to the fixed $\rho \leq 2$ operators used, effective range of GraphAny on all tasks is < 1. All ranges are computed on the checkpointed instantiation of GraphAny from Zhao et al. (2024), trained on Wisconsin.

Dataset	$\rho_{\bar{W}}$	ρ_{B}	Mean geodesic distance	Val. Acc. (%)
AirBrazil	0.80	1.63	2.17	50.00
Cornell	0.80	3.22	3.18	71.19
Texas	0.80	2.79	3.02	74.14
AirEU	0.82	1.29	2.32	34.59
Squirrel	0.91	2.34	3.09	42.31
Wisconsin	0.80	3.18	3.24	62.50
BlogCatalog	0.88	1.84	2.51	76.04
Chameleon	0.89	2.42	3.56	62.14
Actor	0.92	2.35	4.11	30.26
Minesweeper	0.81	29.87	46.66	80.16
Tolokers	0.93	2.12	2.79	78.26
CoCS	0.82	3.65	5.43	91.09
Pubmed	0.89	5.78	6.34	77.60
Roman	0.71	2152.7	2331.5	63.57
AmzRatings	0.80	16.04	16.24	42.05
CoPhysics	0.82	2.84	5.16	92.38
Questions	0.95	3.62	4.29	97.08

D.3 Gaussian process discussion

Although the evaluation metric is discrete, the objective varies smoothly with operator parameters in practice, since predictions depend continuously on the underlying graph operators and on the corresponding least-squares solutions. Small perturbations of operator parameters typically induce only small changes in node logits, and hence only limited changes in aggregate accuracy, with abrupt changes arising only when a small number of nodes lie close to decision boundaries. To account for this behavior, we model the objective using a Gaussian process with an RBF kernel, encoding the assumption that operators with similar parameters tend to exhibit correlated performance. We include an additive white-noise term to capture residual non-smoothness and evaluation noise arising from finite label sets and discrete predictions. In our implementation we fix the kernel length scale to 1 and the noise variance to $\sigma_n = 0.2$, which we found to provide a stable inductive bias without overfitting. We further reduce sensitivity to unstable decision boundaries by using a trimmed-mean accuracy over nodes as the optimization objective, which empirically improves robustness to small, localized prediction sensitivity.

$\ell(\cdot)$ **objective and mode-mixing options.** `mean`: simple accuracy on $Y_{L_{\text{eval}}}$; `trimmed_10/20`: per-node accuracy on $Y_{L_{\text{eval}}}$, sorted by margin, with the top and bottom 10/20% removed; `lower_half/quartile`: margin-sorted node predictions with the most confidently correct half or quarter removed; `mean_minus_var_0.2/0.5`: $\mu - \lambda\sigma$ objective with $\lambda \in \{0.2, 0.5\}$; `balanced/biased`: sampling in (μ, τ) space with either 50:50 or 25:75 probability mass; `adaptive`: on-the-fly bias toward the mode achieving the highest score so far.

D.4 CityNetworks

See Table 3 for results.

Discussion. GraphAny and TS-GNNs both struggle here due to the long-range nature of the task and their fixed short-range architectures. While performance increases could be obtained by re-training the GFM with more layers, this (i) defeats the purpose of a GFM and (ii) would increase

computational complexity (and likely decrease performance accuracy on short-range tasks). Using our GOBLIN parameterization we obtain results at least 2-3% better than other GFMs, and better than or on-par with GNNs trained explicitly on each dataset.

D.4.1 DEPTH.

As shown in Liang et al. (2025), deeper models perform much better at the CityNetworks tasks. It is likely that increasing depth would improve performance for TS-GNN, and we know it increases performance significantly for non-zero-shot MPNN baselines. In Table 5 we confirm this explicitly; using identical hyperparameters to the MPNN results in Table 3, we only change the depth from $L = 2$ to $L = 16$, and observe a huge performance improvement, especially for GAT.

While GOBLIN's maximum range in our particular hyperparameterization is only 8, in comparison to an 'oracle' MPNN with 16 layers, it is likely that even an expanded GOBLIN would struggle on this task. Simply put, it is too complex for weak linear GNN experts. Rather than undermining our method though, we believe this finding highlights our central argument: we are not trying to argue that GOBLIN is SOTA—indeed, other works exist which already improve on many of the benchmark results in this paper (Hayler et al., 2025). CityNetworks is highly sensitive to a particular architecture/hyperparameter; TS-GNNs and GraphAny fail to generalize to it, so in a real setting we would be forced to do an expensive, time consuming architecture search, training a new MPNN from scratch each time, to learn that $L = 16$ is optimal. GOBLIN cannot currently achieve performance on par with the best results reported by Liang et al. (2025) due to its *own* architectural limitation: expressiveness. But future GFM research can be more mindful of this tradeoff. More expressive mixture-of-experts models can hypothetically generalize to different task types and architectural requirements without needing a one-size-fits-all. We leave such investigations to future work.

