GRAPH NEURAL DIFFUSION AS THE PRIOR FOR IN-CONTEXT NODE CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Prior-fitted networks (PFNs) are pretrained on synthetic tasks sampled from a prior and make predictions on a new dataset in a single forward pass. Recently, PFN-based node classifiers have emerged as competitive models that classify the nodes of a new graph without training on it. For such models, the prior determines how the graph structure shapes the node features, and thus what the network can learn about graphs. However, the priors of existing PFN-based node classifiers either generate the features independently of the graph or only aggregate them over neighbours, and therefore produce a narrow range of the relations between features and graph structure found in real graphs. Graph neural diffusion views message passing as the integration of a diffusion equation on the graph, and subsequent work has proposed a variety of propagation schemes by adding terms such as reaction and advection. In this work, we propose GADRPFN, a PFN for node classification whose prior generates each task through graph neural diffusion. In each task, a node state on a random graph evolves under an advection–diffusion–reaction equation for a randomly drawn time, so that diffusion smooths the state along the edges and reaction sharpens it against them, and the node features are read from this state. GADRPFN is trained from scratch on about half a million synthetic tasks in two hours on a single GPU, whereas the strongest in-context node classifiers build on tabular foundation models pretrained on far more data. On 28 benchmarks, GADRPFN achieves the best mean accuracy and average rank on both homophilous and heterophilous graphs.

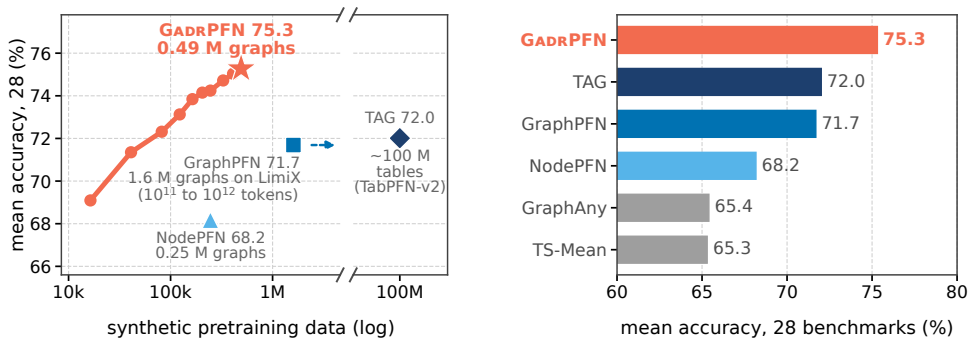
1 INTRODUCTION

Prior-fitted networks (PFNs) are pretrained on synthetic tasks sampled from a prior and make predictions on a new dataset in a single forward pass (Müller et al., 2022; 2025). They classify and regress small tables (Hollmann et al., 2023; 2025; Qu et al., 2025), forecast and classify time series (Dooley et al., 2023; Xie et al., 2026), and guide Bayesian optimisation (Müller et al., 2023). Since the network sees no real data during pretraining, its behaviour on a new dataset is determined by the tasks drawn from its prior, and new capabilities have often come from new priors. For example, TabPFN handles temporal drift once its prior drifts (Helli et al., 2024), and tabular and time-series foundation models improve when only their prior changes (Zhang et al., 2025b; Xie et al., 2026).

Recently, PFNs have been extended to classify the nodes of a new graph without training on it (Choi et al., 2026; Eremeev et al., 2026), as other in-context node classifiers do (Zhao et al., 2025; Finkelshtein et al., 2025; Hayler et al., 2025). The strongest of these build on tabular foundation models (Hollmann et al., 2025; Zhang et al., 2025a) and so rely on far more pretraining data than a graph PFN trained from scratch, which learns about graphs only from its prior (Figure 1a).

For node classification, the prior must specify how the graph shapes the node features and labels. Graph neural diffusion views message passing in graph neural networks as the integration of a diffusion equation on the graph (Chamberlain et al., 2021). The diffusion time determines how far a node's state moves, and the terms of the equation determine whether it moves towards or away from its neighbours (Han et al., 2024; Choi et al., 2023; Wang et al., 2023). This raises the question of how far, and in which direction, the prior of a graph PFN should let the graph act on node features.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Accuracy against pretraining data.

(b) Accuracy on the 28 benchmarks.

Figure 1: **GADRPFN trained from scratch against models built on pretrained tabular foundation models.** (a) Mean accuracy over the 28 benchmarks of Table 1 against the synthetic data seen in pretraining, with graphs left of the axis break and tables right of it. The arrow marks the LimiX checkpoint that GraphPFN continues from (Eremeev et al., 2026; Zhang et al., 2025a), and TAG runs TabPFN-v2 with no graph pretraining (Hayler et al., 2025; Hollmann et al., 2025). (b) The same means for all baselines of Table 1.

