Fixed Aggregation Features Can Rival GNNs

Celia Rubio-Madrigal [1] **Rebekka Burkholz** [1]

Abstract

Graph neural networks (GNNs) are widely believed to excel at node representation learning through trainable neighborhood aggregations. We challenge this view by introducing Fixed Aggregation Features (FAFs), a training-free approach that transforms graph learning tasks into tabular problems. This simple shift enables the use of well-established tabular methods, offering strong interpretability and the flexibility to deploy diverse classifiers. Across 14 benchmarks, well-tuned multilayer perceptrons trained on FAFs rival or outperform state-of-the-art GNNs and graph transformers on 12 tasks—often using only mean aggregation. The only exceptions are the Roman Empire and Minesweeper datasets, which typically require unusually deep GNNs. To explain the theoretical possibility of non-trainable aggregations, we connect our findings to Kolmogorov–Arnold representations and discuss when mean aggregation can be sufficient. In conclusion, our results call for (i) richer benchmarks benefiting from learning diverse neighborhood aggregations, (ii) strong tabular baselines as standard, and (iii) employing and advancing tabular models for graph data to gain new insights into related tasks.

1. Introduction

Graph neural networks (GNNs) have become the standard approach for learning from graph based data, and in particular, for solving node classification. Most models follow the message-passing paradigm (Gilmer et al., 2017), where each node updates its representation by alternating neighborhood aggregation with learned linear combinations across multiple hops. This framework has been remarkably successful at combining node features with graph structure, driving applications in domains ranging from social networks to biology (Bongini et al., 2023; Sharma et al., 2024). Yet, it comes at the cost of high model complexity that poses challenges for interpretation. We ask the question whether this high complexity is really necessary.

Recent evidence (Luo et al., 2024) shows that classic models, such as GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), and GraphSAGE (Hamilton et al., 2017), surprisingly remain competitive when equipped with proper hyperparameter tuning and optimization techniques. When carefully tuned, they can rival more sophisticated approaches, including state-of-the-art Graph Transformers (Wu et al., 2023; Deng et al., 2024b; Kong et al., 2023; Wu et al., 2022; Chen et al., 2023; Rampášek et al., 2022; Shirzad et al., 2023) and models designed for heterophily (Zhu et al., 2020; 2021; Chien et al., 2021; Maurya et al., 2022; Li et al., 2022).

These results invite a closer look at which components of graph architectures are essential for strong performance, and thus raise a natural next question: *How relevant is learning the aggregation?* In fact, the field has invested in learning increasingly complex convolution layers and attention mechanisms. In this paper we challenge that premise from first principles. Leveraging the Kolmogorov–Arnold representation theorem (Kolmogorov, 1957; Schmidt-Hieber, 2021), we give an explicit, lossless construction of neighborhood aggregations. Consequently, one can in theory encode neighbor features without discarding information. However, the same construction exposes a crucial gap between expressiveness and learnability: These lossless encoders are numerically brittle (e.g., sensitive to floating-point noise) and tend to produce "rough" embeddings that are ill-suited for standard classifiers on Euclidean space such as MLPs.

Surprisingly, we find that standard, untrained aggregation operators—like sum, mean, max, min, and std—though not information-preserving, yield useful features without any learnable parameters. Building on this observation, we propose Fixed Aggregation Features (FAF) (§ 3): a training-free aggregation pipeline that applies fixed aggregation functions—also referred to as "reducers"—over neighborhoods at multiple hops, concatenates the results into a tabular feature matrix, and then trains only a downstream classifier (e.g., an MLP). This data transformation brings several

[1]CISPA Helmholtz Center for Information Security. Correspondence to: Celia Rubio-Madrigal <celia.rubio-madrigal@cispa.de>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

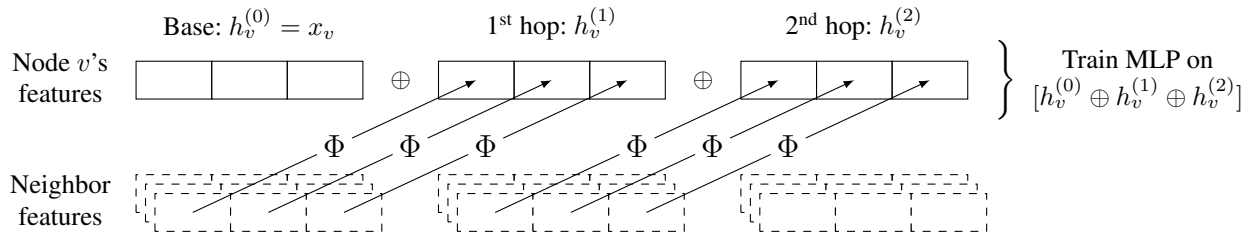


Figure 1. Fixed Aggregation Features (FAFs) are calculated as a pre-processing step, concatenated to the input ($\oplus$), and fed to an MLP. If the aggregation function Φ is injective, the neighborhood information is preserved. The Kolmogorov-Arnold representation theorem ensures the existence of such a function, although simple reducers are empirically more amenable for optimization.

advantages: high interpretability (feature importance and ablations over hops/reducers), compatibility with the rich toolbox of tabular learning (designed to handle noise, class imbalance, feature selection, etc.), architectural flexibility, and reduced training compute.

Empirically, FAFs combined with well-tuned MLP classifiers are competitive on 12/14 common node-classification benchmarks, including citation (McCallum et al., 2000; Sen et al., 2008; Namata et al., 2012), coauthor and Amazon co-purchase graphs (Shchur et al., 2018), Wikipedia (Mernyei & Cangea, 2020; Rozemberczki et al., 2021), and other heterophilous datasets (Platonov et al., 2023b). Performance only really trails on Minesweeper and Roman-Empire, where the best GNNs rely on linear residual connections; in fact, the remaining gap aligns with the gains from the parameterized residuals reported by Luo et al. (2024). This pattern suggests that these datasets (Minesweeper and Roman-Empire) benefit from hop-specific aggregations or combinations of consecutive hops. Also, while these GNNs profit from many layers (10–15), the best-performing FAFs use only 2–4 hops. On the other hand, why do FAFs perform on par for the other datasets? Our theoretical analysis of the employed aggregation functions (§ 4) and our empirical findings (§ 5) suggest that, for most benchmarks, the relevant signal is concentrated within hops 0–2; on hops 0–1, sum and mean preserve information. At higher hops, different reducers are complementary, but the information gain from min/max diminishes.

FAFs let us examine datasets from an optimization-first viewpoint without hard-to-interpret architectural factors (§ 3.1). By turning neighborhoods into tabular features, we decouple representation from optimization and enable standard interpretability tools (e.g., feature importance) to identify which hop distances and reducers carry signal. Beyond revisiting the homophily–heterophily dichotomy or one-hop informativeness (Platonov et al., 2023a; Zheng et al., 2024; 2025), our method supports a richer characterization of interaction patterns: how signal varies across scales, which effects are additive vs. redundant, and where long-range dependencies appear. Moreover, gains reported by new methods could be dissected by constructing and comparing to their FAF counterparts. The tabular view also makes it natural to augment features, e.g., with network-science descriptors, or with neighborhood-masking features inspired by rewiring (Rubio-Madrigal et al., 2025; Roth et al., 2025). Furthermore, concurrent work has begun to explore a tabularization perspective to adapt tabular foundation models to graph data (Eremeev et al., 2025; Choi et al., 2025; Hayler et al., 2025), a promising direction we help contextualize.

Together, our results suggest that many benchmarks do not require sophisticated learned aggregations, and that a large portion of GNN performance can be matched by powerful tabular predictors fed with fixed, transparent features. Thus, FAFs can serve both as a *strong baseline* and as a *diagnostic tool* for graph benchmarks and methods. In summary, here are our main contributions:

1. Theory: We construct lossless neighborhood aggregations via Kolmogorov–Arnold representations, clarifying that learnability and numeric stability, not just expressiveness, govern practical success. We also analyze what information common reducers extract from neighborhoods, revealing information preservation at 0 and 1 hops and diminishing information with higher depth.

2. Method: We introduce FAFs, which convert graph data into a tabular task by stacking fixed multi-hop aggregations, offering an interpretable framework to study the interplay between graph structure, features, and the task.

3. Empirical evidence: FAFs match or exceed classic GNNs on 12/14 standard benchmarks. Our experiments corroborate our analysis, finding low-hop features to be more important and diminishing information at higher depth.

4. Implications: Our findings question the necessity of learned neighborhood aggregation on current standard benchmarks, motivate strong tabular baselines for graph data, and open a path to more interpretable, efficient graph learning—and to designing harder benchmarks that genuinely benefit from learning aggregation.

2. Related Work

Simplifying GNNs. A growing body of work shows that much of a GNN's power can be retained—even improved for some tasks—when message passing is simplified or fixed. Early evidence comes from Kipf & Welling (2017), inspiring lines of work where aggregation layers are frozen or randomized: Ramachandra et al. (2025) are competitive on relational graphs with randomly transformed random features; Kelesis et al. (2025) analyze partially frozen GCNs, showing that fixing aggregation can mitigate oversmoothing and ease optimization; and GESN (Gallicchio & Micheli, 2010) compute embeddings via a dynamical system with randomly initialized reservoir weights, after which only a linear readout is trained (Micheli & Tortorella, 2024).

Another simplification comes from SGC (Wu et al., 2019), which remove nonlinearities from a GCN, yielding a model equivalent to applying a low-pass graph filter followed by a linear readout. For link prediction (Qarkaxhija et al., 2024), SGCNs are found to be better than GCNs, but even removing the linear classifier—and thus all trainable parameters—provides a good baseline. Subsequent work has explored this idea further by precomputing propagation before training. SIGN (Frasca et al., 2020) precomputes and concatenates features obtained from several fixed linear diffusion operators, then applies learned transformations to the representations—while discussing the need for exploring more expressive or wider local operators; GAMLP (Zhang et al., 2022) and HOGA (Deng et al., 2024a) learn complex attention or gating mechanisms on them to decide which hops matter. These methods demonstrate separating propagation from prediction, though only as a way to improve scalability. Moreover, our FAFs are not limited to linear diffusions: reducers such as max, min, and std are nonlinear multiset operations, and several datasets require such reducers to reach their best performance. Fixed aggregation is also used in APPNP (Gasteiger et al., 2019): An MLP is first trained to produce node embeddings, and a subsequent Personalized PageRank–based propagation is applied; though the propagation itself is fixed, gradients flow through during backpropagation. And for graph classification on non-attributed graphs, Cai & Wang (2019) show that first-neighborhood statistics with an SVM form a surprisingly strong baseline, but performance lags on attributed graphs.

In contrast, we focus on node classification with rich node features. We aggregate and concatenate across all hops, and we place a more carefully tuned graph-agnostic classifier on top, all of which are necessary for our results—as shown in Tables 7, 10, and 11 (Appendix C, D). We thus highlight the value of concatenating dependent but informative hop-wise features (Reddy et al., 2026), and the benefits of overparameterization—echoing evidence from random-feature models (Donghi et al., 2024).

Benchmarking GNNs. Our work also connects to the growing literature on properly benchmarking GNNs and what constitutes a meaningful graph-learning dataset. Recent analyses warn that graph learning risks losing relevance without application-grounded benchmarks (Bechler-Speicher et al., 2025) and principled criteria for dataset quality beyond accuracy (Coupette et al., 2025). For graph classification, Errica et al. (2020) show that, under controlled protocols, simple and even structure-agnostic baselines can rival complex GNNs, suggesting that architectures often fail to exploit graph structure. On the dataset side, Bazhenov et al. (2025) recently introduced industrial node property prediction benchmarks with graph-agnostic baselines. Their neighborhood feature aggregation (NFA), which augments tabular models with one-hop aggregated neighbor statistics, can be seen as a one-hop instance of our FAFs. We include preliminary results for four of their datasets in Appendix F. In this context, FAFs serve as a simple stress test of whether proposed benchmarks genuinely benefit from learned message passing, and we argue that such well-tuned, fixed, multi-hop baselines should be routinely included when assessing new graph models and datasets.

Tabularizing graphs for TFMs. Concurrent work has explored converting node classification into a tabular prediction problem in order to leverage recent tabular foundation models (TFMs). Eremeev et al. (2025) propose G2T-FM, which augments node features with one-hop NFA and structural embeddings before applying a tabular foundation model. Choi et al. (2025) similarly introduce TabPFN-GN, which constructs tabular features from node attributes, structural embeddings, and smoothed fixed aggregation. Finally, Hayler et al. (2025) uses ensembles of subsampled tables constructed from feature encodings, structural encodings, and labels. These works are similar in spirit to FAFs in that they expose graph information through tabular features, but differ in scope: they focus on adapting tabular foundation models for graph prediction, whereas our goal is to study how far simple fixed multi-hop aggregations with a carefully tuned graph-agnostic classifier can go, and what this implies about the need to learn neighborhood aggregation.

GNN aggregation functions. Classical message-passing GNNs differ primarily in how they aggregate neighbor features under permutation invariance. Sum, mean, and max are the canonical choices, with injectivity and stability trade-offs of each of them tied to their multiset representations (Xu et al., 2019). Beyond single operators, principal neighborhood aggregation (PNA) mixes several base reducers with degree-aware scalers to boost expressiveness and well-conditioning for continuous reducers (Corso et al., 2020). Attention mechanisms instantiate learned weighted sums (Veličković et al., 2018; Brody et al., 2022), although it has been shown that they suffer from trainability problems, including small relative gradients on the attention param-

eters, slowed-down layer-wise training speed, and the inability to mute neighbors (Mustafa et al., 2023; Mustafa & Burkholz, 2024a;b). Our perspective is complementary: We study fixed reducers whose strength comes from (i) their information preservation and (ii) their separability by a powerful downstream classifier. This decoupling clarifies what must be learned (the readout) versus what can be fixed (the propagation), and it aligns with our empirical finding that stronger, well-tuned classifiers capitalize on rich, concatenated neighborhood views. Because of this, we argue that good optimization and learnability are as important as expressivity results. In line with this argument, Gorishniy et al. (2022) show that, for tabular data, having the right embeddings for continuous features is key to closing the gap between transformer-like architectures and feed forward networks, proposing a lossless piecewise linear embedding to improve the trainability of the latter.

Kolmogorov-Arnold theorem. The Kolmogorov–Arnold representation (Kolmogorov, 1957) admits several equivalent formulations that reduce multivariate functions to compositions of univariate functions. Recent architectures explicitly instantiate such decompositions with learnable spline-based univariate components and linear mixing (Liu et al., 2025; Carlo et al., 2025). In contrast, we use a specific fixed-aggregation formulation with predetermined aggregation weights and a fixed univariate encoding (Schmidt-Hieber, 2021), so that any multivariate function f can be learned exclusively from a single univariate readout g applied to a fixed weighted sum of univariate transforms.

3. Fixed Aggregation Features

Node neighborhoods can be compressed into single-node features, eliminating the need to learn feature embeddings before every layer of message passing. Our approach, Fixed Aggregation Features (FAFs), recursively constructs and concatenate features via a set of reducers $\mathcal{R} \subset \{\text{mean}, \text{sum}, \text{max}, \text{min}, \text{std}, \ldots\}$ in the following way:

$$h_v^{(0,r)} = x_v, \quad h_v^{(k,r)} = r\Big(\{\, h_u^{(k-1,r)} : u \in N(v) \,\}\Big), \quad (1)$$

where $k \in \{1, \ldots, K\}$ and $r \in \mathcal{R}$. We then train an MLP on the concatenated representation

$$z_v = x_v \oplus \left(\bigoplus_{r \in \mathcal{R}} \bigoplus_{k \in \{1,\ldots,K\}} \left(h_v^{(k,r)} \right) \right) \quad (2)$$

with input dimensionality $|x_v| \cdot (1 + |\mathcal{R}| \cdot K)$ per node v. Figure 1 illustrates the case for one reducer $\mathcal{R} = \{\Phi\}$ with $K = 2$. If the reducers are injective, then the neighborhood information at each depth is preserved in z_v. This waives the need for aggregating learned embeddings in GNNs, thus transforming graph data into high-dimensional tabular data. Our analysis explains why this is theoretically possible (§ 4). Additionally, in our experiments (§ 5), we show that MLPs trained on FAFs can match the performance of classic GNNs on most standard node-classification benchmarks and, by comparing with Luo et al. (2024), of Graph Transformers and heterophily-aware models.

3.1. Advantages of Tabular over Graph Representations

Interpretability. Our construction concatenates each node's original features with K-hop neighborhood statistics and feeds this expanded representation to a tabular classifier. This setup explicitly separates the feature and hop aggregation factors, allowing us to assess their individual contributions using the widely-used toolbox for tabular interpretability. For instance, we can analyze effects across hops by examining feature importance of the MLP. We compute Shapley Additive Explanations (SHAP) (Lundberg & Lee, 2017) on Minesweeper with mean aggregation (Fig. 2), one of the two datasets where FAFs lag. In this dataset, labels are bombs, feature 0 masks other features, and features 1–6 one-hot encode the number of neighboring bombs. The top signal is the hop-1 mean of feature 1, i.e., the fraction of neighbors whose local bomb count is null. When this proportion is greater than zero, the model knows that the node cannot have a bomb; when it is zero, all its neighbors observe bombs so there is a possibility of having a bomb. This heuristic does not completely solve the problem, as neighbors can be merely observing second-hop bombs, creating ambiguity that likely causes residual errors. The model also correctly gives importance to the number of masked neighbors (hop-0 feature 0). These attributions clarify where the model succeeds and where it fails. We also report SHAP importances for two other datasets in Fig. 6 in Appendix D: Pubmed (homophilic) and Amazon-Ratings (heterophilic).

Moreover, rather than explaining a particular classifier, one can aim to explain the dataset, localizing which hops and features carry signal independent of model choice. Following Donnelly et al. (2023), noisy tabular datasets often admit a "Rashomon set" of comparably well-performing models. Accordingly, feature importance is better assessed over this set—preferably constrained to simpler or sparser

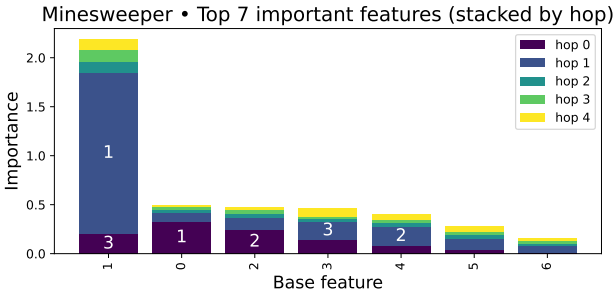


Figure 2. SHAP feature importance for Minesweeper, stacked by hop. Numbers show features' rankings per hop.

models—than from a single fit. This lens offers a principled way to interrogate graph data beyond feature homophily–heterophily (Zheng et al., 2024) or strictly one-hop neighborhoods (Zheng et al., 2025).