Table 5: CityNetworks accuracy (%) for deep ($L = 16$) MeanGNN and GAT trained directly on each dataset using the hyperparameter search described in Finkelshtein et al. (2025). Results are mean $\pm$ std over 4 training seeds with fixed data splits.

Dataset	MeanGNN	GAT
Paris	27.10 ± 0.67	46.75 ± 0.50
Shanghai	28.64 ± 0.36	58.50 ± 0.75
LA	26.89 ± 0.15	53.36 ± 0.59
London	28.13 ± 0.25	47.72 ± 0.25

D.4.2 ZERO STANDARD DEVIATION RESULTS ON GOBLIN.

In Table 3, GOBLIN has zero standard deviation for all tasks. This is because, on all tasks, GOBLIN's operator bases saturate at the maximum values for μ (8) and τ (5) allotted to the hyperparameter search space during BO (see Table 7). This is expected if we consider CityNetwork benchmarking results from Liang et al. (2025), in which deeper GNNs (i.e. GNNs capable of longer ranges) perform better than shallow ones, with 16-layer models generally performing the best. Indeed, these tasks are designed to have a range of 16 hops, so it is unsurprising that GOBLIN's parameters saturate in this direction. We would likely expect improved performance for an even more flexible range of operators, but we maintain the same hyperparameter limits across all experiments for consistent comparison.

D.4.3 CITYNETWORKS ON TS-GNNS

The codebase accompanying Finkelshtein et al. (2025) does not include model weights, so we retrained the models with hyperparameters(/search) as detailed. We approximately recover the results given in the paper; see Table 6 for examples.

Table 6: Results reported in Finkelshtein et al. (2025) for several benchmarks, and the performance we were able to reproduce. Reproduced scores are similar.

Dataset	TS-Mean	Paper	TS-GAT	Paper
Squirrel	41.5 ± 1.4	41.8 ± 0.8	38.5 ± 4.1	38.3 ± 4.3
Wisconsin	62.0 ± 10.8	61.2 ± 11.4	65.5 ± 9.9	65.9 ± 9.9
Texas	73.0 ± 4.3	73.5 ± 4.0	74.1 ± 4.1	74.1 ± 4.1
AirBrazil	38.5 ± 4.7	39.2 ± 5.7	36.9 ± 9.7	36.9 ± 10.0
AirUSA	42.2 ± 1.9	42.3 ± 2.1	41.8 ± 2.6	41.7 ± 2.6
AirEU	36.3 ± 7.2	35.9 ± 6.9	38.7 ± 6.2	38.4 ± 6.3
Cornell	68.1 ± 1.2	68.7 ± 2.4	69.7 ± 5.5	71.4 ± 4.5

Table 7: Hyperparameter search space for GOBLIN experiments in Tables 1 & 2.

Hyperparameter	Search space	Chosen
$\ell(\cdot)$ objective	`mean, trimmed_20, trimmed_10, lower_half, lower_quartile, mean_minus_var_0.2, mean_minus_var_0.5`	`trimmed_20`
Operator mode mixing	`balanced, adaptive, biased_heat, biased_gauss`	`balanced`
Basis size	2,3,4	2
Enforce Gauss & Heat	True, False	False
#Explore steps	6, 9, 12, 16	9
#Exploit steps	2, 3, 4, 6	6
τ min, max	0, 5.0	—
τ min. separation search	0.1	—
τ min. separation basis	0.2, 0.5, 1.0	1.0
μ min, max	0, 8.0	—
μ min. separation search	0.2	—
μ min. separation basis	0.5, 1.0	0.5
#Fixed operators	0, 1, 2, 3, 4, 5	1
Fixed operator pool (sampled randomly)	$I, A, A^2, (I-A), (I-A)^2$	A^2
RBF kernel lengthscale	1.0	—
GP kernel white noise	0.2	—
LR	3e-4, 3e-3	3e-4
Hidden dim.	16, 32, 64	32
Attention temp.	1, 3, 5, 10	10
#Deep Set layers	1, 2, 3	2
#Head layers	1, 2	1
#Epochs	200, 500, 1000, 1500	500

Table 8: Ranges and weights of GraphAny linear GNN components for the **AirBrazil** dataset.