Existing priors limit how the graph acts on the features. NodePFN (Choi et al., 2026) generates the features independently of the graph and then samples the graph from the labels, so the edges do not affect its features. GraphPFN (Eremeev et al., 2026) lets a random share of the channels in each layer of its causal model aggregate over the neighbours. Apart from GraphPFN's degree and PageRank inputs, the graph acts in both priors only by aggregating neighbouring features into a node or not at all. Real graphs show a wider range of relations between features and structure (Zhu et al., 2020; Platonov et al., 2023a), and half of the 28 benchmarks lie outside the range generated by either prior (Figure 4a). A PFN learns the relations generated by its prior, and on the heterophilous benchmarks our prior alone increases the accuracy of the same network (Table 2).

Graph neural diffusion as the prior. So far, graph neural diffusion has been used to design network architectures. We instead use it to design the prior of our graph PFN, which we call GADRPFN[1], in the same way that drift was built into the prior of TabPFN (Helli et al., 2024). In our prior, a hidden node state evolves on the sampled graph under the advection–diffusion–reaction equation of graph neural diffusion (Chamberlain et al., 2021; Choi et al., 2023; Eliasof et al., 2024). Each task draws the terms of the equation and the time of the flow. Diffusion smooths the state along the edges, and reaction sharpens it against them. The labels can additionally be smoothed or sharpened for a *separate label time* (§3.1). We call this a graph neural diffusion prior. NodePFN's prior is the special case in which the state does not depend on the graph and the graph is generated from the labels, and GraphPFN's GCN aggregation is one explicit Euler step of diffusion (§3.3). This allows us to compare priors with the graphs and the labels held fixed. The network integrates the same equation on the observed graph with learned coefficients (§3.2).

Our prior increases accuracy. On 28 benchmarks under the protocol of GraphAny (Zhao et al., 2025), GADRPFN reaches 75.3 mean accuracy against 72.0 for the strongest baseline (Figure 1b). It is trained from scratch in about two hours on one GPU with a fraction of their pretraining data, and its accuracy is still increasing at the end of pretraining (Figure 1a). It achieves the best average rank on both homophilous and heterophilous graphs, and its gain over NodePFN is larger on heterophilous graphs (§4.2). With the network, the recipe and the budget fixed, replacing NodePFN's prior with ours increases the mean accuracy on both networks (§4.3). On our prior, the network relies more on its graph layer and remains more non-linear throughout pretraining, whereas on NodePFN's prior it becomes closer to linear once its accuracy stops increasing (§4.5).

Contributions. (i) A space of priors for node classification in which the time and the direction of the graph's action on the features and on the labels are drawn per task, and a network that integrates

[1]Pronounced *gadder*: a Graph Advection–Diffusion–Reaction prior-fitted network.

the same equation (§3). (ii) A comparison on 28 benchmarks and three further protocols (§4.2), in which GADRPFN outperforms models built on tabular foundation models with far less pretraining data (Figure 1a). (iii) Controlled comparisons that change only the prior (§4.3), which our sampler allows because it contains NodePFN's prior and can use GraphPFN's aggregation (Appendix B). (iv) An analysis of what a graph prior generates (§4.4) and what the network learns from it (§4.5).

2 BACKGROUND AND RELATED WORK

Problem setting and PFNs. A task is a graph $G = (\mathcal{V}, \mathcal{E})$ with N nodes, normalised adjacency $\hat{A}$ with self loops and Laplacian $L = I - \hat{A}$, a feature matrix $X \in \mathbb{R}^{N \times F}$, labels y over c classes, and a context set $\mathcal{C} \subset \mathcal{V}$ of labelled nodes. The network receives $(G, X, y_{\mathcal{C}})$ and returns a class distribution for every node outside $\mathcal{C}$ in one forward pass. A PFN is pretrained on tasks $\xi = (G, X, y)$ drawn from a synthetic prior $p(\xi)$ by minimising

$$\mathcal{L}(\theta) = \mathbb{E}_{\xi \sim p(\xi)}\, \mathbb{E}_{\mathcal{C}} \Big[- \textstyle\sum_{v \notin \mathcal{C}} \log q_\theta\big(y_v \mid G, X, y_{\mathcal{C}}\big) \Big],$$

over context sets of random size and membership. The minimiser of $\mathcal{L}$ over q_θ is the posterior predictive distribution under $p(\xi)$, so in-context prediction on a real task approximates Bayesian inference under the prior (Müller et al., 2022).