Optimization. GNNs usually exhibit early overfitting, where training accuracy converges almost immediately while validation and test accuracy plateau or even decay. Thus, the best validation accuracy is often achieved before relevant aggregations are learned. This might partially explain why FAFs can often compete with trained GNNs: They avoid overfitting aggregations. GNNs can also suffer from ineffective aggregation learning (Mustafa et al., 2023; Mustafa & Burkholz, 2024b), so their potential to outcompete FAFs is likely underexplored due to trainability issues. By contrast, optimization on tabular data (like FAFs) is better tractable and understood by standard toolkits.

Efficiency. Precomputing aggregation and then training an MLP is more scalable than repeatedly running message passing layers and backpropagating through them, as required in GNNs. However, as the number of reducers, original features, and hops in FAF increases, so does the input dimensionality, which in turn enlarges the parameter count of the MLP's first layer. This issue could be mitigated through common feature reduction techniques. For the original features, we report the average training runtimes of our FAF and GNN models in Table 3 (Appendix B). FAFs are generally more efficient, particularly when using a single reducer.

Feature augmentations. Adding more features does not improve accuracy monotonically; beyond a point, some feature selection is needed. Still, the tabular view lets us concatenate diverse features atop the aggregations, which serve as additional informative covariates (Reddy et al., 2026). This can, in principle, include degree, centrality, or other node-level structural encodings (Grötschla et al., 2026). Our framework is also compatible with pre-processing graph rewiring, where aggregation is computed on a modified adjacency matrix to fight over-squashing (Topping et al., 2022; Jamadandi et al., 2024). But unlike standard rewiring, we can concatenate the rewired features instead of replacing the originals. As fixed aggregations can suffer from similar issues as trainable GNNs, FAFs can also benefit from proposed remedies. To extract more precise information from complex environments, we examine a feature similarity-based rewiring based on Rubio-Madrigal et al. (2025), where edges of negative feature cosine similarity between nodes are dropped. We then append features aggregated on the rewired graph, or split edges into positive/negative sets and aggregate separately, inspired by computational-graph splitting that fights over-smoothing (Roth et al., 2025). Results in Table 13 (Appendix D) show that on most datasets, concatenating these extra features yields larger gains than using them as substitution. These augmentations not only improve performance but also help disentangle where the gains come from: additional extracted signal versus changes to the optimization of graph models. This is akin to analyses of SGC and GESN (Micheli & Tortorella, 2025), though in our case we obtain benefits from the operations. We therefore advocate FAF constructions as baselines for methods that modify the aggregation component of GNNs.

4. Does Aggregation Need Learning?

Let $G = (V, E)$ be a graph with node features $\mathcal{X} \in \mathbb{R}^F$. A neighborhood function is a map $f : \mathcal{M}(\mathbb{R}^F) \to \mathbb{R}$, where $\mathcal{M}(\mathbb{R}^F)$ denotes the set of finite multisets of vectors in $\mathbb{R}^F$, acting on the neighborhood multiset $X_v := \{\mathcal{X}_u : u \in N(v)\}$ for $v \in V$. We treat $v \in N(v)$ when self-loops are added, depending on the setup. We seek a fixed encoder Φ such that we can learn any neighborhood function via a univariate map g with $f = g \circ \Phi^{-1}$. This enables tabular classifiers to learn graph data losslessly.

What is preserved by standard aggregations? FAFs apply a transformation of neighborhoods that is not learnable, which raises the question of what information gets lost by the aggregation. Permutation invariant aggregations treat graph neighborhoods as multisets consisting of feature vectors of neighbors. Accordingly, they extract distribution information and forget about the identity of specific neighbors. This property is usually regarded as helpful inductive bias and therefore of no concern. Our next theorems analyze which information is preserved by sum and mean aggregations from these multisets. To do so, we first generalize Lemma 5 by Xu et al. (2019) for one-hot encoded discrete features to orthogonal features. Combined with the fact that hop features are concatenated, this insights establishes that information from the 1-hop neighborhood can be preserved.

Theorem 4.1 (1-hop aggregation)**.** *Assume the features $\mathcal{X}$ are orthogonal. Then, the function $h(X) = \sum_{x \in X} x$ defined on multisets $X \subseteq \mathcal{X}$ of bounded size is injective. Moreover, any multiset function f can be decomposed as $f(X) = g\big(\sum_{x \in X} x\big)$ for some function g.*

The proof is given in Appendix A.2. Note that a multiset $X \subseteq \mathcal{X}$ is characterized by the count n_x of elements that have specific features x. These counts can also be extracted from the sum $h(X)$ (as demonstrated in the proof). Consequently, any multiset function f transforms such counts by $f(n_x)$. The function g would thus first extract the counts from the sum $h(X)$ and then apply f to the counts. If the features of a node v include its degree d_v, then mean aggregation contains the same information, as a classifier can learn to multiply $h(X) = 1/d_v \sum_{x \in X} x$ by d_v. In contrast, max and min aggregations extract whether at least one neighbor has a specific feature property. They focus on the tails of distributions rather than full neighborhoods.

Information loss for k-hops. One might hope that the above theorem also applies to aggregations from hop k to $k+1$. The orthogonality assumption, however, is essential and no longer met by the aggregated neighbor features $h_n^{(k)}$ for $k \geq 1$. As a consequence, from $k \geq 2$, not all information about the distribution of features across neighbors is preserved, as Figure 4 exemplifies in Appendix A.3. In particular, $h_1^{(2)}$ captures neither the degrees of its neighbors, nor the number and types of second-hop neighbors associated with each first-hop neighbor. Even so, aggregation still extracts useful information, and different aggregations concatenate complementary properties of neighborhoods:

1. Sum aggregation: Sums count, for each of the n distinct orthogonal feature vectors x_f, how many nodes in the k-hop neighborhood exhibit feature f. A classifier can extract it by computing $x_f^T h_v^{(k)}$. Note that nodes reachable through multiple length-k paths are counted multiple times.

2. Mean aggregation: Means can partially distinguish neighbors with different degrees by considering the fraction of nodes that exhibit a specific feature vector x_f. The quantity $x_f^T h_v^{(k)}$ weights each node i with feature x_f by $1/d_i$. Note that nodes reachable through multiple length-k paths are again counted with multiplicity.

3. Max/Min aggregation: Max aggregation on one-hot encoded features returns whether at least one node within k hops has a given feature. For large neighborhoods as k increases, this indicator quickly saturates, so increasing hops adds little further information. The same reasoning applies when taking the maximum entry of the orthogonal features. It also applies to the minimum: It indicates whether any node within k hops lacks the feature, and increasing k adds little further information.

Lossless neighborhood aggregation. When node features are real-valued in general, Corso et al. (2020) show that no single *continuous*, permutation-invariant aggregation function can be lossless for all multiset functions. This mirrors a classical topological obstruction (Netto, 1879): There is no continuous bijection $\mathbb{R} \to \mathbb{R}^2$ (Dauben, 1975). However, there can exist discontinuous bijections, namely space filling functions. We adopt a concrete construction based on ternary expansions and the Cantor set, adapted from a Kolmogorov-Arnold representation variant from Theorem 2 by Schmidt-Hieber (2021).

Theorem 4.2 (Thm. 2 of Schmidt-Hieber (2021))**.** *For any fixed $d \geq 2$, there exists a monotone function $\phi : [0,1] \to \mathcal{C}$ (the Cantor set) such that the map $\Phi(x_1, \ldots, x_d) = 3 \sum_{p=1}^{d} 3^{-p}\, \phi(x_p)$ is injective on $[0,1]^d$. Moreover, for every continuous $f : [0,1]^d \to \mathbb{R}$ there exists a continuous $g : \Phi([0,1]^d) \to \mathbb{R}$ with $f(x_1, \ldots, x_d) = g(\Phi(x_1, \ldots, x_d))$.*

Theorem 4.2 isolates all required discontinuity into a *fixed* aggregation. While Φ is not continuous, its inverse is, which makes the learnable $g := f \circ \Phi^{-1}$ inherit the continuity properties of f. Schmidt-Hieber (2021) has also quantified how much information is lost if g is learned instead of f. For f β-smooth with $\beta \leq 1$, there is no difference in the approximation rate. However, for higher order smoothness, the multivariate and univariate function approximation may vary. Note that this aggregation even remembers node identities. From this theorem we can learn the following insight:

> A lossless, fixed, even univariate neighborhood aggregation function exists, but it has to be discontinuous for general continuous features.

Implications and open challenges. When we encode neighborhoods via Φ and learn g so that $f = g \circ \Phi^{-1}$, the information content, smoothness, and approximation rates of the neighborhood function f transfer to g. However, this theoretical sufficiency does not guarantee strong empirical performance when Φ is used as a reducer for FAF (see Table 12, Appendix D). In Appendix A.1, we visualize how Φ maps 2D circles into the univariate Cantor set, and how Φ^{-1} can recover them continuously. We also compare against mean and std, and observe that Φ pushes inputs that are close together into far-apart representations, whereas mean and std bring together far-apart inputs that share commonalities. It is the case that, in practice, the simple statistics studied above often provide distributional summaries that downstream classifiers exploit more effectively.

An ideal aggregation would be both injective, like Φ, and would extract useful statistical insights, like mean. It is still an open challenge to design, or potentially learn, efficient embeddings that extract relevant information from neighborhoods, while easing the learning problem for the classifier (Burkholz et al., 2022). One might expect GNNs to learn such representations end-to-end without overfitting. Our experiments with FAFs (Table 1) suggest—despite some information loss at iterative hops—that simple reducers suffice for most standard node-classification benchmarks.

Experimentally, we find that mean aggregation alone is often among the top performers. This suggests that neighborhood distributions provide most task-relevant signal, and that neighbor degrees encode useful structural information, helping to distinguish the contribution of distinct neighbors. Consistent with this, the most relevant information is already provided in the immediate neighborhood ($k = 1$–2, see Table 8 on increasing hops) and the concatenation of this information is key so that it is not lost by repeated aggregations (see Table 11 on only using the last hop). Consequently, information loss at larger k is of little practical concern—except for two datasets that appear to require subtler information. Taken together, these observations motivate the following hypothesis:

Hypothesis: For most standard node-classification benchmarks, either the predictive signal is already concentrated within the first one or two hops, or current GNNs struggle to learn layer-wise aggregations that extract relevant information beyond mean or sum.

The first part of this hypothesis underscores the need for more real-world datasets where long-range interactions and richer aggregations matter, supporting prior calls to revisit benchmark design (Errica et al., 2020; Bechler-Speicher et al., 2025). Although some tasks like Roman Empire benefit from long-range signal, making deep graph models work reliably remains a challenge. Recent evidence (Zhou et al., 2025) indicates that graph models generally struggle to capture interactions beyond 13 hops, irrespective of over-smoothing, over-squashing, or vanishing gradients.

The second part of the hypothesis concerns the ability of GNNs to actually realize useful aggregations in practice. For instance, GNNs may not move far enough from their initializations. Indeed, the two datasets on which GNNs hold an advantage require linear residual transformations to realize that gap (Luo et al., 2024). Prior work also shows that GATs cannot flexibly adjust attention to shut off unhelpful neighbors (Mustafa & Burkholz, 2024b). This supports our results on rewiring the adjacency matrix before aggregation, as shown in § 3.1. If GATs could learn to prune the edges that we manually drop, they would enjoy similar gains.

Future work. We see opportunity for future work along three fronts that build directly on our findings:

1. Feature/reducer engineering: FAFs highlight untapped potential for designing meaningful node features that encode graph structure, require less learning, potentially preserve more—but ideally only relevant—information, and allow for higher learning efficiency. In combination with partial feature learning, they might form the basis of a new generation of graph based learning architectures.

2. Moving with and beyond injectivity: As our theory and experiments highlight, improving GNN expressiveness and thus injectivity alone is not likely to inspire practical improvements on current benchmarks, as those can be competitively solved with simple, non-injective aggregation. We therefore call for a shift in focus from mere injectivity to other learning properties—a gap to be addressed not only for FAFs but for GNNs in general.

3. New benchmarks: Enough information to solve current benchmark tasks is already contained in early hops and can be extracted with simple, non-injective aggregation. If we really want to showcase the capabilities of GNNs to learn meaningful features, we need more difficult benchmarks that require this ability.

In theory, fixed information-preserving aggregations can reduce graph learning to tabular prediction. In practice, task relevant representations and information preservation are a challenge. Progress likely requires both more amenable reducers and better tasks for evaluation.

5. Experiments

Setup. We report test performance for GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), and GraphSAGE (Hamilton et al., 2017) versus our approach, which feeds our Fixed Aggregation Features (FAFs) into MLPs (Figure 1 and Eqs. 1, 2), in Table 1. We aggregate up to the same hop as the GNN baselines. We obtain the best FAF variant from validation results, which are included in Table 6 (Appendix C). FAF_4 uses the reducers {mean, sum, max, min}, and is tuned with the hyperparameter grid from Luo et al. (2024); this makes our results directly comparable to their Graph Transformers and heterophily-aware architectures, where they find that classic GNNs can also rival them. Additional FAF variants include mean+std, mean only, max+std, max only, sum only, and std only. The best overall result is shown in **bold**, the second best is <u>underlined</u>. More details on the experimental setup are given in Appendix B. Detailed accuracy values of all FAF variants are in Appendix C.1. Training, validation, and test curves of all datasets for FAF_4 and GCN are shown in Fig. 5 in Appendix C.2. Ablation results are included in Appendix D.

Comparison to classic GNNs. Overall, we improve on 5 datasets, match within error or 1% on another 5, and trail on 4. On most datasets, FAF_4 performs comparably to mean+std. Among the ones within 1%, we have Coauthor-CS and Coauthor-Physics (Figs. 5f, 5g), which are the largest and most feature-rich; targeted feature selection may close the gap. Among the 4 trailing datasets, two are homophilic and two heterophilic; the homophilic tasks are close to parity. Citeseer exhibits optimization instability (Fig. 5e), and Cora has a large test-validation gap in GCNs (Fig. 5h), not present in any other. The two heterophilic datasets, Minesweeper and Roman-Empire (Figs. 5i, 5l) show larger performance drops. This behavior mirrors the decrease reported by Luo et al. (2024) when residual connections are removed. Notably, the best-performing FAFs on these two datasets use far fewer hops (4 and 2) than the GNN baselines (15 and 10), suggesting that key signal lies at longer ranges. The shallower FAFs under-aggregate relative to what those tasks require, but adding extra hops does not provide extra information, as discussed in § 4. We also show it in Table 8, where we concatenate up to different amount of hops; we show its validation counterpart in Table 9 (Appendix D). In fact, many datasets peak at $k = 2$, and either plateau or decrease in performance. These results also suggest that including later hops in any graph

Table 1. Test accuracy on node classification: FAFs against classic GNNs.

Dataset	computer	photo	ratings	chameleon	citeseer	coauthor-cs	coauthor-physics
GCN	93.58 ± 0.44	95.77 ± 0.27	53.86 ± 0.48	**44.62 ± 4.50**	**72.72 ± 0.45**	95.73 ± 0.15	**97.47 ± 0.08**
GAT	93.91 ± 0.22	96.45 ± 0.37	**55.51 ± 0.55**	42.90 ± 5.47	71.82 ± 0.65	96.14 ± 0.08	97.12 ± 0.13
SAGE	93.31 ± 0.17	96.17 ± 0.44	55.26 ± 0.27	43.11 ± 4.73	71.82 ± 0.81	**96.21 ± 0.10**	97.10 ± 0.09
$FAF_{bestval}$	**94.01 ± 0.21**	**96.54 ± 0.13**	55.09 ± 0.24	42.96 ± 2.45	70.48 ± 1.24	95.37 ± 0.17	97.05 ± 0.18

Dataset	cora	minesweeper	pubmed	questions	roman-empire	squirrel	wikics
GCN	**84.38 ± 0.81**	97.48 ± 0.06	80.00 ± 0.77	78.44 ± 0.23	**91.05 ± 0.15**	44.26 ± 1.22	80.06 ± 0.81
GAT	83.02 ± 1.21	97.00 ± 1.02	79.80 ± 0.94	77.72 ± 0.71	90.38 ± 0.49	39.31 ± 2.42	**81.01 ± 0.23**
SAGE	83.18 ± 0.93	**97.72 ± 0.70**	77.42 ± 0.40	76.75 ± 1.07	90.41 ± 0.10	40.22 ± 1.47	80.57 ± 0.42
$FAF_{bestval}$	82.84 ± 0.63	90.00 ± 0.39	**80.96 ± 1.06**	**78.69 ± 0.50**	78.11 ± 0.38	**44.59 ± 1.62**	80.25 ± 0.34

learning method may provoke overfitting, which can be an alternative explanation for the degradation of performance on deep GNNs apart from over-smoothing—as here it is not variable by construction. We also include an MLP with $k = 0$, which performs worse than the other models.

Best hyperparameters. We include the best hyperparameters found for FAF_4 in Table 5 (Appendix B). All FAF variants benefit from normalization components, as aggregated features can vary widely in scale across reducers and hops. Compared with GNNs, FAFs typically favor larger learning rates, which can yield faster training, improved generalization via implicit regularization, and feature sparsity (Mohtashami et al., 2023; Sadrtdinov et al., 2024). Dropout levels, however, are broadly similar to those used for GNNs. This suggests that dropout's gains on these node-classification tasks are driven more by dataset properties than by the specifics of training graph convolutions, which nuances prior interpretations (Luo et al., 2025b).