LinearGNN	$\rho_{\bar{W}}$	ρ_{B} (131)	α_{Cora}	$\alpha_{\text{Wisconsin}}$	α_{Arxiv}	α_{Product}
X	0.00	1.36	0.30	0.37	0.43	0.37
AX	1.00	1.75	0.35	0.35	0.34	0.35
A^2X	1.61	1.84	0.35	0.28	0.23	0.28
$(I-A)X$	0.50	1.84	—	—	—	—
$(I-A)^2X$	0.87	1.77	—	—	—	—

Table 9: Ranges and weights of GraphAny linear GNN components for the **Cornell** dataset.

LinearGNN	$\rho_{\bar{W}}$	ρ_{B} (183)	α_{Cora}	$\alpha_{\text{Wisconsin}}$	α_{Arxiv}	α_{Product}
X	0.00	2.98	0.30	0.34	0.35	0.37
AX	1.00	3.42	0.35	0.38	0.36	0.34
A^2X	1.48	3.23	0.35	0.29	0.30	0.29
$(I-A)X$	0.50	3.09	—	—	—	—
$(I-A)^2X$	0.86	3.18	—	—	—	—

Table 10: Ranges and weights of GraphAny linear components for the **Texas** dataset.

LinearGNN	$\rho_{\bar{W}}$	ρ_{B} (183)	α_{Cora}	$\alpha_{\text{Wisconsin}}$	α_{Arxiv}	α_{Product}
X	0.00	2.82	0.30	0.32	0.33	0.34
AX	1.00	2.11	0.35	0.42	0.36	0.34
A^2X	1.50	3.89	0.35	0.25	0.31	0.31
$(I-A)X$	0.50	2.94	—	—	—	—
$(I-A)^2X$	0.87	3.02	—	—	—	—

Table 11: Ranges and weights of GraphAny linear GNN components for the **Wisconsin** dataset.

LinearGNN	$\rho_{\bar{W}}$	ρ_{B} (251)	α_{Cora}	$\alpha_{\text{Wisconsin}}$	α_{Arxiv}	α_{Product}
X	0.00	2.94	0.30	0.34	0.34	0.36
AX	1.00	3.28	0.35	0.38	0.36	0.34
A^2X	1.48	3.32	0.35	0.29	0.30	0.30
$(I-A)X$	0.50	3.06	—	—	—	—
$(I-A)^2X$	0.86	3.11	—	—	—	—

Table 12: Ranges and weights of GraphAny linear components for the **AirEU** dataset.

LinearGNN	$\rho_{\bar{W}}$	ρ_{B} (399)	α_{Cora}	$\alpha_{\text{Wisconsin}}$	α_{Arxiv}	α_{Product}
X	0.00	0.36	0.29	0.40	0.42	0.48
AX	1.00	1.68	0.35	0.29	0.33	0.30
A^2X	1.71	2.11	0.36	0.31	0.25	0.22
$(I-A)X$	0.50	1.90	—	—	—	—
$(I-A)^2X$	0.91	2.29	—	—	—	—

Table 13: Ranges and weights of GraphAny linear components for the **Squirrel** dataset.

LinearGNN	$\rho_{\bar{W}}$	ρ_{B} (500)	α_{Cora}	$\alpha_{\text{Wisconsin}}$	α_{Arxiv}	α_{Product}
X	0.00	1.44	0.30	0.35	0.34	0.38
AX	1.00	2.64	0.35	0.31	0.33	0.31
A^2X	1.76	2.99	0.35	0.34	0.33	0.31
$(I-A)X$	0.50	2.38	—	—	—	—
$(I-A)^2X$	0.93	2.59	—	—	—	—

Table 14: Ranges and weights of GraphAny linear components for the **BlogCatalog** dataset.

LinearGNN	$\rho_{\bar{W}}$	ρ_{B} (500)	α_{Cora}	$\alpha_{\text{Wisconsin}}$	α_{Arxiv}	α_{Product}
X	0.00	0.92	0.30	0.38	0.37	0.41
AX	1.00	2.38	0.35	0.33	0.34	0.33
A^2X	1.89	2.42	0.35	0.29	0.29	0.26
$(I-A)X$	0.50	2.37	—	—	—	—
$(I-A)^2X$	0.97	2.37	—	—	—	—

Table 15: Ranges and weights of GraphAny linear components for the **Chameleon** dataset.

LinearGNN	$\rho_{\bar{W}}$	ρ_{B} (500)	α_{Cora}	$\alpha_{\text{Wisconsin}}$	α_{Arxiv}	α_{Product}
X	0.00	1.13	0.30	0.31	0.30	0.32
AX	1.00	2.79	0.35	0.40	0.35	0.34
A^2X	1.70	3.33	0.35	0.29	0.35	0.34
$(I-A)X$	0.50	2.47	—	—	—	—
$(I-A)^2X$	0.90	2.74	—	—	—	—

Table 16: Ranges and weights of GraphAny linear components for the **Actor** dataset.