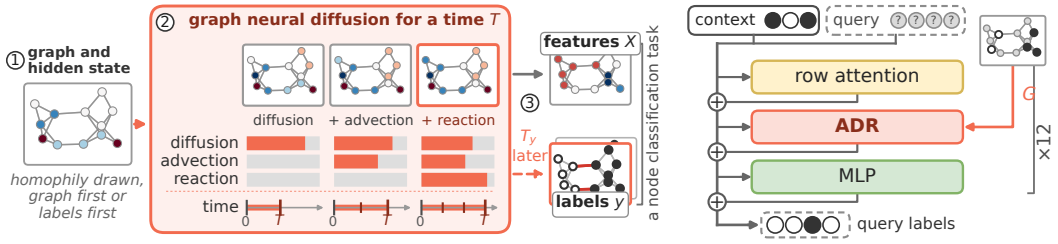
Priors of PFNs. TabPFN (Hollmann et al., 2023; 2025) and TabICL (Qu et al., 2025) generate a table by propagating random causes through a random structural causal model (SCM), and Mitra extends this prior with tree-based generators (Zhang et al., 2025b). Priors for time series generate dynamical processes, from Gaussian-process kernels composed with an SCM (Xie et al., 2026) to chaotic systems (Lai et al., 2026). Recent work compares the tasks of a prior with real data through dataset statistics and relates it to accuracy (Türkmen et al., 2026; Zhao et al., 2026).

PFNs for node classification. NodePFN (Choi et al., 2026) and GraphPFN (Eremeev et al., 2026) pretrain a network for node classification on synthetic graphs drawn from a prior. NodePFN is trained from scratch and processes a task with context–query attention and a GCN layer. GraphPFN adds attention-based neighbourhood aggregation to the tabular foundation model LimiX (Zhang et al., 2025a) and takes the raw feature matrix as input. TAG (Hayler et al., 2025) and TabPFN-GN (Choi et al., 2025) instead convert the graph into a table for TabPFN without graph pretraining. §3.3 relates the priors of NodePFN and GraphPFN to our sampler.

Towards graph foundation models. Other work also aims to build a single model for many graphs. GraphAny (Zhao et al., 2025) and TS-GNNs (Finkelshtein et al., 2025) classify a new graph without training on it, by weighting closed-form predictors and through label-permutation-equivariant statistics, respectively. Graph foundation models pretrain on collections of real graphs and adapt to a new one from its labelled nodes or by fine-tuning on it, some with a language model (Ma et al., 2026; Eliasof et al., 2026; Lee et al., 2026; Liu et al., 2024; Xia & Huang, 2026; Gutteridge et al., 2026). Appendix E describes these models and why Table 1 excludes them.

Graph neural diffusion. GRAND (Chamberlain et al., 2021) interprets message passing as the integration of a diffusion equation on the graph. One explicit Euler step of $\partial_t X = -LX$ of size τ gives $(1 - \tau)X + \tau \hat{A}X$, which for $\tau = 1$ is the averaging of a GCN layer. Subsequent models add further terms to the equation. Reaction counters oversmoothing on heterophilous graphs (Choi et al., 2023), and advection transports features along the edges (Eliasof et al., 2024; Wu et al., 2025). Repulsive forces sharpen a node against its neighbours (Wang et al., 2023; Di Giovanni et al., 2023), and a second time derivative makes the flow oscillate (Rusch et al., 2022). Han et al. (2024) survey these models. We take the equation and its terms from this line of work, draw their coefficients per task in the prior (§3.1) and learn them in the network (§3.2).

Two statistics of a graph. We characterise real and synthetic graphs by the extent to which their labels and features follow the edges. For the labels we use the adjusted homophily h_{adj} of Platonov et al. (2023a), which is 0 for random labels regardless of the number and the sizes of the classes, and negative when neighbours share a label less often than random labels would. For the features we



(a) One task of the prior.

(b) Network architecture.

Figure 2: **Graph neural diffusion in the prior and in the network.** (a) A hidden node state from a random causal model flows on a random graph under equation (1) for a time T, and the features and a target are read before or after it. Three tasks show the drawn terms and time with a tick per integration step, and the labels of the coral one are smoothed or sharpened for a separate label time T_y. (b) Twelve blocks of row attention, ADR on G with learned coefficients, and an MLP.

use a Dirichlet-energy form of feature smoothness (Hou et al., 2020), which is 0 for features drawn without the graph (Appendix H). Studies of heterophily usually vary how far the labels follow the edges (Zhu et al., 2020; Platonov et al., 2023b), often using contextual stochastic block models that generate the graph and the features from the labels (Deshpande et al., 2018; Chien et al., 2021). In such graphs the features follow the edges only through the labels.