Ablation on single reducers. Concatenating multiple aggregations has advantages and drawbacks. On the plus side, an MLP can learn to weight each reducer, removing the need to pick one per dataset. Because our individual reducers are not lossless, different datasets may favor different ones; moreover, adding informative, correlated covariates can improve robustness and reduce variance (Reddy et al., 2026). On the downside, concatenation increases input dimensionality, with corresponding memory and optimization costs. Table 7 (Appendix C) reports test results when using a single aggregation at a time. Note that this resembles a simple feature selection over FAF_4. We keep the same hyperparameters as FAF_4 to isolate the effect of the aggregation choice, though the lower dimensionality could allow for alternative settings that further improve performance. Surprisingly, a single reducer often suffices, though the preferred choice varies by dataset. The mean is most frequently strongest, sometimes surpassing FAF_4 and $FAF_{mean,std}$—i.e., on Amazon-Computer and Amazon-Photo. This may reflect optimization challenges from high-dimensional inputs or increased overfitting. Still, mean is not universally best, as sum and max win on some datasets (e.g., Citeseer favors sum; Amazon-Ratings favors max). Combining reducers therefore remains beneficial when one wishes to avoid committing to a specific one a priori.

Comparison to one-layer classifier and last hop. Some prior simplifications of GNNs (Wu et al., 2019; Micheli & Tortorella, 2024) effectively fix the aggregation and train a single linear layer on the final-hop representation. In contrast, we concatenate representations from all hops and train a well-tuned MLP classifier. This choice is crucial for matching GNN performance. As shown in Table 10 (Appendix D), MLPs consistently outperform a single linear layer applied to the same concatenated features, indicating that their nonlinearity and increased capacity are important to learn from multi-hop features. Moreover, Table 11 (Appendix D) shows that only using the last hop lacks important information that is not transmitted across aggregations.

Kolmogorov-Arnold aggregation. Our hypothesis that Roman-Empire lags due to information loss is reinforced by a FAF that uses the Kolmogorov–Arnold (KA) function Φ, which is theoretically lossless (Theorem 4.2). As we exemplified in Fig. 3, in practice KA is hard to use for classification. We observe in Table 12 (Appendix D) that in some datasets it struggles to fit during training, while others show mild overfitting. Nevertheless, in Roman-Empire this obtains a test accuracy of 80.33 ± 0.47, the highest among all FAFs, suggesting that providing full neighborhood information helps close the gap in this task. This, in turn, highlights the need for benchmarks where predictive signal genuinely arises at distant hops in complex ways.

6. Conclusions

We have introduced Fixed Aggregation Features (FAFs), a non-learnable tabular mapping from local neighborhoods of graph features to univariate representations that an MLP can learn to classify. Our analysis shows that fixed, injective

neighborhood aggregation functions exist, linking multiset expressivity to Kolmogorov–Arnold factorizations; thus learned message passing is not required for expressivity in theory. But in practice, common non-injective reducers (mean, sum, max, min) train more reliably, underscoring a gap between what is expressive in principle and what is reliably learnable. We also highlight the practical advantages of a tabular view, such as access to the rich tabular toolkit of interpretability and tuning, and isolated representation from inference so we can attribute gains or failures to the features themselves rather than to message-passing optimization.

On node classification datasets, FAFs are a strong baseline: They match or beat classic GNNs on many benchmarks, and trail only on two datasets needing longer-range interactions, where residualized GNNs help. Two ablations explain most gains: A well-tuned MLP beats a single linear classifier on top of FAF features, and concatenating all hop beats using only the last hop. This is consistent with later hops losing detail for these practical aggregation schemes. Surprisingly, two hops usually suffice, suggesting either limited signal in current benchmarks, or difficulty training deep GNNs to exploit more of it. While our theory carries over other downstream tasks, other benchmarks may surface different constraints that could alter the empirical outcomes.

Our findings have immediate implications. We recommend always including a tuned FAF baseline in future studies to calibrate what fixed aggregation alone can achieve; re-evaluating—and, when appropriate, retiring—datasets on which FAFs reach state-of-the-art performance; and developing benchmarks that genuinely require long-range dependencies and inter-hop dynamics. More broadly, we advocate for simplifying models and balancing expressiveness against optimizability, rather than assuming that extra parameters or higher expressiveness extract more relevant signal than simple baselines. Notably, several phenomena that are often blamed on graph architectures—overfitting, depth-related degradation, and sensitivity to dropout—also arise in tabular settings, indicating that some limitations may stem from dataset properties rather than the graph-aware architectures alone.

Acknowledgments and Disclosure of Funding

The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. for funding this project by providing computing time on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC). We also gratefully acknowledge funding from the European Research Council (ERC) under the Horizon Europe Framework Programme (HORIZON) for proposal number 101116395 SPARSE-ML.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Bazhenov, G., Platonov, O., and Prokhorenkova, L. Graphland: Evaluating graph machine learning models on diverse industrial data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=Gyq8lMgdk5.

Bechler-Speicher, M., Finkelshtein, B., Frasca, F., Müller, L., Tönshoff, J., Siraudin, A., Zaverkin, V., Bronstein, M. M., Niepert, M., Perozzi, B., Galkin, M., and Morris, C. Position: Graph learning will lose relevance due to poor benchmarks. In *Forty-second International Conference on Machine Learning Position Paper Track*, 2025. URL https://openreview.net/forum?id=nDFpl2lhoH.

Bongini, P., Pancino, N., Scarselli, F., and Bianchini, M. Biognn: How graph neural networks can solve biological problems. In *Artificial Intelligence and Machine Learning for Healthcare*, pp. 211–231. Springer, 2023.

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

Burkholz, R., Laha, N., Mukherjee, R., and Gotovos, A. On the existence of universal lottery tickets. In *International Conference on Learning Representations*, 2022.

Cai, C. and Wang, Y. A simple yet effective baseline for non-attributed graph classification. In *ICLR Workshop: Representation Learning on Graphs and Manifolds*, 2019. URL https://arxiv.org/pdf/1811.03508.

Carlo, G. D., Mastropietro, A., and Anagnostopoulos, A. Kolmogorov–arnold graph neural networks, 2025. URL https://openreview.net/forum?id=udfjje2xXb.

Chen, J., Gao, K., Li, G., and He, K. NAGphormer: A tokenized graph transformer for node classification in large graphs. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=8KYeilT3Ow.

Chien, E., Peng, J., Li, P., and Milenkovic, O. Adaptive universal generalized PageRank graph neural network. In

International Conference on Learning Representations, 2021. URL https://openreview.net/forum?id=n6jl7fLxrP.

Choi, J., Kang, W., Kim, M., Kim, J., and Park, N. Can tabpfn compete with gnns for node classification via graph tabularization?, 2025. URL https://arxiv.org/abs/2512.08798.

Corso, G., Cavalleri, L., Beaini, D., Liò, P., and Velickovic, P. Principal neighbourhood aggregation for graph nets. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Coupette, C., Wayland, J., Simons, E., and Rieck, B. No metric to rule them all: Toward principled evaluations of graph-learning datasets. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=XbmBNwrfG5.

Dauben, J. W. The invariance of dimension: Problems in the early development of set theory and topology [1]. *Historia Mathematica*, 2(3):273–288, 1975. ISSN 0315-0860. doi: https://doi.org/10.1016/0315-0860(75)90066-X. URL https://www.sciencedirect.com/science/article/pii/031508607590066X.

Deng, C., Yue, Z., Yu, C., Sarar, G., Carey, R., Jain, R., and Zhang, Z. Less is more: Hop-wise graph attention for scalable and generalizable learning on circuits. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, DAC '24, New York, NY, USA, 2024a. Association for Computing Machinery. ISBN 9798400706011. doi: 10.1145/3649329.3657386. URL https://doi.org/10.1145/3649329.3657386.

Deng, C., Yue, Z., and Zhang, Z. Polynormer: Polynomial-expressive graph transformer in linear time. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=hmv1LpNfXa.

Donghi, G., Pasa, L., Oneto, L., Gallicchio, C., Micheli, A., Anguita, D., Sperduti, A., and Navarin, N. Investigating over-parameterized randomized graph networks. *Neurocomputing*, 606:128281, 2024. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2024.128281. URL https://www.sciencedirect.com/science/article/pii/S092523122401052X.

Donnelly, J., Katta, S., Rudin, C., and Browne, E. P. The rashomon importance distribution: Getting RID of unstable, single model-based variable importance. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=TczT2jiPT5.

Eremeev, D., Bazhenov, G., Platonov, O., Babenko, A., and Prokhorenkova, L. Turning tabular foundation models into graph foundation models. In *New Perspectives in Graph Machine Learning*, 2025. URL https://openreview.net/forum?id=KyiooU4ECI.

Errica, F., Podda, M., Bacciu, D., and Micheli, A. A fair comparison of graph neural networks for graph classification. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HygDF6NFPB.

Frasca, F., Rossi, E., Eynard, D., Chamberlain, B., Bronstein, M., and Monti, F. Sign: Scalable inception graph neural networks. In *ICML 2020 Workshop on Graph Representation Learning and Beyond*, 2020.

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

Gasteiger, J., Bojchevski, A., and Günnemann, S. 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.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1263–1272. JMLR.org, 2017.

Gorishniy, Y., Rubachev, I., and Babenko, A. On embeddings for numerical features in tabular deep learning. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24991–25004. Curran Associates, Inc., 2022.

Grötschla, F., Xie, J., and Wattenhofer, R. Benchmarking positional encodings for gnns and graph transformers. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, KDD '26, pp. 2663–2674, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400722585. doi: 10.1145/3770854.3785701. URL https://doi.org/10.1145/3770854.3785701.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Hayler, A., Huang, X., Ceylan, I. I., Bronstein, M. M., and Finkelshtein, B. Of graphs and tables: Zero-shot node classification with tabular foundation models. In *New Perspectives in Graph Machine Learning*, 2025. URL https://openreview.net/forum?id=u3vyZf5Jv2.

Jamadandi, A., Rubio-Madrigal, C., and Burkholz, R. Spectral graph pruning against over-squashing and over-smoothing. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=EMkrwJY2de.

Kelesis, D., Fotakis, D., and Paliouras, G. Partially trained graph convolutional networks resist oversmoothing. *Machine Learning*, 114(10):211, Aug 2025. ISSN 1573-0565. doi: 10.1007/s10994-025-06865-3. URL https://doi.org/10.1007/s10994-025-06865-3.

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

Kolmogorov, A. N. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. In *Doklady Akademii Nauk*, volume 114(5), pp. 953–956. Russian Academy of Sciences, 1957.

Kong, K., Chen, J., Kirchenbauer, J., Ni, R., Bruss, C. B., and Goldstein, T. GOAT: A global transformer on large-scale graphs. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17375–17390. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/kong23a.html.

Li, X., Zhu, R., Cheng, Y., Shan, C., Luo, S., Li, D., and Qian, W. Finding global homophily in graph neural networks when meeting heterophily. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 13242–13256. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/li22ad.html.

Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljacic, M., Hou, T. Y., and Tegmark, M. KAN: Kolmogorov–arnold networks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Ozo7qJ5vZi.

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 4768–4777, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Luo, Y., Shi, L., and Wu, X.-M. Classic GNNs are strong baselines: Reassessing GNNs for node classification. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=xkljKdGe4E.

Luo, Y., Shi, L., and Wu, X.-M. Can classic GNNs be strong baselines for graph-level tasks? simple architectures meet excellence. In *Forty-second International Conference on Machine Learning*, 2025a. URL https://openreview.net/forum?id=ZH7YgIZ3DF.

Luo, Y., Wu, X.-M., and Zhu, H. Beyond random masking: When dropout meets graph convolutional networks. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=PwxYoMvmvy.

Maurya, S. K., Liu, X., and Murata, T. Simplifying approach to node classification in graph neural networks. *Journal of Computational Science*, 62:101695, 2022. ISSN 1877-7503. doi: https://doi.org/10.1016/j.jocs.2022.101695. URL https://www.sciencedirect.com/science/article/pii/S1877750322000990.

McCallum, A. K., Nigam, K., Rennie, J., and Seymore, K. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000. doi: 10.1023/A:1009953814988. URL https://doi.org/10.1023/A:1009953814988.

Mernyei, P. and Cangea, C. Wiki-cs: A wikipedia-based benchmark for graph neural networks. In *ICML Workshop: Graph Representation Learning and Beyond*, 2020. URL https://arxiv.org/abs/2007.02901.

Micheli, A. and Tortorella, D. Designs of graph echo state networks for node classification. *Neurocomputing*, 597:127965, 2024. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2024.127965. URL https://www.sciencedirect.com/science/article/pii/S0925231224007367.

Micheli, A. and Tortorella, D. An empirical evaluation of rewiring approaches in graph neural networks. *Pattern Recognition Letters*, 196:134–141, 2025. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2025.05.

021. URL https://www.sciencedirect.com/science/article/pii/S0167865525002168.

Mohtashami, A., Jaggi, M., and Stich, S. U. Special properties of gradient descent with large learning rates. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 25082–25104. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/mohtashami23a.html.

Mustafa, N. and Burkholz, R. Dynamic rescaling for training GNNs. In *Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=IfZwSRpqHl.

Mustafa, N. and Burkholz, R. GATE: How to keep out intrusive neighbors. In *Forty-first International Conference on Machine Learning*, 2024b. URL https://openreview.net/forum?id=Sjv5RcqfuH.

Mustafa, N., Bojchevski, A., and Burkholz, R. Are GATs out of balance? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=qY7UqLoora.

Namata, G., London, B., Getoor, L., and Huang, B. Query-driven active surveying for collective classification. In *Proceedings of the Workshop on Mining and Learning with Graphs*, 2012.

Platonov, O., Kuznedelev, D., Babenko, A., and Prokhorenkova, L. Characterizing graph datasets for node classification: homophily-heterophily dichotomy and beyond. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023a. Curran Associates Inc.

Platonov, O., Kuznedelev, D., Diskin, M., Babenko, A., and Prokhorenkova, L. A critical look at evaluation of gnns under heterophily: Are we really making progress? In *The Eleventh International Conference on Learning Representations*, 2023b.

Qarkaxhija, L., Wegner, A. E., and Scholtes, I. Link prediction with untrained message passing layers. In *The Third Learning on Graphs Conference*, 2024. URL https://openreview.net/forum?id=IRFfBpbdI9.

Ramachandra, S., Degraeve, V., Vandewiele, G., Steenwinckel, B., Hoecke, S. V., and Ongenae, F. Rr-gcn: Exploring untrained random embeddings for relational graphs. *International Journal of Software Engineering and Knowledge Engineering*, 35(06):809–834, 2025. doi: 10.1142/S0218194025500184. URL https://doi.org/10.1142/S0218194025500184.

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. Recipe for a general, powerful, scalable graph transformer. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 14501–14515. Curran Associates, Inc., 2022.

Reddy, A. G., Rubio-Madrigal, C., Burkholz, R., and Muandet, K. When shift happens - confounding is to blame. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=sFjxg8cyJS.

Roth, A., Bause, F., Kriege, N. M., and Liebig, T. Preventing representational rank collapse in mpnns by splitting the computational graph. In Wolf, G. and Krishnaswamy, S. (eds.), *Proceedings of the Third Learning on Graphs Conference*, volume 269 of *Proceedings of Machine Learning Research*, pp. 14:1–14:24. PMLR, 26–29 Nov 2025. URL https://proceedings.mlr.press/v269/roth25a.html.

Rozemberczki, B., Allen, C., and Sarkar, R. Multi-Scale attributed node embedding. *Journal of Complex Networks*, 9(2), 05 2021. ISSN 2051-1329. doi: 10.1093/comnet/cnab014. URL https://doi.org/10.1093/comnet/cnab014.

Rubio-Madrigal, C., Jamadandi, A., and Burkholz, R. GNNs getting ComFy: Community and feature similarity guided rewiring. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=g6v09VxgFw.

Sadrtdinov, I., Kodryan, M., Pokonechny, E., Lobacheva, E., and Vetrov, D. Where do large learning rates lead us? In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 58445–58479. Curran Associates, Inc., 2024.

Schmidt-Hieber, J. The Kolmogorov–Arnold representation theorem revisited. *Neural Networks*, 137:119–126, 2021. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2021.01.020. URL https://www.sciencedirect.com/science/article/pii/S0893608021000289.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI Magazine*, 29 (3):93, Sep. 2008. doi: 10.1609/aimag.v29i3.2157. URL https://ojs.aaai.org/index.php/aimagazine/article/view/2157.

Sharma, K., Lee, Y.-C., Nambi, S., Salian, A., Shah, S., Kim, S.-W., and Kumar, S. A survey of graph neural networks for social recommender systems. *ACM Computing Surveys*, 56(10):1–34, June 2024. ISSN 1557-7341. doi: 10.1145/3661821. URL http://dx.doi.org/10.1145/3661821.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. *Relational Representation Learning Workshop, NeurIPS*, 2018.

Shirzad, H., Velingker, A., Venkatachalam, B., Sutherland, D. J., and Sinop, A. K. Exphormer: Sparse transformers for graphs. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31613–31632. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/shirzad23a.html.

Stoll, T., Qian, C., Finkelshtein, B., Parviz, A., Weber, D., Frasca, F., Shavit, H., Siraudin, A., Mielke, A., Anastacio, M., Müller, E., Bechler-Speicher, M., Bronstein, M., Galkin, M., Hoos, H., Niepert, M., Perozzi, B., Tönshoff, J., and Morris, C. Graphbench: Next-generation graph learning benchmarking, 2026. URL https://arxiv.org/abs/2512.04475.

Topping, J., Giovanni, F. D., Chamberlain, B. P., Dong, X., and Bronstein, M. M. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=7UmjRGzp-A.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph Attention Networks. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ. accepted as poster.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6861–6871. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/wu19e.html.

Wu, Q., Zhao, W., Li, Z., Wipf, D. P., and Yan, J. Nodeformer: A scalable graph structure learning transformer for node classification. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 27387–27401. Curran Associates, Inc., 2022.

Wu, Q., Zhao, W., Yang, C., Zhang, H., Nie, F., Jiang, H., Bian, Y., and Yan, J. SGFormer: Simplifying and empowering transformers for large-graph representations. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 64753–64773. Curran Associates, Inc., 2023.

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

Zhang, W., Yin, Z., Sheng, Z., Li, Y., Ouyang, W., Li, X., Tao, Y., Yang, Z., and Cui, B. Graph attention multi-layer perceptron. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, pp. 4560–4570, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850. doi: 10.1145/3534678.3539121. URL https://doi.org/10.1145/3534678.3539121.

Zheng, Y., Luan, S., and Chen, L. What is missing for graph homophily? disentangling graph homophily for graph neural networks. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 68406–68452. Curran Associates, Inc., 2024.