LinearGNN	$\rho_{\bar{W}}$	ρ_{B} (500)	α_{Cora}	$\alpha_{\text{Wisconsin}}$	α_{Arxiv}	α_{Product}
X	0.00	1.00	0.30	0.33	0.35	0.38
AX	1.00	2.39	0.35	0.30	0.33	0.31
A^2X	1.69	3.55	0.35	0.36	0.33	0.30
$(I-A)X$	0.50	2.46	—	—	—	—
$(I-A)^2X$	0.92	3.11	—	—	—	—

Table 17: Ranges and weights of GraphAny linear components for the **Minesweeper** dataset.

LinearGNN	$\rho_{\bar{W}}$	ρ_{B} (500)	α_{Cora}	$\alpha_{\text{Wisconsin}}$	α_{Arxiv}	α_{Product}
X	0.00	26.63	0.31	0.32	0.32	0.35
AX	1.00	30.65	0.36	0.34	0.34	0.33
A^2X	1.37	32.10	0.34	0.34	0.35	0.32
$(I-A)X$	0.50	34.24	—	—	—	—
$(I-A)^2X$	0.81	30.59	—	—	—	—

Table 18: Ranges and weights of GraphAny linear components for the **Tolokers** dataset.

LinearGNN	$\rho_{\bar{W}}$	ρ_{B} (500)	α_{Cora}	$\alpha_{\text{Wisconsin}}$	α_{Arxiv}	α_{Product}
X	0.00	2.02	0.30	0.33	0.32	0.35
AX	1.00	1.98	0.35	0.36	0.35	0.34
A^2X	1.82	2.39	0.34	0.32	0.32	0.30
$(I-A)X$	0.50	2.05	—	—	—	—
$(I-A)^2X$	0.95	2.05	—	—	—	—

Table 19: Ranges and weights of GraphAny linear components for the **CoCS** dataset.

LinearGNN	$\rho_{\bar{W}}$	ρ_{B} (500)	α_{Cora}	$\alpha_{\text{Wisconsin}}$	α_{Arxiv}	α_{Product}
X	0.00	2.21	0.30	0.34	0.31	0.34
AX	1.00	4.14	0.35	0.37	0.36	0.35
A^2X	1.54	4.73	0.34	0.29	0.33	0.31
$(I-A)X$	0.50	4.02	—	—	—	—
$(I-A)^2X$	0.87	4.41	—	—	—	—

Table 20: Ranges and weights of GraphAny linear components for the **Pubmed** dataset.

LinearGNN	$\rho_{\bar{W}}$	ρ_{B} (500)	α_{Cora}	$\alpha_{\text{Wisconsin}}$	α_{Arxiv}	α_{Product}
X	0.00	6.02	0.30	0.34	0.35	0.39
AX	1.00	5.63	0.35	0.34	0.35	0.33
A^2X	1.71	5.69	0.34	0.32	0.30	0.28
$(I-A)X$	0.50	5.83	—	—	—	—
$(I-A)^2X$	0.93	5.73	—	—	—	—

Table 21: Ranges and weights of GraphAny linear components for the **Roman** dataset.

LinearGNN	$\rho_{\bar{W}}$	ρ_{B} (500)	α_{Cora}	$\alpha_{\text{Wisconsin}}$	α_{Arxiv}	α_{Product}
X	0.00	2088.70	0.30	0.33	0.35	0.39
AX	1.00	2212.81	0.35	0.26	0.29	0.28
A^2X	1.12	2166.40	0.35	0.40	0.36	0.34
$(I-A)X$	0.50	2173.21	—	—	—	—
$(I-A)^2X$	0.75	2179.69	—	—	—	—

Table 22: Ranges and weights of GraphAny linear components for the **AmzRatings** dataset.

LinearGNN	$\rho_{\bar{W}}$	ρ_{B} (500)	α_{Cora}	$\alpha_{\text{Wisconsin}}$	α_{Arxiv}	α_{Product}
X	0.00	16.10	0.31	0.32	0.31	0.35
AX	1.00	15.98	0.35	0.35	0.35	0.34
A^2X	1.39	16.06	0.34	0.33	0.34	0.32
$(I-A)X$	0.50	15.99	—	—	—	—
$(I-A)^2X$	0.80	15.81	—	—	—	—

Table 23: Ranges and weights of GraphAny linear components for the **CoPhysics** dataset.