3 GADRPFN

A GADRPFN task is generated in three steps (Figure 2a). Step ① samples a graph and a hidden node state on it, ② evolves the state under graph neural diffusion for a sampled time T, and ③ reads the features and labels from it. We call such a prior a graph neural diffusion prior, since the graph acts on the state through graph neural diffusion, which our network also integrates (§3.2).

Design goals. We design the prior to reflect properties of real graphs and to support controlled comparisons. On the 28 benchmarks the labels and the features follow the edges to different degrees. On Air-USA and BlogCatalog the labels follow the edges, with h_{adj} near 0.27, while the feature smoothness stays below 0.02, and on 6 of the 28 neighbours share a label less often than random labels would (Appendix H). The prior therefore draws its graphs from the same communities and causes as the features and averages the state along them, so the smoothness of the features varies widely across tasks (①, Figure 4a). Its block model controls how far the labels follow the edges, below or above the level of random labels (①). The flow lets the graph act on the state for a drawn time, smoothing it by diffusion and sharpening it by reaction (②). The separate label time moves the labels further along the edges than the features, or against them (③). It also contains NodePFN's prior and GraphPFN's GCN aggregation as special cases, so that priors can be compared with everything else fixed (§3.3).

3.1 GRAPH NEURAL DIFFUSION PRIOR

Figure 2a illustrates one task of the prior, with $N \in [256, 1024]$ nodes and up to 100 features.

① **Graph and state.** The sampler draws community memberships for the nodes and then the edges. Most graphs come from a block model whose edges favour a node's own community, other communities or one partner community, and the rest come from other random graph generators (Appendix A). The edge homophily is drawn per task from $[0.02, 0.98]$, below or above the level of random labels. In 40% of the tasks the sampler follows NodePFN's order instead, drawing the features and the label first and the graph from the labels. As in TabPFN and NodePFN, a random causal model maps Gaussian causes through random non-linear layers $Z^{(\ell+1)} = W_\ell\, \sigma(Z^{(\ell)})$+noise, with one cause vector per node shifted by its community. We call the $N \times d_z$ matrix of one layer, with d_z hidden units per node, the hidden node state Z. Between its layers the model may average

the state along the graph, and the features and the labels are columns of these layers (③), so the graph affects both through the state.

② **Graph neural diffusion for a time T.** At one to three random layers, Z is replaced by its state after flowing on the sampled graph for a time T,

$$\frac{dZ}{dt} = F(Z) = \underbrace{-\alpha\, LZ}_{\text{diffusion}} + \underbrace{\kappa\, VZ}_{\text{advection}} + \underbrace{\beta\, r(Z)}_{\text{reaction}} + \underbrace{\gamma\,(Z_0 - Z)}_{\text{restart}}, \qquad Z(0) = Z_0, \tag{1}$$

$$Z_{k+1} = Z_k + \tfrac{T}{K}\, F(Z_k), \qquad k = 0, \dots, K-1, \tag{2}$$

where L is the Laplacian of §2 and Z_0 the state where the flow is inserted. Diffusion averages a node with its neighbours (Chamberlain et al., 2021). The reaction r is one of the GREAD terms (Choi et al., 2023). Its band-pass form sharpens the state against the neighbours, and its Allen–Cahn and Zeldovich forms push each node towards stable values, which keeps communities apart. Advection transports the state along a velocity field V set by a random node potential (Eliasof et al., 2024; Wu et al., 2025). Restart pulls the state back towards Z_0 (Gasteiger et al., 2019). The coefficients are drawn per task from mixtures with mass at zero, so most tasks use one or two terms, which are unknown to the network, and some add a second-order term (Appendix A). The terms determine whether the state moves towards or away from its neighbours, and the time determines how far it moves. The time is drawn as $T \sim U(0, T_{\max})$ with $T_{\max} = 1$ and shared evenly by the layers, and K is two at the median and at most twelve (Appendix A).

③ **Observation.** The state itself is never observed. As in TabPFN, the features X and a scalar target are read from the causal model, from its causes, its output or random hidden nodes, so a feature can be read before or after the flow. In half of the tasks the labels are the community memberships, as in a block model. In the other half the target is cut into classes at quantiles, as in a tabular prior, and in the graph-first order it is first smoothed or sharpened on the graph for a further time $T_y \sim U(0, T_{\max})$, or left unchanged (Appendix A). Smoothing makes neighbours share a class more often than the features suggest, and sharpening places them in different classes. We call T_y the *separate label time*. The labels thus reflect another point of the flow than the features (Appendix H).