Zheng, Y., Li, X., Luan, S., Peng, X., and Chen, L. Let your features tell the differences: Understanding graph convolution by feature splitting. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=I9omfcWfMp.

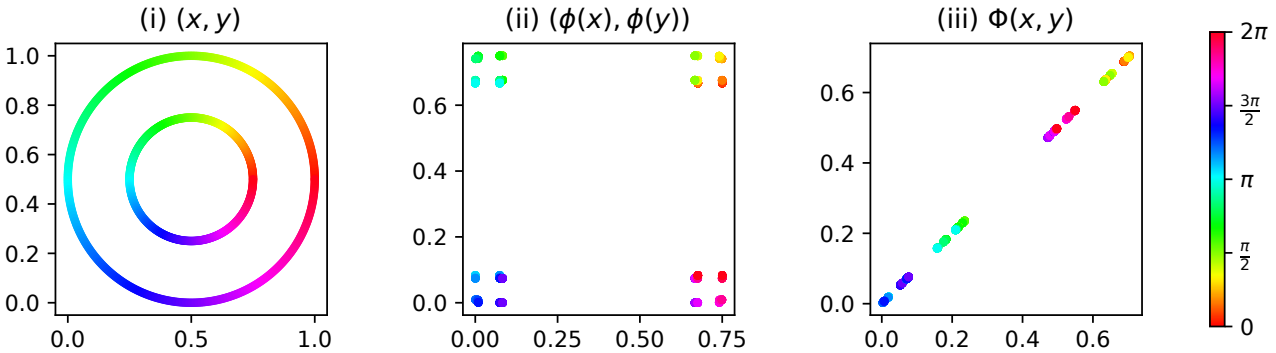
Zhou, D., Kharlamov, E., and Kostylev, E. V. GLora: A benchmark to evaluate the ability to learn long-range dependencies in graphs. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=2jf5x5XoYk.

Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., and Koutra, D. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33, 2020.

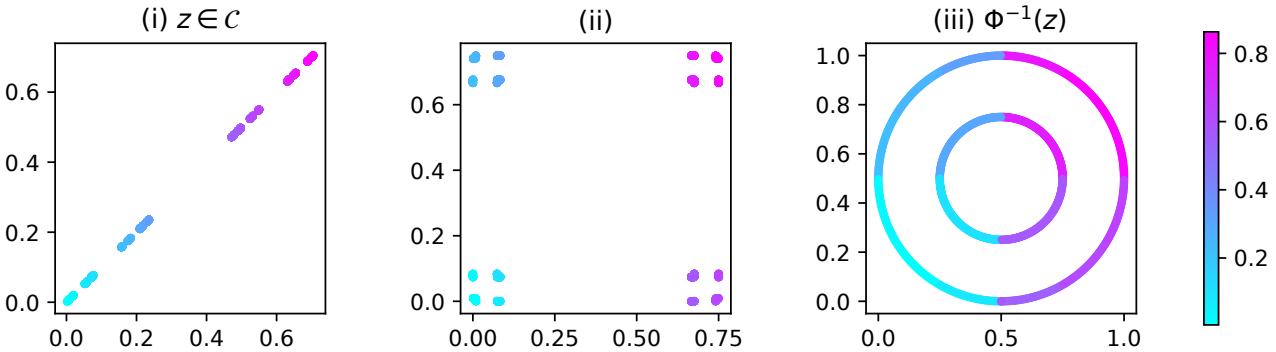
Zhu, J., Rossi, R. A., Rao, A., Mai, T., Lipka, N., Ahmed, N. K., and Koutra, D. Graph neural networks with heterophily. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12): 11168–11176, May 2021. doi: 10.1609/aaai.v35i12.17332. URL https://ojs.aaai.org/index.php/AAAI/article/view/17332.

A. Study of Reducers for Neighborhood Aggregation

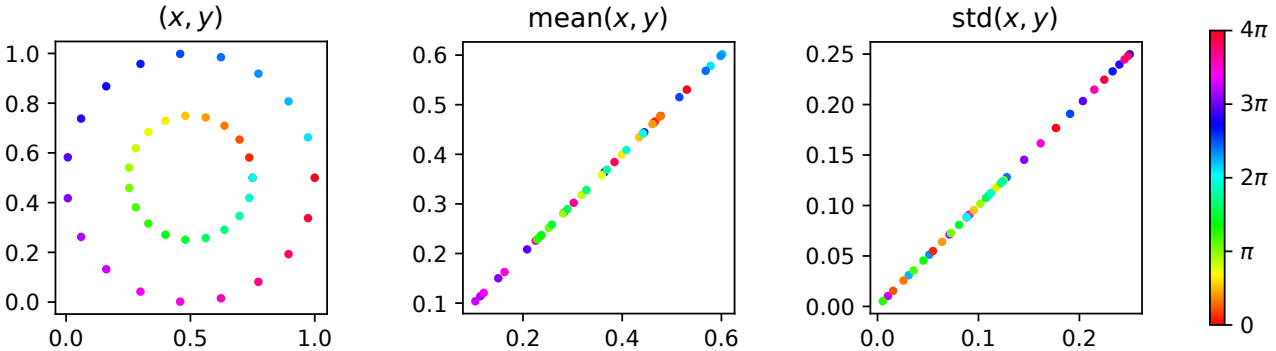
A.1. Kolmogorov-Arnold Function Φ and its Continuous Inverse



(a) Aggregating from (x, y) to $(\phi(x), \phi(y))$ to $\Phi(x, y)$.



(b) Recovering information from the aggregated variable $z \in \mathcal{C}$ to $\Phi^{-1}(z)$.



(c) Mean and std as aggregation functions.

Figure 3. Functions Φ (Thm 4.2)—and its inverse—, mean and std. Circles and square-like panels (a.i, a.ii, b.ii, b.iii, c.i) live in the 2D space, while segments and Cantor sets (a.iii, b.i, c.ii, c.iii) live in 1D. Colors in (a) and (c) are based on angles on 2D, while colors in (b) are based on position.

Here we showcase the behavior of Φ from the Kolmogorov-Arnold representation from Theorem 4.2. In Fig. 3a, we see the Φ image of two circles colored by their angle. Colors that were close together end up in separate parts of the Cantor set; for instance, oranges and reds, or purples and blues. In contrast, in Fig. 3b we see Φ^{-1} maps the Cantor set to the circles in such a way that all colors maintain their closeness.

If we use Φ as a fixed neighborhood aggregation, the classifier on top needs to learn to reverse it, therefore it is advantageous to have a continuous inverse. However, this does not give information about neighborhood distributions like the commonly

used mean, sum, and max. In Fig. 3c we show the behavior of mean and std; mean gives approximate location but fuses together points that are very far apart. For instance, blues and reds have an unusually large first (but different) coordinate and are mapped to the center; however, this information can be recovered with std.

The fact that the injective aggregator derived from theory is not practically useful is itself an insight. Our findings suggest that injectivity, and thus whether information is preserved by the aggregation, may not be the right lens through which to explain GNN performance. What seems to matter more is the model's ability to learn to discard less relevant information as depth increases. Preserving all neighborhood information across many hops may simply not be tractable or useful.

A.2. Proof of Main Theorem

For convenience, the following theorem restates Theorem 4.1 of the main paper. Here, "orthogonal features" refers to the distinct feature states that occur in the graph: if $\mathcal{X} = \{x_1, \ldots, x_n\} \subset \mathbb{R}^F$ denotes these states, then $x_i^\top x_j = 0$ for $i \neq j$. This does not require $x_u^\top x_v = 0$ for every pair of distinct nodes $u \neq v$.

Theorem (1-hop aggregation). Assume the features $\mathcal{X}$ are orthogonal. Then, the function $h(X) = \sum_{x \in X} x$ defined on multisets $X \subseteq \mathcal{X}$ of bounded size is injective. Moreover, any multiset function f can be decomposed as $f(X) = g\big(\sum_{x \in X} x\big)$ for some function g.

Proof. Note that the multiset X is fully characterized by the number n_f of nodes in the set that have a feature x_f for all possible features x_f. Our objective is to show that this information is contained in the aggregated form $h(X) = \sum_{x \in X} x$.

Let us assume that the features are orthogonal. Accordingly, the features x_v of each node v assume one of a finite number of possible states $x_1, \cdots x_n \in \mathbb{R}^{n_f}$ with $n_f \geq n$ and $x_i^T x_j = 0$ for any pair $i, j \in V$ with $i \neq j$. Note that the number of possible feature states n must be finite even in an infinitely large graph, as long as the number of features are finite, i.e. $n_f < \infty$. Since the feature values must be pairwise orthogonal, there can maximally exist n_f distinct feature vectors, as n_f orthogonal vectors would form a basis of $\mathbb{R}^{n_f}$ and therefore an additional vector would become linearly dependent on the basis vectors.

So let us consider any of the possible feature states x_f. Then $x_f^T h(X) = \sum_{x \in X} x_f^T x = \sum_{x \in X, x = x_f} 1 = n_{x_f}$ counts the number of nodes in the set S that have features x_f. Since this holds for all possible feature vectors x_f, all information about any multiset X is preserved by $h(X)$.

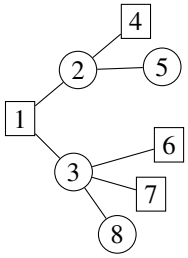
Accordingly, we can write any multiset function $f(X) = (f(n_{x_1}), \cdots, f(n_{x_n}))$ (which transforms the feature counts) into a function g that extracts first the count information from the sum $h(X)$. Concretely, we can define: $g(h(X))_f := f(n_{x_f}) = f(x_f^T h(X))$. □

While the theorem shares a similarity with proofs of injectivity used in prior work to establish expressive power for learned aggregation, most notably GIN (Xu et al., 2019), our result is a strict generalization, as GIN is restricted to discrete features and one-hot encodings. Crucially, the injective aggregator from their proof is also task-independent and can therefore be treated as fixed, even if this is not considered for their architectural development. We instead question whether these injectivity arguments actually explain the empirical success of GNNs.

A.3. Loss of Information Over Second Hops

We now explore an example of a computational tree of a node with two rounds of sum aggregation, and the qualitative kind of information that is lost from the first to the second hop. As shown by Xu et al. (2019) and generalized in Theorem 4.1, sum is injective over one-hot encoded features, but the second aggregation round sums features that are not necessarily orthogonal, and therefore loses neighborhood information. The computational tree and calculation of hops are displayed in Figure 4.

Given the previous hops $H_1^{(0)} = \binom{1}{0}$ and $H_1^{(1)} = \binom{0}{2}$, and the second hop $H_1^{(2)} = \binom{5}{2}$, what other combinations of two-hop neighborhoods can there be for node 1? Apart from itself, node 1's two-hop neighbors are in a $\binom{3}{2}$ feature ratio. However, we have lost the ability to recognize a) how many belong to each of its one-hop neighbor; and b) the distribution or homogeneity of each neighborhood. In reality, these 5 nodes are approximately spread out in number and distribution across the one-hop neighbors 2 and 3. But alternatively, all 5 nodes could have belonged to node 2, or all squared nodes could have belonged to node 3.



Feature encodings: $\square = \binom{1}{0}$ and $\bigcirc = \binom{0}{1}$

$$H_1^{(0)} = \binom{1}{0}$$
$$H_2^{(0)} = H_3^{(0)} = \binom{0}{1} \implies H_1^{(1)} = \binom{0}{2}$$
$$H_2^{(1)} = \binom{2}{1} \text{ and } H_3^{(1)} = \binom{3}{1} \implies H_1^{(2)} = \binom{5}{2}$$

Figure 4. Example of a two-hop neighborhood with one-hot encoded features and sum aggregation.

Note that, without previous hops $H_1^{(0)}$ and $H_1^{(1)}$, we cannot even distinguish node 1's original features, nor distinguish its presence as a neighbor in its one-hop neighbors. Therefore, concatenating all hops is advantageous.

For completeness, we also include the calculations for mean aggregation, which are qualitatively similar to the sum in this case.

$$H_1^{(0)} = \binom{1}{0}\,; \quad H_2^{(0)} = H_3^{(0)} = \binom{0}{1} \quad \implies H_1^{(1,m)} = \binom{0}{1}$$
$$H_2^{(1,m)} = \binom{2/3}{1/3} \text{ and } H_3^{(1.m)} = \binom{3/4}{1/4} \quad \implies H_1^{(2,m)} = \binom{17/24}{7/24}$$

A good lossless aggregation scheme should take all possible second-hop neighborhood distributions and map them to values that would not lose information when aggregated. For instance, choosing a, b such that $\binom{2}{1} \to a$ from node 2, and $\binom{3}{1} \to b$ from node 3, so that, $H_1^{(2)} = a + b$ could recover both values separately. Naturally, mapping them to one-hot encodings per distribution would suffice, but it would grow exponentially. This opens the door for better suitable fixed aggregations, or perhaps other kinds of learnable aggregation beyond current understanding of message passing.

B. Experimental Details

The codebase can be found here: `https://github.com/celrm/fixed-aggregation-features`.

B.1. Dataset Details

Datasets are taken directly from the setup of Luo et al. (2024), which includes varied node classification datasets. Here in Table 2 we include for completeness the same overview of these benchmarks.

Table 2. Details of the node classification datasets.

Dataset	Type	# Nodes	# Edges	# Features	Classes	Metric	Origin
Cora	Homophily	2,708	5,278	1,433	7	Accuracy	(McCallum et al., 2000)
CiteSeer	Homophily	3,327	4,522	3,703	6	Accuracy	(Sen et al., 2008)
PubMed	Homophily	19,717	44,324	500	3	Accuracy	(Namata et al., 2012)
Computer	Homophily	13,752	245,861	767	10	Accuracy	(Shchur et al., 2018)
Photo	Homophily	7,650	119,081	745	8	Accuracy	(Shchur et al., 2018)
CS	Homophily	18,333	81,894	6,805	15	Accuracy	(Shchur et al., 2018)
Physics	Homophily	34,493	247,962	8,415	5	Accuracy	(Shchur et al., 2018)
WikiCS	Homophily	11,701	216,123	300	10	Accuracy	(Mernyei & Cangea, 2020)
Squirrel	Heterophily	2,223	46,998	2,089	5	Accuracy	(Rozemberczki et al., 2021)
Chameleon	Heterophily	890	8,854	2,325	5	Accuracy	(Rozemberczki et al., 2021)
Roman-Empire	Heterophily	22,662	32,927	300	18	Accuracy	(Platonov et al., 2023b)
Amazon-Ratings	Heterophily	24,492	93,050	300	5	Accuracy	(Platonov et al., 2023b)
Minesweeper	Heterophily	10,000	39,402	7	2	ROC-AUC	(Platonov et al., 2023b)
Questions	Heterophily	48,921	153,540	301	2	ROC-AUC	(Platonov et al., 2023b)

B.2. Hyperparameters

Each experiment is run on an NVIDIA A100 GPU. The setup is taken from Luo et al. (2024). That is, for a maximum of 2500 epochs, we tune the following parameters:

1. DROPOUT $\in$ (0.0 0.2 0.3 0.5 0.7)
2. LR $\in$ (0.01 0.005 0.001 0.0001)
3. NORMALIZATION $\in$ (ln bn none)
4. HIDDEN CHANNELS $\in$ (64 256 512)

While Luo et al. (2024) includes weight decay as a hyperparameter, there are no concrete ranges specified for it. Therefore, we tune all the different values from the best runs of the given datasets:

5. WEIGHT DECAY $\in$ (0.0 1e-2 1e-3 5e-4 5e-5).

Moreover, Luo et al. (2024) tunes the local layers from 1 to 10 or 15. We instead take for each dataset the same value that they have found best for the GNNs, and use it to construct our fixed aggregation features up to that depth. In some cases where there are too many features, we restrict the depth to a smaller value, thus including a strict subset of features instead. This serves as an ad hoc feature selection to reduce overfitting.

We include as hyperparameter the MLP depth. We also include results for MLP = 1 in Table 10.

6. MLP LAYERS $\in$ (2 3 5).

On the other hand, we do not include linear residual connections, as these are used to bypass the convolutional layers in the classical GNNs. This creates a direct difference on the two datasets that most benefit from this component, Minesweeper and Roman-Empire.

In Table 5 we include the best hyperparameter choices for the four models: GCN, GAT, GraphSAGE, and FAF_4, the results of which are in Table 6. We run baselines directly from the setup of Luo et al. (2024), and we sweep FAF_4 with the same ranges included in their work. Each dataset has a specific number of splits given by their setup (from 3 to 10), which we then average.

B.3. Runtime and Feature Reduction

We also include in Table 3 the training runtime of our algorithms, grouped by the number of reducers: FAF_i with $i = |\mathcal{R}|$—as MLPs with the same input width will have the same training time. Note that all runs have the same number of epochs (2500), as in the original setup, and all datasets match the number of runs of the setup.

Table 3. Empirical training time in seconds of FAF and GNN models, averaged over runs.