LinearGNN	$\rho_{\bar{W}}$	ρ_{B} (500)	α_{Cora}	$\alpha_{\text{Wisconsin}}$	α_{Arxiv}	α_{Product}
X	0.00	0.83	0.30	0.33	0.29	0.31
AX	1.00	3.36	0.35	0.39	0.38	0.37
A^2X	1.57	4.54	0.34	0.27	0.34	0.32
$(I-A)X$	0.50	3.05	—	—	—	—
$(I-A)^2X$	0.88	4.10	—	—	—	—

Table 24: Ranges and weights of GraphAny linear components for the **Questions** dataset.

LinearGNN	$\rho_{\bar{W}}$	ρ_{B} (200)	α_{Cora}	$\alpha_{\text{Wisconsin}}$	α_{Arxiv}	α_{Product}
X	0.00	3.34	0.30	0.32	0.32	0.35
AX	1.00	3.46	0.35	0.35	0.35	0.34
A^2X	1.82	4.07	0.34	0.33	0.33	0.31
$(I-A)X$	0.50	3.84	—	—	—	—
$(I-A)^2X$	0.96	3.82	—	—	—	—

Table 25: Statistics for the 26 datasets used in this paper. APSPD is all-pairs shortest path (geodesic) distance.

Dataset	#Nodes	#Edges	#Feature	#Classes	#Labeled Nodes	Diam.	Mean APSPD	Train/Val/Test Ratios (%)	Hetero-philic	Source
AirBrazil	131	1074	131	4	80	4	2.17	61.1/19.1/19.8		Ribeiro et al. (2017)
Cornell	183	554	1703	5	87	8	3.18	47.5/32.2/20.2	✓	Pei et al. (2020)
Texas	183	558	1703	5	87	8	3.02	47.5/31.7/20.2	✓	Pei et al. (2020)
Wisconsin	251	900	1703	5	120	8	3.25	47.8/31.9/20.3	✓	Pei et al. (2020)
AirEU	399	5995	399	4	80	5	2.32	20.1/39.8/40.1		Ribeiro et al. (2017)
AirUS	1190	13599	1190	4	80	—	—	6.7/46.6/46.6		Ribeiro et al. (2017)
Chameleon	2277	36101	2325	5	1092	11	3.56	48.0/32.0/20.0	✓	Pei et al. (2020)
Wiki	2405	17981	4973	17	340	—	—	14.1/42.9/43.0		Yang et al. (2023)
Cora	2708	10556	1433	7	140	—	—	5.2/18.5/36.9		Yang et al. (2016)
Citeseer	3327	9104	3703	6	120	—	—	3.6/15.0/30.1		Yang et al. (2016)
BlogCatalog	5196	343486	8189	6	120	4	2.51	2.3/48.8/48.8		Yang et al. (2023)
Squirrel	5201	217073	2089	5	2496	10	3.09	48.0/32.0/20.0	✓	Pei et al. (2020)
Actor	7600	30019	932	5	3648	12	4.11	48.0/32.0/20.0	✓	Pei et al. (2020)
AmzPhoto	7650	238162	745	8	160	—	—	2.1/49.0/49.0		Shchur et al. (2018)
Minesweeper	10000	78804	7	2	5000	99	46.66	50.0/25.0/25.0	✓	Platonov et al. (2023)
WikiCS	11701	431206	300	10	580	—	—	5.0/15.1/49.9		Mernyei & Cangea (2020)
Tolokers	11758	1038000	10	2	5879	11	2.79	50.0/25.0/25.0	✓	Platonov et al. (2023)
AmzComp	13752	491722	767	10	200	—	—	1.5/49.3/49.3		Shchur et al. (2018)
DBLP	17716	105734	1639	4	80	—	—	0.5/49.8/49.8		Bojchevski & Günnemann (2017)
CoCS	18333	163788	6805	15	300	24	5.43	1.6/49.2/49.2		Shchur et al. (2018)
Pubmed	19717	88648	500	3	60	18	6.34	0.3/25.5/51.5		Yang et al. (2016)
FullCora	19793	126842	8710	70	1400	—	—	7.1/46.5/46.5		Bojchevski & Günnemann (2017)
Roman Empire	22662	65854	300	18	11331	6824	2331.5	50.0/25.0/25.0	✓	Platonov et al. (2023)
AmzRatings	24492	186100	300	5	12246	46	16.24	50.0/25.0/25.0	✓	Platonov et al. (2023)
CoPhysics	34493	495924	8415	5	100	17	5.16	0.3/49.9/49.9		Shchur et al. (2018)
Questions	48921	307080	301	2	24460	16	4.29	50.0/25.0/25.0	✓	Platonov et al. (2023)