3.2 Network architecture

The network is a stack of 12 blocks of width $d = 512$ over one row per node (Figure 2b). A node's row starts as a linear embedding of its feature vector, plus an embedding of its label if the node is in the context (Appendix A). Each block applies three residual branches in turn, each behind a normalisation layer N,

$$h \leftarrow h + \mathrm{Attn}(\mathrm{N}(h)), \qquad h \leftarrow h + \mathrm{ADR}(\mathrm{N}(h);\, G), \qquad h \leftarrow h + \mathrm{MLP}(\mathrm{N}(h)), \tag{3}$$

and only the middle branch uses the edges. The first branch is self-attention among the rows of the task with the in-context mask, context rows and query rows alike attending to the context rows only. Through it, a query node is compared with the labelled nodes. The last branch is the feed-forward network of a Transformer block, which transforms each row independently. The row embedding follows NodePFN, and Appendix A details the attention, the heads and the ADR layer.

ADR layer. So that the graph module has the terms of the prior's equation, ADR integrates equation (1) on the observed graph with coefficients learned per channel instead of drawn. The input row is mapped to $h_0 = W_{\text{in}} x$ and takes $K = 2$ steps of equation (2) over unit time, with a learned momentum between them, on

$$F_\theta(h) = \underbrace{-\alpha\,(I - A_{\text{att}})\, h}_{\text{diffusion}} + \underbrace{\kappa\, Vh}_{\text{advection}} + \underbrace{b_{\text{ac}}\,(h - h^3) + b_{\text{bs}}\,(\hat{A} - \hat{A}^2)\, h}_{\text{reaction}} + \underbrace{\gamma\,(h_0 - h)}_{\text{restart}}, \tag{4}$$

and returns $W_{\text{out}}(h_K - h_0)$, which is the change the flow makes to the row. Equation (4) has the terms of F in equation (1), with learned coefficients θ (Table 4 in Appendix A). The layer learns two operators that the prior keeps fixed. The diffusivity A_{att} replaces $\hat{A}$ with an edge softmax that lets the layer choose which neighbours to average, as in GRAND, and the velocity field V comes from a learned potential. The coefficients and the momentum are per-channel parameters (Appendix A). Since A_{att} is initialised as the mean-neighbour operator, the layer starts from the diffusion of the prior. §4.3 compares other graph modules.

3.3 Relation to the existing priors

This section relates the priors of NodePFN and GraphPFN to our sampler, which allows §4.3 to exchange priors with everything else fixed. With the flow and the averaging of ① off, and the other switches of Appendix B, the state does not depend on the graph and the sampler draws NodePFN's tasks. GraphPFN's GCN aggregation is one Euler step of diffusion alone with $\alpha T = 1$ on the normalised adjacency without self loops. Equation (2) takes such a step as $Z_T = (1-\lambda)Z_0 + \lambda \hat{A} Z_0$ with $\lambda = \alpha T$, which is the averaging layer of ① and differs from GraphPFN's step only by the self loops of $\hat{A}$. We verify both identities in the code, and Table 5 lists the switches (Appendix B).

In addition, the sampler includes terms that neither prior has. As in GREAD, the reaction sharpens the state or keeps communities apart, advection transports the state, and restart retains part of the initial state. The labels can also be smoothed or sharpened for a separate label time.

3.4 Pretraining and inference

Every pretraining step draws eight tasks of up to $1{,}024$ nodes from the prior and minimises the loss of §2 on context sets of random size. The final model trains for 60 epochs of $1{,}024$ steps in about two hours on one GPU (Appendix A.3). At inference all labelled nodes form the context of one forward pass. As in NodePFN, the features are reduced in dimension and smoothed along the graph, and an ensemble over feature permutations is averaged, with these settings chosen per dataset (Appendix D). Feature columns beyond the input width of the network are subsampled, and classes beyond its class count are read through random class partitions (Appendix C).

4 Experiments

§4.2 compares GADRPFN with the in-context node classifiers on 28 benchmarks and with recent models on the graphs they report, and then with fewer labels, on text-attributed graphs and on a large graph, to test whether its lead holds outside that protocol. §4.3 changes the prior with the network fixed, and the network with the prior fixed, to test whether the prior accounts for the lead. §4.4 and §4.5 then examine how the prior changes the generated graphs and the network pretrained on it.

4.1 Setup

Benchmarks and baselines. We use 28 benchmarks under the protocol of GraphAny (Zhao et al., 2025). They are the 23 graphs of NodePFN's Table 1 (Choi et al., 2026) and five more GraphAny graphs, split into NodePFN's homophilous and heterophilous blocks of 17 and 11 (Appendix C). Graphs without public splits have $20c$ labelled nodes, and accuracy is the mean over five runs. The baselines are the in-context node classifiers of §2, namely the four GraphAny models, TS-Mean, TAG, NodePFN and GraphPFN, and Appendix E adds supervised ones.