Dataset	GCN	GAT	SAGE	FAF_4	FAF_2	FAF_1	MLP
computer	127.30 ± 0.58	29.33 ± 3.22	45.33 ± 0.58	42.67 ± 0.58	25.00 ± 0.00	17.00 ± 0.00	9.33 ± 0.58
photo	82.00 ± 0.00	26.00 ± 0.00	33.33 ± 0.58	66.67 ± 0.58	36.00 ± 0.00	21.33 ± 0.58	7.33 ± 0.58
ratings	140.30 ± 0.58	161.30 ± 0.58	330.00 ± 0.00	46.00 ± 0.00	26.00 ± 0.00	17.33 ± 0.58	10.00 ± 0.00
chameleon	22.30 ± 0.48	15.20 ± 0.42	18.00 ± 0.00	44.50 ± 0.97	22.50 ± 0.53	15.10 ± 0.32	10.00 ± 0.00
citeseer	16.20 ± 0.45	20.00 ± 0.00	26.00 ± 0.00	62.80 ± 1.79	36.20 ± 0.45	25.80 ± 1.79	13.60 ± 0.55
coauthor-cs	157.00 ± 0.00	70.33 ± 0.58	301.30 ± 20.50	296.30 ± 1.16	159.00 ± 0.00	91.00 ± 0.00	26.33 ± 0.58
coauthor-physics	65.33 ± 0.58	190.00 ± 0.00	400.30 ± 0.58	2383.00 ± 0.58	1004.00 ± 1.00	532.70 ± 0.58	183.00 ± 0.00
cora	16.40 ± 0.89	20.20 ± 0.45	11.20 ± 0.45	46.80 ± 0.45	26.00 ± 0.00	17.00 ± 0.00	8.20 ± 0.45
minesweeper	69.67 ± 0.58	100.30 ± 0.58	68.67 ± 2.08	19.00 ± 0.00	20.00 ± 3.46	19.00 ± 0.00	18.00 ± 1.73
pubmed	20.20 ± 0.45	42.00 ± 0.00	78.20 ± 0.45	34.00 ± 0.00	19.20 ± 0.45	12.00 ± 0.00	6.20 ± 0.45
questions	650.00 ± 0.00	258.00 ± 1.00	363.70 ± 0.58	180.70 ± 0.58	122.70 ± 0.58	94.00 ± 0.00	64.67 ± 1.16
roman-empire	240.30 ± 0.58	294.30 ± 0.58	93.00 ± 0.00	38.67 ± 2.89	23.33 ± 0.58	18.00 ± 0.00	14.67 ± 2.89
squirrel	29.00 ± 0.00	87.10 ± 0.32	24.70 ± 1.89	43.30 ± 0.48	25.20 ± 0.42	18.00 ± 0.00	11.70 ± 0.48
wikics	60.00 ± 3.46	97.33 ± 0.58	27.33 ± 0.58	10.00 ± 0.00	7.33 ± 0.58	7.00 ± 0.00	7.00 ± 0.00

In general, MLPs are more efficient than MPGNNs, as backpropagation over message-passing is costly. However, we increase the number of features in the data—depending on the aggregation depth and number of reducers—so for some datasets with many features the improvement is not necessarily observed. Moreover, this can lead to memory constraints. A way to reduce this overhead is to apply dimensionality reduction to the tabular FAF representation.

Applying PCA before aggregation discards information from the original node features, which is undesirable. Post-aggregation reduction is more natural because many aggregated columns can be redundant. That said, PCA mixes raw features into dense linear combinations, which can make optimization harder when the task signal is sparse, e.g., concentrated in interactions between a few specific features. As an alternative, in Table 4 we drop columns that lie in the span of others while leaving the original features untouched. This preserves performance, and the runtime gain is approximately proportional to the number of dropped features.

Table 4. Comparison of FAF_4 against post-dropping columns in the same span than previous ones, and to PCA.

Dataset	Cora			Citeseer			Pubmed			Coauthor-CS			Amazon-Photo		
Model	FAF_4	Drop-Span	PCA	FAF_4	Drop-Span	PCA	FAF_4	Drop-Span	PCA	FAF_4	Drop-Span	PCA	FAF_4	Drop-Span	PCA
Test perf.	0.818	0.8226	0.6632	0.6768	0.6846	0.471	0.772	0.7688	0.627	0.9541	0.95	0.9203	0.9651	0.9612	0.9388
OG. dim.	1433	1433	1433	3703	3703	3703	500	500	500	6805	6805	6805	745	745	745
MLP dim.	18629	2708	2708	33327	3327	3327	8500	7364	8500	88465	18333	18333	18625	7650	7650

B.4. FAFs Beyond Transductive Node Classification

Our theory applies to any task that learns multiset functions over neighborhoods. In our experiments, we focus on node classification for two main reasons. First, these are the benchmarks on which GNNs have been shown to be competitive with more complex architectures in Luo et al. (2024), so they are amenable to simple models for which we have strong, well-tuned hyperparameters. Second, node-classification datasets typically provide rich features that depend on neighborhood distributions. Thus, non-injective but commonly used reducers such as mean and sum still convey highly informative distributional signals. Regarding inductive settings, they would require computing the new aggregation rounds at test time. We would not have access to the test node features when precomputing training aggregations. Otherwise, our approach is just as feasible as in the transductive case.

Table 5. Best hyperparameters for FAF. Classic GNNs are taken from (Luo et al., 2024).

Dataset	Model	dropout	lr	bn	ln	hidden channels	wd	hops	mlp layers	res
computer	GCN	0.5	0.001	0	1	512	5e-05	3	0	0
	GAT	0.5	0.001	0	1	64	5e-05	2	0	0
	SAGE	0.3	0.001	0	1	64	5e-05	4	0	0
	FAF	0.7	0.005	1	0	256	5e-05	2	2	0
photo	GCN	0.5	0.001	0	1	256	5e-05	6	0	1
	GAT	0.5	0.001	0	1	64	5e-05	3	0	1
	SAGE	0.2	0.001	0	1	64	5e-05	6	0	1
	FAF	0.5	0.005	1	0	256	0.0005	4	2	0
ratings	GCN	0.5	0.001	1	0	512	0	4	0	1
	GAT	0.5	0.001	1	0	512	0	4	0	1
	SAGE	0.5	0.001	1	0	512	0	9	0	1
	FAF	0.2	0.001	1	0	256	0	3	2	0
chameleon	GCN	0.2	0.005	0	0	512	0.001	5	0	0
	GAT	0.7	0.01	1	0	256	0.001	2	0	1
	SAGE	0.7	0.01	1	0	256	0.001	4	0	1
	FAF	0.3	0.001	1	0	512	0.01	5	5	0
citeseer	GCN	0.5	0.001	0	0	512	0.01	2	0	0
	GAT	0.5	0.001	0	0	256	0.01	3	0	1
	SAGE	0.2	0.001	0	0	512	0.01	3	0	0
	FAF	0	0.005	0	1	512	0.001	2	3	0
coauthor-cs	GCN	0.3	0.001	0	1	512	0.0005	2	0	1
	GAT	0.3	0.001	0	1	256	0.0005	1	0	1
	SAGE	0.5	0.001	0	1	512	0.0005	2	0	1
	FAF	0.2	0.005	1	0	64	0.01	2	2	0
coauthor-physics	GCN	0.3	0.001	0	1	64	0.0005	2	0	1
	GAT	0.7	0.001	1	0	256	0.0005	2	0	1
	SAGE	0.7	0.001	1	0	64	0.0005	2	0	1
	FAF	0	0.001	1	0	512	0.001	1	2	0
cora	GCN	0.7	0.001	0	0	512	0.0005	3	0	0
	GAT	0.2	0.001	0	0	512	0.0005	3	0	1
	SAGE	0.7	0.001	0	0	256	0.0005	3	0	0
	FAF	0.7	0.01	0	1	512	0.01	3	3	0
minesweeper	GCN	0.2	0.01	1	0	64	0	12	0	1
	GAT	0.2	0.01	1	0	64	0	15	0	1
	SAGE	0.2	0.01	1	0	64	0	15	0	1
	FAF	0.2	0.01	1	0	64	0	4	12	0
pubmed	GCN	0.7	0.005	0	0	256	0.0005	2	0	0
	GAT	0.5	0.01	0	0	512	0.0005	2	0	0
	SAGE	0.7	0.005	0	0	512	0.0005	4	0	0
	FAF	0.7	0.01	0	1	64	0	4	2	0
questions	GCN	0.3	3e-05	0	0	512	0	10	0	1
	GAT	0.2	3e-05	0	1	512	0	3	0	1
	SAGE	0.2	3e-05	0	1	512	0	6	0	0
	FAF	0.2	0.005	1	0	512	0.01	4	3	0
roman-empire	GCN	0.5	0.001	1	0	512	0	9	0	1
	GAT	0.3	0.001	1	0	512	0	10	0	1
	SAGE	0.3	0.001	1	0	256	0	9	0	0
	FAF	0.7	0.01	1	0	256	0	2	3	0
squirrel	GCN	0.7	0.01	1	0	256	0.0005	4	0	1
	GAT	0.5	0.005	1	0	512	0.0005	7	0	1
	SAGE	0.7	0.01	1	0	256	0.0005	3	0	1
	FAF	0.7	0.01	1	0	512	0.01	4	5	0
wikics	GCN	0.5	0.001	0	1	256	0	3	0	0
	GAT	0.7	0.001	0	1	512	0	2	0	1
	SAGE	0.7	0.001	0	1	256	0	2	0	0
	FAF	0.7	0.01	1	0	64	0.001	2	2	0

C. Performance Details For the Main Experiments

C.1. Validation and Test Accuracy of Main Results

In Tables 6, 7 we report the validation and test accuracy of the main FAF variants of our experimental results (§ 5) where in Table 1 we only have the best validation FAF's test results. FAFs in all datasets are $\pm 1\%$ away from the best classic GNN, except for those already mentioned in the main text (Citeseer, Cora, Roman-Empire, and Minesweeper).

Table 6. Validation accuracy on node classification: all FAFs against classic GNNs. Test accuracy is shown in Table 7.

Dataset	computer	photo	ratings	chameleon	citeseer	coauthor-cs	coauthor-physics
GCN	92.58 ± 0.10	95.42 ± 0.11	54.01 ± 0.23	48.15 ± 2.35	**70.36 ± 0.09**	95.32 ± 0.07	**97.16 ± 0.07**
GAT	92.86 ± 0.06	95.93 ± 0.15	55.56 ± 0.68	46.97 ± 2.07	69.52 ± 0.27	95.30 ± 0.08	97.11 ± 0.03
SAGE	92.33 ± 0.17	95.60 ± 0.16	**55.90 ± 0.54**	46.22 ± 2.12	68.48 ± 1.05	**95.51 ± 0.04**	97.02 ± 0.09
MLP	87.89 ± 0.13	93.33 ± 0.07	48.98 ± 0.72	41.43 ± 1.77	56.80 ± 1.24	93.70 ± 0.07	95.89 ± 0.02
FAF_4	93.05 ± 0.04	**96.34 ± 0.07**	55.53 ± 0.43	**48.51 ± 2.31**	67.28 ± 0.64	94.93 ± 0.07	96.83 ± 0.01
$FAF_{mean,std}$	93.04 ± 0.13	96.23 ± 0.08	55.11 ± 0.40	48.42 ± 1.64	67.20 ± 0.28	94.94 ± 0.07	96.84 ± 0.03
FAF_{mean}	**93.16 ± 0.04**	96.06 ± 0.10	53.78 ± 0.52	47.99 ± 2.02	66.92 ± 0.87	95.20 ± 0.14	97.00 ± 0.04
$FAF_{max,std}$	92.32 ± 0.08	95.80 ± 0.04	55.70 ± 0.45	48.42 ± 2.14	66.64 ± 0.54	95.04 ± 0.04	96.56 ± 0.03
FAF_{max}	91.93 ± 0.04	95.60 ± 0.04	55.63 ± 0.29	48.06 ± 2.30	66.56 ± 0.50	95.19 ± 0.13	96.54 ± 0.01
FAF_{sum}	90.95 ± 0.04	94.88 ± 0.04	53.48 ± 0.59	47.29 ± 1.92	67.84 ± 1.45	95.13 ± 0.09	96.65 ± 0.05
FAF_{std}	92.50 ± 0.06	95.86 ± 0.10	55.31 ± 0.32	47.27 ± 2.15	63.44 ± 0.17	95.01 ± 0.12	96.76 ± 0.02

Dataset	cora	minesweeper	pubmed	questions	roman-empire	squirrel	wikics
GCN	81.28 ± 0.33	97.36 ± 0.46	79.08 ± 0.23	78.63 ± 0.23	**91.14 ± 0.58**	44.88 ± 1.27	81.52 ± 0.37
GAT	81.16 ± 0.52	97.08 ± 1.16	78.84 ± 0.52	78.12 ± 1.03	90.49 ± 0.68	43.30 ± 1.43	**82.38 ± 0.57**
SAGE	81.32 ± 0.41	**97.68 ± 0.63**	78.88 ± 0.91	77.35 ± 1.09	90.44 ± 0.66	40.58 ± 1.17	82.27 ± 0.38
MLP	62.68 ± 1.15	51.12 ± 0.93	71.12 ± 0.52	71.58 ± 1.46	66.28 ± 0.27	40.57 ± 0.92	74.86 ± 0.33
FAF_4	82.84 ± 0.43	89.63 ± 1.03	79.08 ± 0.36	**79.53 ± 1.12**	78.68 ± 0.19	47.31 ± 1.39	81.92 ± 0.43
$FAF_{mean,std}$	**83.36 ± 0.17**	89.18 ± 0.71	**81.28 ± 0.30**	77.32 ± 0.36	77.59 ± 0.41	47.30 ± 1.32	81.37 ± 0.51
FAF_{mean}	83.28 ± 0.30	89.89 ± 0.93	81.16 ± 0.97	78.53 ± 0.87	76.67 ± 0.36	46.29 ± 1.50	81.58 ± 0.46
FAF_{max}	81.80 ± 0.42	86.08 ± 0.77	77.48 ± 0.30	79.15 ± 0.86	75.06 ± 0.14	46.47 ± 1.38	80.30 ± 0.56
$FAF_{max,std}$	82.08 ± 0.33	87.83 ± 0.63	78.28 ± 0.30	78.86 ± 0.89	76.19 ± 0.26	**47.44 ± 1.51**	80.46 ± 0.53
FAF_{sum}	82.60 ± 0.65	89.86 ± 0.85	79.40 ± 0.57	78.12 ± 0.27	77.13 ± 0.23	46.85 ± 1.28	78.17 ± 0.23
FAF_{std}	81.40 ± 0.51	88.20 ± 0.52	80.00 ± 0.40	76.25 ± 0.53	73.95 ± 0.49	45.91 ± 1.32	77.65 ± 0.31

C.2. Training, Validation, and Test Accuracy Curves

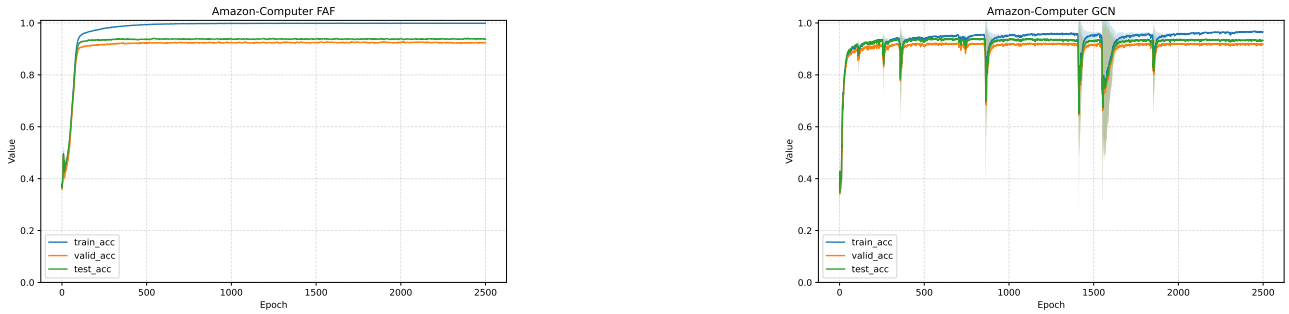
In this section we compare training MLPs on FAF features to training GCNs, by tracking train/validation/test accuracy over epochs (Figure 5). Below we summarize the behaviors on datasets where differences arise between the two methods:

- Amazon-Computer (5a) and Amazon-Photo (5b) behave similarly, but GCNs are more unstable.
- FAF for Chameleon (5d) has much better training accuracy but similar generalization; in contrast, GCN for Squirrel (5m) has much better training, but slightly worse generalization than FAF.
- Citeseer (5e) with FAF breaks at the end of training, which indicates instability. However, this could be overcome with standard learning rate schedules.
- Coauthor-CS (5f) and Coauthor-Physics (5g) have dips in all metrics for both models.
- Questions (5k) with FAF is more (locally) unstable but also more stationary and does not degrade performance later on.
- As mentioned in the main text, Minesweeper (5i) and Roman-Empire (5l) are the two datasets that seem to truly lose neighborhood information with FAF.

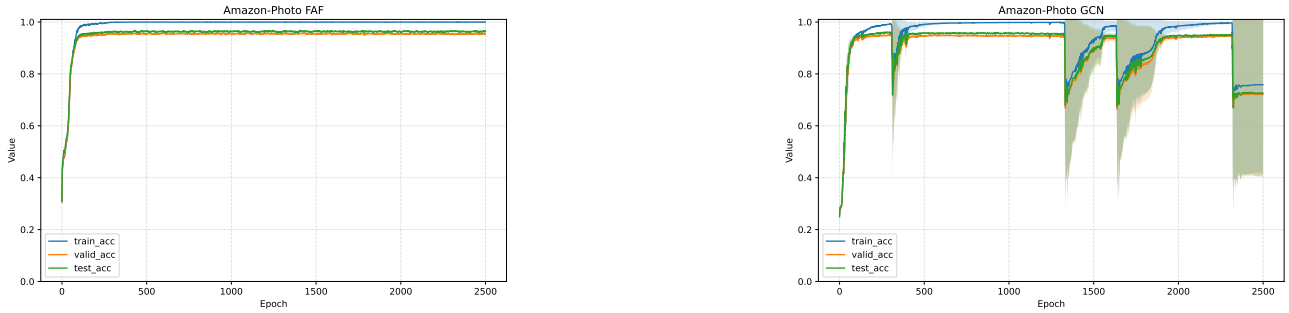
Table 7. Test accuracy on node classification: all FAFs against classic GNNs. Validation accuracy is shown in Table 6.