Budgets and settings. Our models share the recipe of §3.4, and every prior comparison is made at equal budget. GADRPFN is evaluated with per-dataset inference hyperparameters (Appendix D), and Appendix B gives the settings of the controlled comparisons.

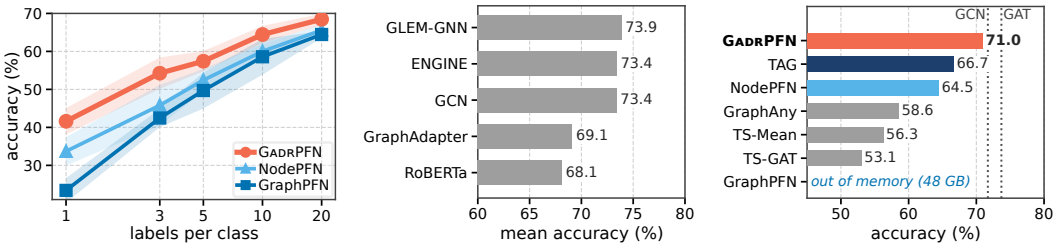
4.2 How well does GADRPFN classify nodes?

Main comparison. In Table 1, GADRPFN has the best mean accuracy, 75.3 against 72.0 for TAG and 71.7 for GraphPFN, and the best average rank on the homophilous graphs, on the heterophilous graphs and on all 28. It outperforms NodePFN on every graph, by a larger margin on the heterophilous block, and its difference from every baseline is significant under a paired Wilcoxon test with Holm correction (Appendix C).

Recent models. RGVT and Sigil-nc fit their predictor on each target graph's labels and report results on 27 and 23 of the graphs. On those, GADRPFN averages 74.9 against 71.6 for RGVT and 77.7 against 68.1 for Sigil-nc (Table 8). It also outperforms MeanGNN, TS-GAT and the Cora-trained GOBLIN on every graph they report (Table 9, Appendix E).

Table 1: Test accuracy on the 28 benchmarks. Bold marks the best per row and underline the runner-up, and the marks give the source or preprocessing of a cell (Appendix C).