Dataset	computer	photo	ratings	chameleon	citeseer	coauthor-cs	coauthor-physics
GCN	93.58 ± 0.44	95.77 ± 0.27	53.86 ± 0.48	44.62 ± 4.50	**72.72 ± 0.45**	95.73 ± 0.15	**97.47 ± 0.08**
GAT	93.91 ± 0.22	96.45 ± 0.37	**55.51 ± 0.55**	42.90 ± 5.47	71.82 ± 0.65	96.14 ± 0.08	97.12 ± 0.13
SAGE	93.31 ± 0.17	96.17 ± 0.44	55.26 ± 0.27	43.11 ± 4.73	71.82 ± 0.81	**96.21 ± 0.10**	97.10 ± 0.09
MLP	87.75 ± 0.42	93.62 ± 0.36	49.04 ± 0.39	38.59 ± 3.29	57.22 ± 2.25	93.80 ± 0.19	96.02 ± 0.16
$FAF_{bestval}$	**94.01 ± 0.21**	96.54 ± 0.13	55.09 ± 0.24	42.96 ± 2.45	70.48 ± 1.24	95.37 ± 0.17	97.05 ± 0.18
FAF_4	93.75 ± 0.04	96.54 ± 0.13	54.42 ± 0.45	42.96 ± 2.45	69.42 ± 1.32	95.33 ± 0.20	96.96 ± 0.09
$FAF_{mean,std}$	94.00 ± 0.25	96.30 ± 0.23	54.73 ± 0.22	**45.13 ± 3.42**	67.90 ± 0.95	95.34 ± 0.14	96.93 ± 0.04
FAF_{mean}	**94.01 ± 0.21**	**96.71 ± 0.16**	53.12 ± 0.44	43.21 ± 2.24	66.82 ± 1.74	95.37 ± 0.17	97.05 ± 0.18
$FAF_{max,std}$	93.60 ± 0.25	96.01 ± 0.41	55.09 ± 0.24	43.20 ± 2.42	67.18 ± 0.88	95.53 ± 0.10	96.61 ± 0.04
FAF_{max}	92.98 ± 0.22	96.12 ± 0.10	54.79 ± 0.15	42.15 ± 3.19	67.52 ± 0.40	95.55 ± 0.08	96.84 ± 0.13
FAF_{sum}	91.77 ± 0.24	95.08 ± 0.61	53.44 ± 0.18	39.63 ± 2.90	70.48 ± 1.24	95.08 ± 0.12	96.86 ± 0.06
FAF_{std}	93.54 ± 0.26	96.17 ± 0.10	54.77 ± 0.14	42.68 ± 2.75	62.70 ± 1.18	95.77 ± 0.12	96.97 ± 0.09

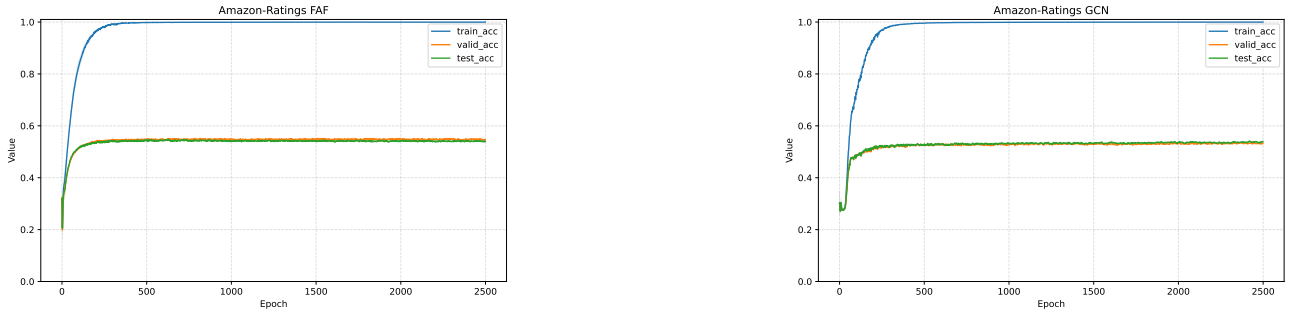
Dataset	cora	minesweeper	pubmed	questions	roman-empire	squirrel	wikics
GCN	**84.38 ± 0.81**	97.48 ± 0.06	80.00 ± 0.77	78.44 ± 0.23	**91.05 ± 0.15**	44.26 ± 1.22	80.06 ± 0.81
GAT	83.02 ± 1.21	97.00 ± 1.02	79.80 ± 0.94	77.72 ± 0.71	90.38 ± 0.49	39.31 ± 2.42	**81.01 ± 0.23**
SAGE	83.18 ± 0.93	**97.72 ± 0.70**	77.42 ± 0.40	76.75 ± 1.07	90.41 ± 0.10	40.22 ± 1.47	80.57 ± 0.42
MLP	58.56 ± 1.75	51.74 ± 0.83	68.22 ± 0.96	70.40 ± 1.17	66.43 ± 0.12	39.11 ± 1.93	72.98 ± 0.49
$FAF_{bestval}$	82.84 ± 0.63	90.00 ± 0.39	**80.96 ± 1.06**	**78.69 ± 0.50**	78.11 ± 0.38	**44.59 ± 1.62**	80.25 ± 0.34
FAF_4	81.44 ± 0.38	90.01 ± 0.51	77.20 ± 0.45	**78.69 ± 0.50**	78.11 ± 0.38	44.02 ± 2.08	80.25 ± 0.34
$FAF_{mean,std}$	82.84 ± 0.63	90.17 ± 0.51	**80.96 ± 1.06**	75.82 ± 1.27	77.14 ± 0.52	43.83 ± 2.34	79.48 ± 0.81
FAF_{mean}	82.80 ± 0.70	90.00 ± 0.39	79.88 ± 0.92	76.83 ± 1.19	76.36 ± 0.55	42.44 ± 1.73	79.61 ± 0.56
$FAF_{max,std}$	79.34 ± 0.95	88.36 ± 0.74	77.52 ± 0.77	76.62 ± 0.79	75.89 ± 0.30	**44.59 ± 1.62**	78.44 ± 0.67
FAF_{max}	79.34 ± 0.67	86.39 ± 1.22	77.18 ± 0.13	77.59 ± 1.67	75.01 ± 0.43	43.03 ± 1.90	78.63 ± 0.35
FAF_{sum}	81.46 ± 0.62	89.96 ± 0.45	77.46 ± 0.43	76.12 ± 1.08	76.90 ± 0.28	44.07 ± 1.98	76.59 ± 0.36
FAF_{std}	79.50 ± 0.39	88.93 ± 0.68	79.06 ± 1.09	73.99 ± 1.67	73.80 ± 0.21	43.63 ± 1.43	76.09 ± 0.26



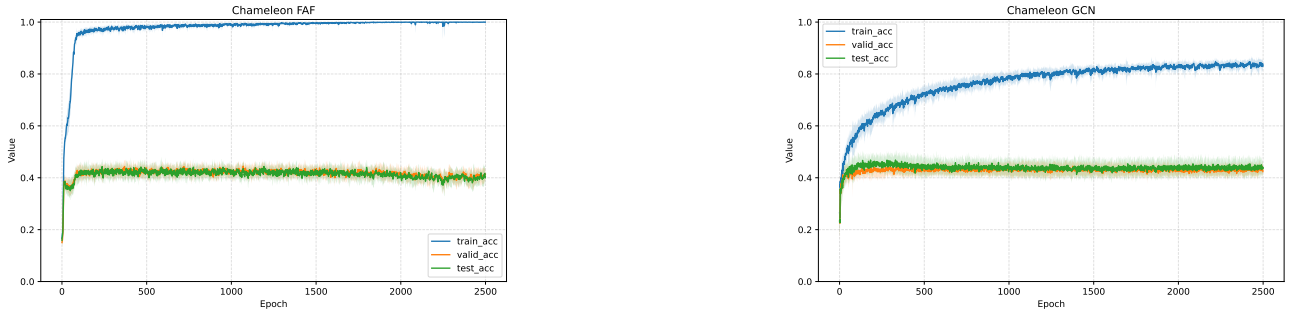
(a) Amazon-Computer: FAF vs. GCN



(b) Amazon-Photo: FAF vs. GCN

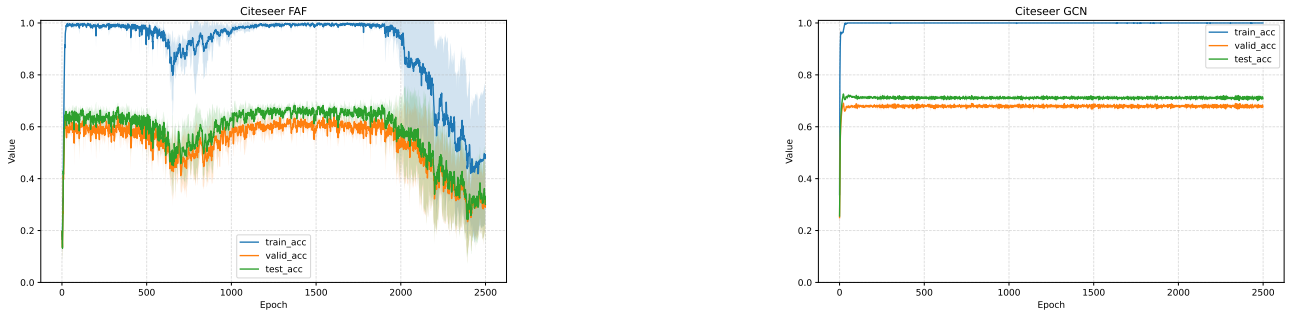


(c) Amazon-Ratings: FAF vs. GCN

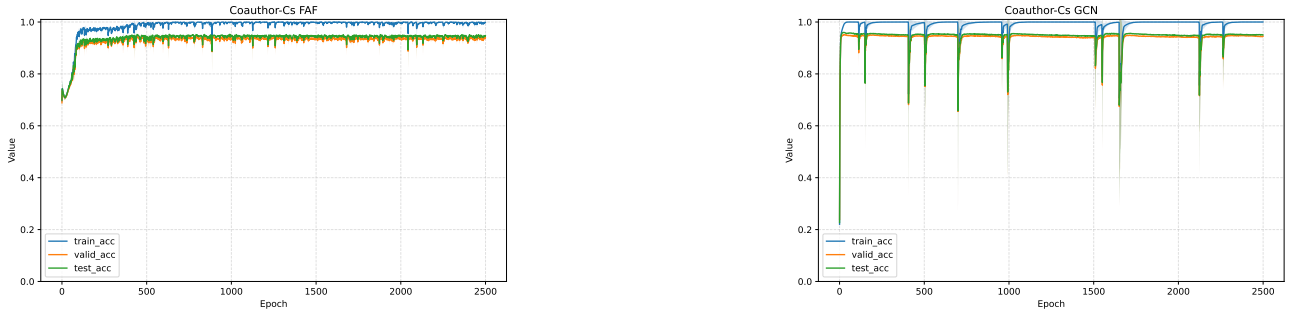


(d) Chameleon: FAF vs. GCN

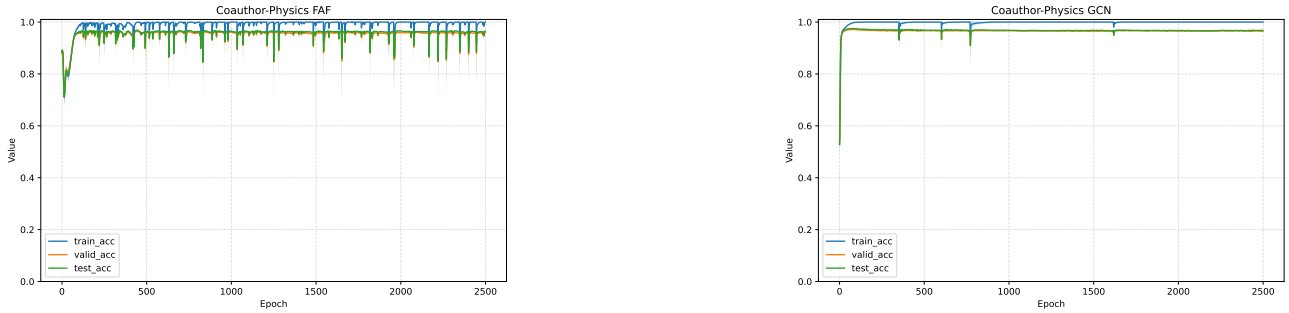
Figure 5. Train, validation, and test accuracy of FAF+MLP versus GCN. (i)



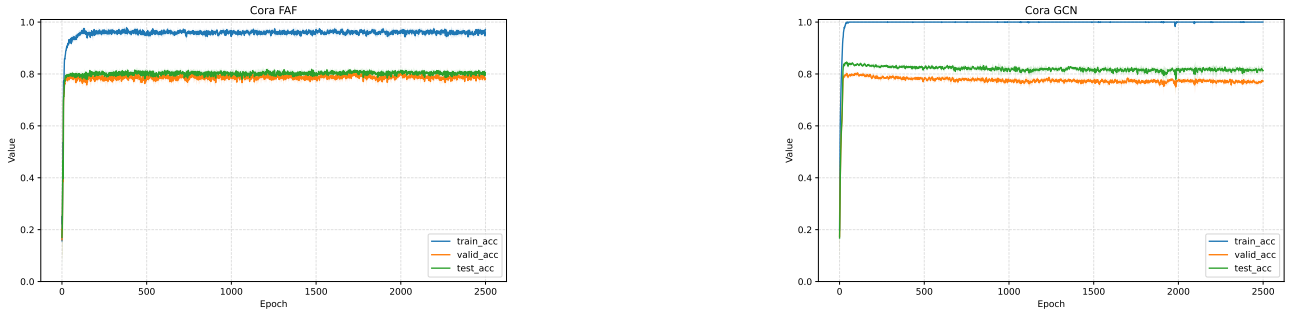
(e) Citeseer: FAF vs. GCN



(f) Coauthor-CS: FAF vs. GCN

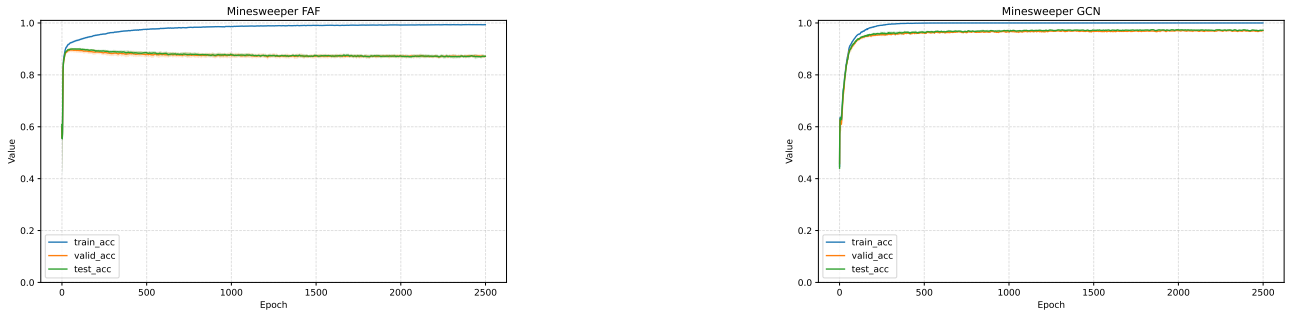


(g) Coauthor-Physics: FAF vs. GCN

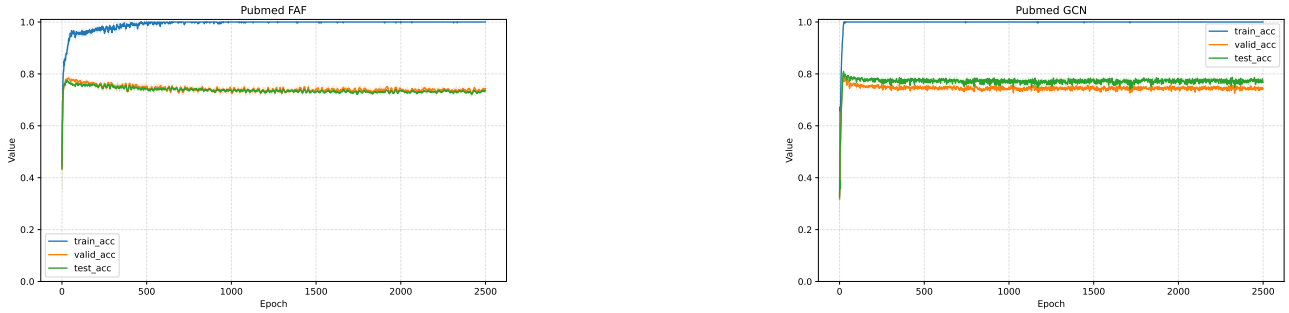


(h) Cora: FAF vs. GCN

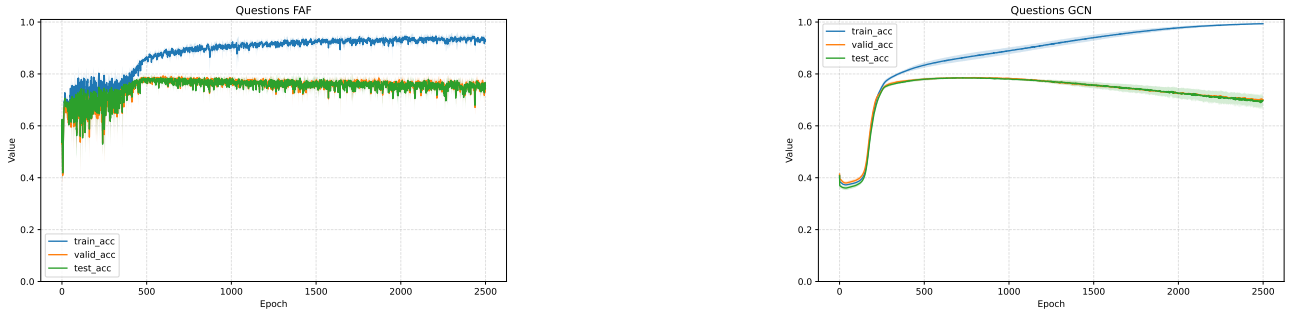
Figure 5. Train, validation, and test accuracy of FAF+MLP versus GCN. (ii)



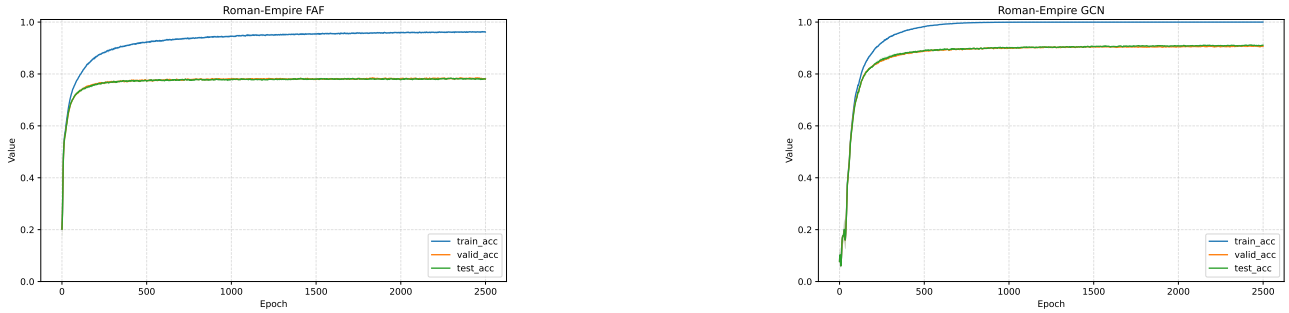
(i) Minesweeper: FAF vs. GCN



(j) Pubmed: FAF vs. GCN

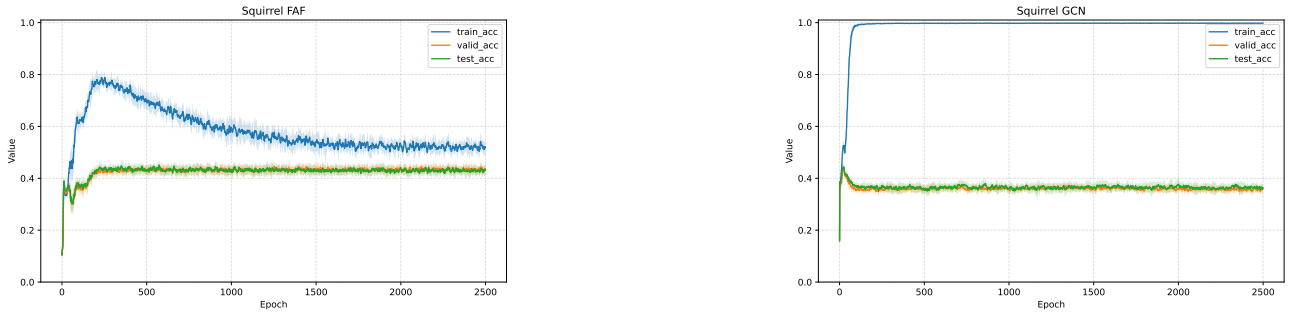


(k) Questions: FAF vs. GCN

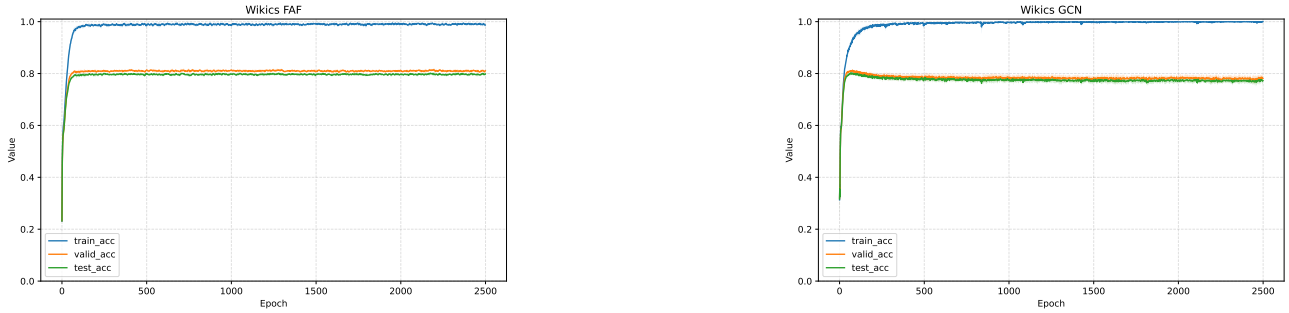


(l) Roman-Empire: FAF vs. GCN

Figure 5. Train, validation, and test accuracy of FAF+MLP versus GCN. (iii)



(m) Squirrel: FAF vs. GCN



(n) Wikics: FAF vs. GCN

Figure 5. Train, validation, and test accuracy of FAF+MLP versus GCN. (iv)

D. Additional Experiments and Ablations

Number of hops. In Tables 8 and 9 we include the test and validation for different number of hops concatenated as features for FAF_4. One hop already gives much of the information, and two hops often give the best performance.

Table 8. Test accuracy for increasing number of concatenated hops in FAF_4.

Dataset	computer	photo	ratings	chameleon	citeseer	coauthor-cs	coauthor-physics
FAF+0	87.75 ± 0.42	93.62 ± 0.36	49.04 ± 0.39	38.59 ± 3.29	52.98 ± 2.32	93.80 ± 0.19	96.02 ± 0.16
FAF+1	93.41 ± 0.11	95.88 ± 0.17	53.61 ± 0.07	**42.42 ± 3.44**	65.80 ± 1.15	95.11 ± 0.18	**96.99 ± 0.11**
FAF+2	93.76 ± 0.11	96.21 ± 0.36	53.88 ± 0.20	42.31 ± 3.02	**69.40 ± 0.56**	**95.37 ± 0.33**	96.86 ± 0.13
FAF+4	93.93 ± 0.07	**96.51 ± 0.14**	54.67 ± 0.11	42.25 ± 2.33	51.48 ± 4.42	95.31 ± 0.10	96.86 ± 0.16
FAF+8	**94.12 ± 0.09**	96.47 ± 0.26	**54.81 ± 0.45**	42.08 ± 1.90	42.12 ± 2.96	95.17 ± 0.23	OOM

Dataset	cora	minesweeper	pubmed	questions	roman-empire	squirrel	wikics
FAF+0	58.56 ± 1.75	51.74 ± 0.83	68.22 ± 0.96	70.40 ± 1.17	66.43 ± 0.12	39.11 ± 1.93	72.98 ± 0.49
FAF+1	81.12 ± 1.14	88.15 ± 0.82	76.54 ± 0.21	76.00 ± 1.23	77.39 ± 0.19	43.71 ± 2.25	79.69 ± 0.33
FAF+2	**82.30 ± 0.31**	89.76 ± 0.47	**78.38 ± 0.56**	78.43 ± 0.82	**78.16 ± 0.30**	44.22 ± 2.13	80.07 ± 0.40
FAF+4	79.24 ± 0.52	**90.01 ± 0.50**	77.20 ± 0.45	**78.54 ± 1.51**	76.65 ± 0.41	**45.09 ± 2.21**	**80.18 ± 0.59**
FAF+8	71.50 ± 0.82	89.21 ± 0.75	74.50 ± 0.34	77.85 ± 2.16	74.34 ± 0.92	44.17 ± 2.03	79.88 ± 0.70

Linear vs. MLP classifier. In Table 10 we show the performance of two classifiers on the same FAF_4 features: one linear layer and a multilayer perceptron—this being our choice for other experiments.

Table 9. Validation accuracy for increasing number of concatenated hops in FAF_4.

Dataset	computer	photo	ratings	chameleon	citeseer	coauthor-cs	coauthor-physics
FAF+0	87.89 ± 0.13	93.33 ± 0.07	48.98 ± 0.72	41.43 ± 1.77	53.80 ± 0.82	93.70 ± 0.07	95.89 ± 0.02
FAF+1	92.53 ± 0.08	96.14 ± 0.07	54.17 ± 0.14	46.91 ± 1.43	65.52 ± 0.64	94.84 ± 0.08	**96.83 ± 0.01**
FAF+2	92.99 ± 0.08	96.23 ± 0.08	55.02 ± 0.67	47.30 ± 1.76	**67.28 ± 0.64**	**94.93 ± 0.07**	96.63 ± 0.02
FAF+4	**93.04 ± 0.10**	**96.23 ± 0.04**	55.08 ± 0.19	48.42 ± 2.22	50.52 ± 3.84	94.88 ± 0.07	96.63 ± 0.04
FAF+8	92.96 ± 0.13	96.21 ± 0.07	**55.26 ± 0.30**	**48.74 ± 1.69**	40.72 ± 2.26	94.92 ± 0.09	OOM

Dataset	cora	minesweeper	pubmed	questions	roman-empire	squirrel	wikics
FAF+0	62.68 ± 1.15	51.12 ± 0.93	71.12 ± 0.52	71.58 ± 1.46	66.28 ± 0.27	40.57 ± 0.92	74.86 ± 0.33
FAF+1	82.16 ± 0.33	87.65 ± 0.47	78.24 ± 0.36	77.44 ± 1.07	77.36 ± 0.55	47.26 ± 1.31	81.37 ± 0.51
FAF+2	**82.84 ± 0.38**	89.48 ± 1.08	78.52 ± 0.41	79.71 ± 0.86	**78.66 ± 0.19**	47.18 ± 1.44	**81.81 ± 0.46**
FAF+4	81.80 ± 0.35	**89.63 ± 1.02**	**79.08 ± 0.36**	79.67 ± 0.89	77.48 ± 0.17	47.61 ± 1.43	81.73 ± 0.53
FAF+8	74.28 ± 0.23	89.10 ± 1.03	76.80 ± 0.20	**79.94 ± 0.88**	75.08 ± 0.23	**47.95 ± 1.36**	81.58 ± 0.63

Table 10. Comparison of 1 linear layer (1L) versus multiple layers (MLP) as the classifier over FAF_4.

Dataset	computer	photo	ratings	chameleon	citeseer	coauthor-cs	coauthor-physics
FAF+MLP	93.75 ± 0.04	96.54 ± 0.13	54.42 ± 0.45	42.96 ± 2.45	69.42 ± 1.32	95.33 ± 0.20	96.96 ± 0.09
FAF+1L	92.78 ± 0.06	96.19 ± 0.10	47.80 ± 0.23	43.97 ± 3.18	67.80 ± 1.22	94.07 ± 0.06	96.82 ± 0.11

Dataset	cora	minesweeper	pubmed	questions	roman-empire	squirrel	wikics
FAF+MLP	81.44 ± 0.38	90.01 ± 0.51	77.20 ± 0.45	78.69 ± 0.50	78.11 ± 0.38	44.02 ± 2.08	80.25 ± 0.34
FAF+1L	81.10 ± 0.91	89.45 ± 0.65	77.00 ± 0.60	77.87 ± 1.05	74.62 ± 0.03	44.14 ± 2.90	79.77 ± 0.96

Last hop. In Table 11 we ablate on using only the last hop as features to an MLP, or using the last hop concatenated to the original features. This mimics the choice of directly freezing a GNN and using its output as features to an (often linear) classifier. In almost all cases, having all hops is more performant, despite the increase in input size.

Table 11. Using only the last hop (H^K), that and original features ($H^{0\oplus K}$), and all hops (FAF_4).

Dataset	computer	photo	ratings	chameleon	citeseer	coauthor-cs	coauthor-physics
FAF_4	93.75 ± 0.04	96.54 ± 0.13	54.42 ± 0.45	42.96 ± 2.45	69.42 ± 1.32	95.33 ± 0.20	96.96 ± 0.09
$H^{0\oplus K}$	93.13 ± 0.15	96.45 ± 0.04	54.39 ± 0.56	40.33 ± 4.49	68.48 ± 1.04	95.43 ± 0.33	96.82 ± 0.13
H^K	92.68 ± 0.11	92.42 ± 0.11	48.87 ± 0.36	44.28 ± 2.63	68.64 ± 1.24	93.08 ± 0.23	96.33 ± 0.11

Dataset	cora	minesweeper	pubmed	questions	roman-empire	squirrel	wikics
FAF_4	81.44 ± 0.38	90.01 ± 0.51	77.20 ± 0.45	78.69 ± 0.50	78.11 ± 0.38	44.02 ± 2.08	80.25 ± 0.34
$H^{0\oplus K}$	81.38 ± 0.59	73.28 ± 0.94	77.04 ± 0.27	77.80 ± 0.92	75.78 ± 0.05	43.07 ± 2.13	79.24 ± 0.40
H^K	81.20 ± 0.97	69.96 ± 1.72	77.00 ± 0.64	77.66 ± 1.44	54.11 ± 0.42	42.15 ± 1.64	77.77 ± 0.77

KA reducer. Table 12 shows the last epoch training accuracy, best epoch validation accuracy, and corresponding test accuracy when using the Kolmogorov-Arnold function Φ as a reducer for FAF. Following Corso et al. (2020), we make it act on multisets by sorting, which we fix by the given data order.

Rewiring augmentations. In Table 13 we include results on rewiring the input graph by deleting edges based on pairwise cosine similarity, as mentioned in § 3.1. We use mean as reducer for the original and the rewired aggregation. Coauthor-Physics is not included, as it has a large amount of original features. REW includes hop-wise features where all negative similarity neighbors are set to 0. SP includes hop-wise features where positive and negative similarity neighbors are aggregated in different features and concatenated together. Rewiring methods can provide extra signal for the tasks and mitigate fundamental issues like over-squashing and over-smoothing, and are combinable with FAFs.

Table 12. Last epoch training accuracy, best epoch validation accuracy, and corresponding test accuracy of the KA function Φ from Theorem 4.2.

Dataset	computer	photo	ratings	chameleon	citeseer	coauthor-cs	coauthor-physics
$\mathrm{FAF}_{\mathrm{KA}}$ (train)	94.73 ± 0.12	99.65 ± 0.09	99.92 ± 0.02	96.35 ± 3.44	70.17 ± 41.23	99.49 ± 0.22	99.84 ± 0.28
$\mathrm{FAF}_{\mathrm{KA}}$ (val)	87.88 ± 0.19	93.40 ± 0.07	51.97 ± 0.12	41.65 ± 1.91	55.76 ± 1.07	93.66 ± 0.08	95.90 ± 0.05
$\mathrm{FAF}_{\mathrm{KA}}$ (test)	87.56 ± 0.42	93.73 ± 0.07	51.46 ± 0.50	35.96 ± 3.89	56.18 ± 1.50	93.70 ± 0.08	95.97 ± 0.00

Dataset	cora	minesweeper	pubmed	questions	roman-empire	squirrel	wikics
$\mathrm{FAF}_{\mathrm{KA}}$ (train)	14.29 ± 0.00	51.68 ± 0.44	33.33 ± 0.00	99.85 ± 0.09	86.41 ± 0.23	39.30 ± 2.30	97.64 ± 1.09
$\mathrm{FAF}_{\mathrm{KA}}$ (val)	29.56 ± 0.79	51.61 ± 0.42	42.80 ± 0.76	74.28 ± 1.95	80.45 ± 0.25	40.43 ± 1.10	77.38 ± 0.88
$\mathrm{FAF}_{\mathrm{KA}}$ (test)	29.78 ± 0.91	52.21 ± 0.81	41.06 ± 3.63	71.82 ± 0.36	80.33 ± 0.47	38.62 ± 1.24	76.31 ± 0.77

Table 13. Feature augmentations based on similarity-based rewiring (REW) (Rubio-Madrigal et al., 2025) and computational-graph splitting (SP) (Roth et al., 2025). Comparison of test accuracy to $\mathrm{FAF}_{\mathrm{mean}}$ and to their combination.

Dataset	computer	photo	ratings	chameleon	citeseer	coauthor-cs	cora
$\mathrm{FAF}_{\mathrm{mean}}$	94.01 ± 0.21	96.71 ± 0.16	53.12 ± 0.44	43.21 ± 2.24	66.82 ± 1.74	95.37 ± 0.17	**82.80 ± 0.70**
$\mathrm{REW}_{\mathrm{mean}}$	94.22 ± 0.15	96.58 ± 0.23	54.43 ± 0.12	38.54 ± 1.94	67.96 ± 0.82	95.27 ± 0.09	80.88 ± 0.80
$\mathrm{SP}_{\mathrm{mean}}$	94.10 ± 0.23	**96.75 ± 0.36**	**54.60 ± 0.38**	39.77 ± 3.20	67.38 ± 1.74	**95.46 ± 0.04**	80.38 ± 0.47
$\mathrm{FAF}_{\mathrm{mean}}$+$\mathrm{REW}_{\mathrm{mean}}$	94.16 ± 0.17	96.64 ± 0.16	53.48 ± 0.28	**44.31 ± 3.82**	67.58 ± 1.07	95.34 ± 0.21	82.68 ± 0.98
$\mathrm{FAF}_{\mathrm{mean}}$+$\mathrm{SP}_{\mathrm{mean}}$	**94.35 ± 0.04**	96.69 ± 0.04	53.48 ± 0.67	43.48 ± 3.71	**68.04 ± 1.30**	95.29 ± 0.16	82.38 ± 0.64

Dataset	minesweeper	pubmed	questions	roman-empire	squirrel	wikics
$\mathrm{FAF}_{\mathrm{mean}}$	**90.00 ± 0.39**	79.88 ± 0.92	76.83 ± 1.19	76.36 ± 0.55	42.44 ± 1.73	79.61 ± 0.56
$\mathrm{REW}_{\mathrm{mean}}$	59.57 ± 1.88	79.74 ± 0.59	75.65 ± 0.28	75.75 ± 0.42	38.24 ± 1.73	80.44 ± 0.37
$\mathrm{SP}_{\mathrm{mean}}$	59.44 ± 2.27	80.06 ± 0.63	75.83 ± 0.54	75.51 ± 0.16	38.40 ± 2.11	80.54 ± 0.37
$\mathrm{FAF}_{\mathrm{mean}}$+$\mathrm{REW}_{\mathrm{mean}}$	89.71 ± 0.31	**80.16 ± 0.42**	**77.15 ± 1.25**	**77.61 ± 0.54**	**43.25 ± 1.86**	**80.56 ± 0.52**
$\mathrm{FAF}_{\mathrm{mean}}$+$\mathrm{SP}_{\mathrm{mean}}$	89.71 ± 0.42	79.96 ± 0.71	76.58 ± 1.19	77.18 ± 0.26	42.54 ± 1.65	80.41 ± 0.67

SHAP on other datasets. In Figure 6 we show two more plots of feature importance using SHAP (Lundberg & Lee, 2017) for Pubmed and Amazon-Ratings, on the MLP over single-reducer FAFs. Features are sorted by global importance and broken down over the different hops by color. While the implementation of SHAP on MLPs used (GradientExplainer) relies on local linearization and often assumes input feature independence, the explanations still reveal informative qualitative patterns. In Pubmed, feature 346 is most important at hops 1 and 2, and remains second at hops 0 and 4, whereas the most important base feature (205) contributes little at other hops. By contrast, in Amazon-Ratings, importance is more evenly distributed across features and hops.