Dataset	GraphAny (Products)†	GraphAny (Arxiv)†	GraphAny (Wisconsin)†	GraphAny (Cora)†	NodePFN	GraphPFN	TS-Mean†	TAG†	GADRPFN
homo.									
Air-Brazil	34.6 ± 16.5	34.6 ± 16.1	36.1 ± 16.7	33.1 ± 16.7	63.1 ± 6.2	73.8 ± 8.2	39.2 ± 5.7	73.1 ± 4.0	**80.8 ± 7.7**
Air-Europe	41.8 ± 6.8	41.5 ± 6.5	41.1 ± 6.0	40.5 ± 7.0	53.0 ± 4.8	58.6 ± 2.9	35.9 ± 6.9	55.4 ± 2.3	**60.8 ± 3.6**
Air-USA	43.6 ± 2.1	43.6 ± 1.8	43.9 ± 1.4	43.5 ± 1.4	60.6 ± 1.6	62.0 ± 0.8	42.3 ± 2.1	60.5 ± 0.8	**63.3 ± 1.8**
Cora	79.4 ± 0.2	79.4 ± 0.2	77.8 ± 1.1	80.2 ± 0.1	82.1 ± 0.3	76.1 ± 0.7	62.0 ± 12.0§	**83.5 ± 0.4**¶	82.2 ± 0.2
Citeseer	67.9 ± 0.3	68.3 ± 0.2	67.5 ± 0.4	68.9 ± 0.1	67.9 ± 0.3	62.3 ± 1.6	68.7 ± 0.2	68.1 ± 0.6	**70.8 ± 0.4**
Pubmed	76.5 ± 0.3	76.4 ± 0.2	77.5 ± 0.3	76.6 ± 0.3	78.6 ± 0.3	78.0 ± 1.6	75.0 ± 0.6	79.0 ± 0.4	**80.8 ± 0.3**
WikiCS	75.0 ± 0.5	75.0 ± 0.6	73.8 ± 0.8	74.4 ± 0.7	76.0 ± 0.8	78.6 ± 0.7	74.2 ± 2.1	79.1 ± 0.5	**80.8 ± 0.4**
Amazon-photo	90.6 ± 0.8	90.6 ± 0.8	90.2 ± 0.9	90.1 ± 0.9	89.5 ± 0.8	89.5 ± 0.5	90.2 ± 1.3	89.6 ± 0.5	**92.8 ± 0.6**
Amazon-comp.	82.9 ± 1.2	83.0 ± 1.2	82.0 ± 1.1	83.0 ± 1.2	79.5 ± 1.0	84.1 ± 0.9*	81.4 ± 1.2	84.3 ± 0.3	**86.9 ± 0.8**
DBLP	70.6 ± 1.0	70.9 ± 0.9	70.1 ± 0.8	71.7 ± 0.9	76.5 ± 1.5	67.6 ± 0.7	66.4 ± 3.6	71.9 ± 1.5	**78.9 ± 1.2**
Coauthor-CS	90.5 ± 0.5	90.5 ± 0.6	90.8 ± 0.6	90.5 ± 0.6	90.9 ± 0.5	91.7 ± 0.2*	90.9 ± 0.5	90.5 ± 0.5	**92.3 ± 0.2**
Coauthor-phys.	92.7 ± 0.5	92.7 ± 0.5	92.5 ± 0.4	92.7 ± 0.5	92.3 ± 0.8	91.5 ± 1.0*	92.6 ± 0.6	92.3 ± 0.3	**93.1 ± 0.2**
Deezer	52.1 ± 2.8	52.1 ± 2.8	52.1 ± 3.0	52.0 ± 2.8	51.2 ± 1.8	53.7 ± 1.5	52.3 ± 2.5	53.0 ± 1.6	**55.2 ± 0.6**
Wiki	63.1 ± 3.6	63.0 ± 3.7	61.1 ± 4.4	60.6 ± 3.6	66.4 ± 1.6	**79.5 ± 1.1***	69.9 ± 1.3	70.1 ± 0.9	75.0 ± 1.2
BlogCatalog	74.7 ± 3.2	73.6 ± 3.0	77.7 ± 1.9	72.5 ± 3.2	63.4 ± 2.4	82.6 ± 0.7*	76.3 ± 2.9	79.8 ± 1.1	**86.4 ± 0.4**
LastFM Asia	80.2 ± 0.4	80.6 ± 0.6	79.5 ± 1.2	80.8 ± 0.4	77.9 ± 2.1	79.2 ± 1.4*	78.0 ± 1.0	85.5 ± 0.3	**85.9 ± 0.4**
FullCora°	57.1 ± 0.4	57.2 ± 0.4	56.3 ± 0.2	56.7 ± 0.4	47.5 ± 3.6	56.7 ± 1.1*	53.6 ± 0.7	54.6 ± 0.2	**60.5 ± 0.3**
Mean (homo.)	69.02	69.00	68.83	68.69	71.55	74.45	67.58	74.71	**78.03**
Rank (homo.)	5.53	5.47	6.26	5.94	5.82	4.53	6.53	3.79	**1.12**
hetero.									
Cornell	64.9 ± 0.0	65.9 ± 1.5	66.5 ± 1.5	64.9 ± 1.9	71.9 ± 2.8	73.0 ± 3.8	68.7 ± 2.4	75.1 ± 2.2	**77.8 ± 1.1**
Texas	73.5 ± 3.0	73.0 ± 2.7	73.5 ± 1.2	71.9 ± 1.5	74.6 ± 6.5	69.5 ± 21.8	73.5 ± 4.0	81.1 ± 2.1	**85.9 ± 4.0**
Wisconsin	65.9 ± 2.2	65.1 ± 3.2	71.8 ± 6.0	61.2 ± 5.1	80.0 ± 6.1	82.7 ± 4.7	61.2 ± 11.4	83.1 ± 1.3	**89.8 ± 3.8**
Chameleon	32.8 ± 5.2¶	32.7 ± 4.8¶	32.2 ± 4.2¶	32.2 ± 4.1¶	41.2 ± 2.4	46.7 ± 2.5	43.8 ± 4.0¶	45.0 ± 1.4¶	**50.4 ± 4.2**
Actor	29.0 ± 0.6	28.6 ± 0.2	29.5 ± 0.6	27.9 ± 0.2	33.2 ± 0.3	**38.0 ± 1.3**	28.1 ± 0.9	31.2 ± 0.2	37.9 ± 1.0
Minesweeper	80.3 ± 0.2	80.3 ± 0.1	80.1 ± 0.1	80.5 ± 0.1	80.7 ± 0.3	83.6 ± 0.5	80.7 ± 0.4	**85.8 ± 0.1**	**85.8 ± 0.2**
Tolokers	78.2 ± 0.0	78.2 ± 0.0	78.2 ± 0.0	78.2 ± 0.0	78.6 ± 0.2	**83.9 ± 0.6**	78.1 ± 0.1	82.8 ± 0.1	82.4 ± 0.6
Amazon-ratings	42.7 ± 0.1	42.7 ± 0.1	42.6 ± 0.3	42.8 ± 0.0	44.4 ± 0.4	48.7 ± 0.4	42.3 ± 1.4	44.3 ± 0.3	**50.6 ± 0.7**
Questions	97.1 ± 0.0	97.1 ± 0.0	97.1 ± 0.0	97.1 ± 0.0	97.0 ± 0.0	**97.3 ± 0.0***	97.0 ± 0.0	97.1 ± 0.0	**97.3 ± 0.0**
Squirrel	26.5 ± 1.3¶	25.9 ± 1.1¶	25.6 ± 1.1¶	24.9 ± 1.7¶	43.2 ± 1.0	**48.7 ± 1.3**	39.8 ± 0.5¶	46.3 ± 1.0¶	48.2 ± 2.6
Roman-Empire	64.7 ± 0.8	64.2 ± 1.1	64.1 ± 0.8	64.2 ± 0.6	47.7 ± 1.8	69.6 ± 0.6	66.4 ± 1.0	74.1 ± 0.3	**75.7 ± 0.5**
Mean (hetero.)	59.58	59.43	60.11	58.70	62.95	67.43	61.77	67.82	**71.10**
Rank (hetero.)	6.18	6.73	6.73	7.32	4.73	2.68	6.41	2.77	**1.45**
Mean (all 28)	65.31	65.24	65.40	64.77	68.17	71.69	65.30	72.01	**75.31**
Rank (all 28)	5.79	5.96	6.45	6.48	5.39	3.80	6.48	3.39	**1.25**