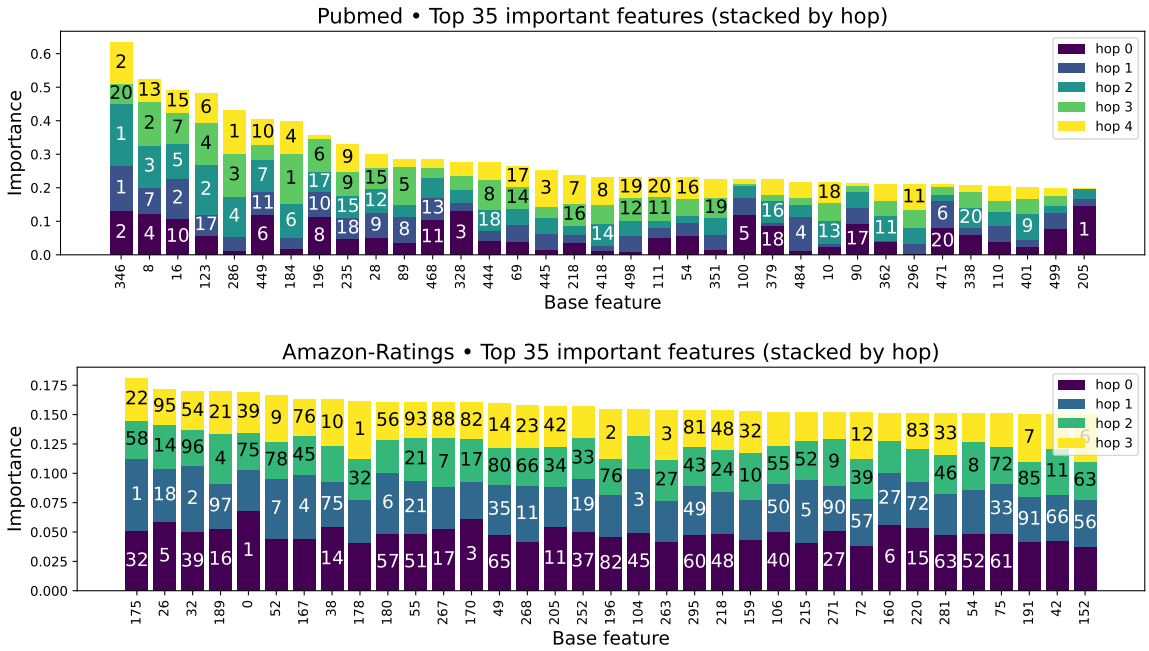


Figure 6. SHAP feature importance for Pubmed and Amazon-Ratings. The base features are ranked according to the sum of their importance values across hops. Numbers on the stacked bars indicate the ranking of that particular feature on that particular hop.

E. Comparison to Graph Echo State Networks

Graph Echo State Networks (GESN) (Gallicchio & Micheli, 2010) compute label-independent node embeddings via fixed "reservoir" layers, one per hop, followed by a linear readout. We compare FAFs to this approach in Table 14 as a representative previously proposed simplification of GNNs. We do not include Coauthor-Physics or Questions, as GESN exceeds memory capacity on them. We use the public implementation at https://github.com/dtortorella/graph-esn, keeping most GESN-specific hyperparameters as in their example. We set the "depth" (local layers) and "hidden units" (hidden channels) as in the best GCN. Each layer is given 10 minutes to compute its embedding. As the classifier, we replace the original linear readout with a MLP of the same architecture as FAFs to provide a fair comparison of the role of the embeddings. For the MLP hyperparameters, we try the best GCN and the best FAF configurations. Since our different FAFs could share MLP hyperparameters, we expected them to transfer well, but in this case the GCN ones perform better. As shown, FAFs seem to be more suitable for the tested benchmark tasks.

Table 14. Test accuracy of GESN (Gallicchio & Micheli, 2010) against FAFs.

Dataset	computer	photo	ratings	chameleon	citeseer	coauthor-cs
$\text{FAF}_{\text{bestval}}$	94.01 ± 0.21	96.54 ± 0.13	55.09 ± 0.24	42.96 ± 2.45	70.48 ± 1.24	95.37 ± 0.17
GESN+MLP	90.80 ± 0.10	92.72 ± 0.27	50.34 ± 0.26	41.64 ± 3.63	41.44 ± 0.34	89.99 ± 0.09

Dataset	cora	minesweeper	pubmed	roman-empire	squirrel	wikics
$\text{FAF}_{\text{bestval}}$	82.84 ± 0.63	90.00 ± 0.39	80.96 ± 1.06	78.11 ± 0.38	44.59 ± 1.62	80.25 ± 0.34
GESN+MLP	65.78 ± 0.26	50.93 ± 1.28	64.98 ± 1.57	11.76 ± 0.38	36.58 ± 1.18	73.98 ± 0.85

F. Preliminary Results on Other Benchmarks

We include preliminary results for FAFs on the GraphLand bechmark (Bazhenov et al., 2025). We do *not* perform the full hyperparameter sweep, therefore we indicate FAFs with an asterisk (*), as there could be better performing versions. We copy baselines from the original paper: MLP, MLP-NFA (one-hop FAFs), GCN and GAT. For FAFs, we only report results

for mean+std and mean aggregations. We choose the best validation hyperparameters we have been able to find so far (shown in Table 16) and report the resulting test accuracy in Table 17. While we do not yet achieve the performance of GATs, we approach that of GCNs, and we improve upon MLPs and NFA.

Table 15. Test accuracy of 4 GraphLand datasets (averaged over 10 runs).

Dataset	artnet-exp	hm-categories	tolokers-2	pokec-regions
ResMLP	35.07 ± 2.34	37.72 ± 0.18	41.16 ± 1.13	4.88 ± 0.01
ResMLP-NFA	38.25 ± 0.56	48.72 ± 0.38	48.14 ± 1.40	8.05 ± 0.03
GCN	43.09 ± 0.38	61.70 ± 0.35	51.32 ± 0.96	34.96 ± 0.38
GAT	46.62 ± 0.32	67.96 ± 0.33	53.78 ± 1.34	46.17 ± 0.32
$\text{FAF*}_{\text{mean,std}}$	41.56 ± 0.26	59.50 ± 0.15	52.74 ± 0.53	28.44 ± 0.22
$\text{FAF*}_{\text{mean}}$	39.25 ± 0.38	54.50 ± 0.13	50.72 ± 0.38	31.23 ± 0.18

Table 16. Best hyperparameters found so far for FAFs on GraphLand datasets.

	dropout	lr	bn	hidden channels	weight decay	local layers	mlp layers
artnet-exp	0.7	0.01	1	256	0.01	3	2
hm-categories	0.5	0.001	1	512	0.0005	3	3
tolokers-2	0.3	0.0001	1	512	5e-05	3	5
pokec-regions	0	0.001	1	512	0.001	3	3

We also include preliminary results on GraphBench (Stoll et al., 2026). We include preliminary results on their first two tasks (test MAE, lower is better). Their baselines are not tuned so neither do we, but we use the hyperparameters from GraphLand's tasks. Again, an asterisk (*) denotes there could be better FAFs. We show that $\text{FAF*}_{\text{mean}}$ performs better than the best reported model (MPGNN).

Table 17. Test MAE of 2 GraphBench datasets (lower is better).

Dataset	quotes	replies
GNN	0.768 ± 0.002	0.694 ± 0.002
$\text{FAF*}_{\text{mean}}$	0.661 ± 0.002	0.579 ± 0.001

Newer benchmarks have not yet been as widely studied as the benchmarks we focus on, and the limits of state-of-the-art models on them are not yet well understood. For instance, it is still unclear whether classic GNNs rival more complex architectures on these datasets. For this reason, we do not perform the same extensive hyperparameter tuning. A more thorough search could yield a better-performing FAF pipeline, so, for instance, we cannot definitely conclude that GraphLand requires learned aggregation; especially so, as we can already rival non-attention-based models. This is consistent with our broader finding that datasets requiring learned aggregation tend to be those that benefit from significantly deeper architectures (more than 10 layers), a pattern we have not observed in the well-studied shallow benchmarks.

We append these preliminary results primarily to motivate the community to include well-tuned FAF baselines alongside graph-agnostic baselines in future benchmark evaluations, as they increase performance considerably from MLPs. In terms of tuning, we note, for instance, that the hyperparameter tuning of GraphLand includes Dropout of up to 0.2, while for MLPs we have found larger values (up to 0.7) to perform better, as well as BatchNorm in exchange of LayerNorm. Our secondary message is that graph-agnostic baselines require well-chosen hyperparameter ranges in order to compete fairly, which they frequently achieve when fed with our well-constructed features.

G. Original results from (Luo et al., 2024)

For convenience, in Tables 18, 19 we present the original results tables from the benchmark we evaluate against (Luo et al., 2024). This shows that the classic GNNs also rival graph transformers and heterophily-aware models. The numbers we report may differ within error, as we re-run their baselines.

Table 18. (Luo et al., 2024) Node classification results over homophilous graphs (%). The top 1st, 2nd and 3rd results are highlighted.

	Cora	CiteSeer	PubMed	Computer	Photo	CS	Physics	WikiCS
# nodes	2,708	3,327	19,717	13,752	7,650	18,333	34,493	11,701
# edges	5,278	4,732	44,324	245,861	119,081	81,894	247,962	216,123
Metric	Accuracy↑	Accuracy↑	Accuracy↑	Accuracy↑	Accuracy↑	Accuracy↑	Accuracy↑	Accuracy↑
GraphGPS	82.84 ± 1.03	72.73 ± 1.23	79.94 ± 0.26	91.19 ± 0.54	95.06 ± 0.13	93.93 ± 0.12	97.12 ± 0.19	78.66 ± 0.49
GraphGPS*	83.87 ± 0.96	72.73 ± 1.23	79.94 ± 0.26	91.79 ± 0.63	94.89 ± 0.14	94.04 ± 0.21	96.71 ± 0.15	78.66 ± 0.49
NAGphormer	82.12 ± 1.18	71.47 ± 1.30	79.73 ± 0.28	91.22 ± 0.14	95.49 ± 0.11	95.75 ± 0.09	97.34 ± 0.03	77.16 ± 0.72
NAGphormer*	80.92 ± 1.17	70.59 ± 0.89	80.14 ± 1.06	91.69 ± 0.30	96.14 ± 0.16	95.85 ± 0.16	97.35 ± 0.12	77.92 ± 0.93
Exphormer	82.77 ± 1.38	71.63 ± 1.19	79.46 ± 0.35	91.47 ± 0.17	95.35 ± 0.22	94.93 ± 0.01	96.89 ± 0.09	78.54 ± 0.49
Exphormer*	83.29 ± 1.36	71.85 ± 1.11	79.67 ± 0.73	91.80 ± 0.35	95.69 ± 0.39	95.92 ± 0.25	97.06 ± 0.13	79.38 ± 0.62
GOAT	83.18 ± 1.27	71.99 ± 1.26	79.13 ± 0.38	90.96 ± 0.90	92.96 ± 1.48	94.21 ± 0.38	96.24 ± 0.24	77.00 ± 0.77
GOAT*	83.26 ± 1.24	72.21 ± 1.29	80.06 ± 0.67	92.29 ± 0.37	94.33 ± 0.21	93.81 ± 0.19	96.47 ± 0.16	77.96 ± 0.63
NodeFormer	82.20 ± 0.90	72.50 ± 1.10	79.90 ± 1.00	86.98 ± 0.62	93.46 ± 0.35	95.64 ± 0.22	96.45 ± 0.28	74.73 ± 0.94
NodeFormer*	82.73 ± 0.75	72.37 ± 1.20	79.59 ± 0.92	87.29 ± 0.58	93.43 ± 0.56	95.69 ± 0.27	96.48 ± 0.34	75.13 ± 0.93
SGFormer	84.50 ± 0.80	72.60 ± 0.20	80.30 ± 0.60	91.99 ± 0.76	95.10 ± 0.47	94.78 ± 0.20	96.60 ± 0.18	73.46 ± 0.56
SGFormer*	84.82 ± 0.85	72.72 ± 1.15	80.60 ± 0.49	92.42 ± 0.66	95.58 ± 0.36	95.71 ± 0.24	96.75 ± 0.26	80.05 ± 0.46
Polynormer	83.25 ± 0.93	72.31 ± 0.78	79.24 ± 0.43	93.68 ± 0.21	96.46 ± 0.26	95.53 ± 0.16	97.27 ± 0.08	80.10 ± 0.67
Polynormer*	83.43 ± 0.89	72.19 ± 0.83	79.35 ± 0.73	93.78 ± 0.10	96.57 ± 0.23	95.42 ± 0.19	97.18 ± 0.11	80.26 ± 0.92
GCN	81.60 ± 0.40	71.60 ± 0.40	78.80 ± 0.60	89.65 ± 0.52	92.70 ± 0.20	92.92 ± 0.12	96.18 ± 0.07	77.47 ± 0.85
GCN*	85.10 ± 0.67 **3.50↑**	73.14 ± 0.67 **1.54↑**	81.12 ± 0.52 **2.32↑**	93.99 ± 0.12 **4.34↑**	96.10 ± 0.46 **3.40↑**	96.17 ± 0.06 **3.25↑**	97.46 ± 0.10 **1.28↑**	80.30 ± 0.62 **2.83↑**
GraphSAGE	82.68 ± 0.47	71.93 ± 0.85	79.41 ± 0.53	91.20 ± 0.29	94.59 ± 0.14	93.91 ± 0.13	96.49 ± 0.06	74.77 ± 0.95
GraphSAGE*	83.88 ± 0.65 **1.20↑**	72.26 ± 0.55 **0.33↑**	79.72 ± 0.50 **0.31↑**	93.25 ± 0.14 **2.05↑**	96.78 ± 0.23 **2.19↑**	96.38 ± 0.11 **2.47↑**	97.19 ± 0.05 **0.70↑**	80.69 ± 0.31 **5.92↑**
GAT	83.00 ± 0.70	72.10 ± 1.10	79.00 ± 0.40	90.78 ± 0.13	93.87 ± 0.11	93.61 ± 0.14	96.17 ± 0.08	76.91 ± 0.82
GAT*	84.46 ± 0.55 **1.46↑**	72.22 ± 0.84 **0.12↑**	80.28 ± 0.64 **1.28↑**	94.09 ± 0.37 **3.31↑**	96.60 ± 0.33 **2.73↑**	96.21 ± 0.14 **2.60↑**	97.25 ± 0.06 **1.08↑**	81.07 ± 0.54 **4.16↑**

Table 19. (Luo et al., 2024) Node classification results on heterophilous graphs (%). The top 1st, 2nd and 3rd results are highlighted.

	Squirrel	Chameleon	Amazon-Ratings	Roman-Empire	Minesweeper	Questions
# nodes	2223	890	24,492	22,662	10,000	48,921
# edges	46,998	8,854	93,050	32,927	39,402	153,540
Metric	Accuracy↑	Accuracy↑	Accuracy↑	Accuracy↑	ROC-AUC↑	ROC-AUC↑
H2GCN	35.10 ± 1.15	26.75 ± 3.64	36.47 ± 0.23	60.11 ± 0.52	89.71 ± 0.31	63.59 ± 1.46
CPGNN	30.04 ± 2.03	33.00 ± 3.15	39.79 ± 0.77	63.96 ± 0.62	52.03 ± 5.46	65.96 ± 1.95
GPRGNN	38.95 ± 1.99	39.93 ± 3.30	44.88 ± 0.34	64.85 ± 0.27	86.24 ± 0.61	55.48 ± 0.91
FSGNN	35.92 ± 1.32	40.61 ± 2.97	52.74 ± 0.83	79.92 ± 0.56	90.08 ± 0.70	78.86 ± 0.92
GloGNN	35.11 ± 1.24	25.90 ± 3.58	36.89 ± 0.14	59.63 ± 0.69	51.08 ± 1.23	65.74 ± 1.19
GraphGPS	39.67 ± 2.84	40.79 ± 4.03	53.10 ± 0.42	82.00 ± 0.61	90.63 ± 0.67	71.73 ± 1.47
GraphGPS*	39.81 ± 2.28	41.55 ± 3.91	53.27 ± 0.66	82.72 ± 0.68	90.75 ± 0.89	72.56 ± 1.33
NodeFormer	38.52 ± 1.57	34.73 ± 4.14	43.86 ± 0.35	64.49 ± 0.73	86.71 ± 0.88	74.27 ± 1.46
NodeFormer*	38.89 ± 2.67	36.38 ± 3.85	43.79 ± 0.57	74.83 ± 0.81	87.71 ± 0.69	75.02 ± 1.61
SGFormer	41.80 ± 2.27	44.93 ± 3.91	48.01 ± 0.49	79.10 ± 0.32	90.89 ± 0.58	72.15 ± 1.31
SGFormer*	42.65 ± 2.41	45.21 ± 3.72	54.14 ± 0.62	80.01 ± 0.44	91.42 ± 0.41	73.81 ± 0.59
Polynormer	40.87 ± 1.96	41.82 ± 3.45	54.81 ± 0.49	92.55 ± 0.37	97.46 ± 0.36	78.92 ± 0.89
Polynormer*	41.97 ± 2.14	41.97 ± 3.18	54.96 ± 0.22	92.66 ± 0.60	97.49 ± 0.48	78.94 ± 0.78
GCN	38.67 ± 1.84	41.31 ± 3.05	48.70 ± 0.63	73.69 ± 0.74	89.75 ± 0.52	76.09 ± 1.27
GCN*	45.01 ± 1.63 **6.34↑**	46.29 ± 3.40 **4.98↑**	53.80 ± 0.60 **5.10↑**	91.27 ± 0.20 **17.58↑**	97.86 ± 0.24 **8.11↑**	79.02 ± 0.60 **2.93↑**
GraphSAGE	36.09 ± 1.99	37.77 ± 4.14	53.63 ± 0.39	85.74 ± 0.67	93.51 ± 0.57	76.44 ± 0.62
GraphSAGE*	40.78 ± 1.47 **4.69↑**	44.81 ± 4.74 **7.04↑**	55.40 ± 0.21 **1.77↑**	91.06 ± 0.27 **5.32↑**	97.77 ± 0.62 **4.26↑**	77.21 ± 1.28 **0.77↑**
GAT	35.62 ± 2.06	39.21 ± 3.08	52.70 ± 0.62	88.75 ± 0.41	93.91 ± 0.35	76.79 ± 0.71
GAT*	41.73 ± 2.07 **6.11↑**	44.13 ± 4.17 **4.92↑**	55.54 ± 0.51 **2.84↑**	90.63 ± 0.14 **1.88↑**	97.73 ± 0.73 **3.82↑**	77.95 ± 0.51 **1.16↑**