(a) Six graphs with few labels. (b) Six GLBench graphs. (c) ogbn-arxiv with 169k nodes.

Figure 3: **Beyond the protocol of Table 1.** (a) Bands are one standard deviation over five label draws. (b) Mean over six of the seven graphs, with grey methods trained on each graph's labels. (c) None of the models shown as bars trains on ogbn-arxiv, unlike the dotted GCN and GAT.

Fewer labels per class. Under the ProG few-shot protocol (Zi et al., 2024) on six graphs, with 1 to 20 labelled nodes per class, GADRPFN has the highest mean accuracy of the three PFNs at every budget, and its lead over GraphPFN grows as labels become scarcer, to 41.6 against 23.4 at one labelled node per class (Figure 3a, per graph in Appendix F.1). From three labelled nodes per class it outperforms ProG's fine-tuned GNN on Cora and Citeseer and its prompt-tuning methods on Cora.

Text-attributed graphs. On the seven graphs of GLBench (Li et al., 2024), GADRPFN trains on none of the graphs and encodes each node's text with a frozen 110M-parameter encoder, far smaller

than the 7B language model that six of the LLM-based methods run. It has the best mean accuracy and rank, 75.8 against 74.2 for GLEM-GNN, and the best accuracy on four of the seven graphs (Appendix F.2). NodePFN and GraphPFN use the same vectors and remain below it (Figure 3b), so the encoder does not explain its lead over them, whereas its lead over the GNNs relies on it.

Large graphs. Beyond the 1,024 nodes of a pretraining task, accuracy falls on none of the eight benchmarks with more labelled nodes and rises by 9.6% on Roman-Empire (Appendix G.2). On ogbn-arxiv, with 91k labelled nodes as context, GADRPFN reaches 71.0 against 66.7 for TAG and is the best of models that do not train on it (Figure 3c). GCN, GAT, RGVT and Sigil-nc, trained on those labels, reach 70.3 to 73.7, and GADRPFN is within that range without training (Appendix F.3).

4.3 DOES THE GAIN COME FROM THE PRIOR OR THE NETWORK?

Changing only the prior increases accuracy. With the network, the recipe and the budget fixed, replacing NodePFN's prior with ours increases the mean accuracy by 3.8% on NodePFN's network and 5.1% on ours (Table 2). Our prior with GraphPFN's aggregation keeps our graphs and labels and replaces our diffusion with GraphPFN's aggregation (Appendix B). It accounts for about half of the gain on both networks, so our graphs and labels account for that half, and our diffusion adds the rest, 2.6% on our network and 1.5% on NodePFN's.

Table 2: **Three priors on two networks.** Mean accuracy over the 28 benchmarks. GADRPFN w/ GraphPFN agg. is our prior with one GraphPFN aggregation per layer of its causal model in place of its averaging, flow and separate label time (Appendix B).

Network	NodePFN's			Ours		
Prior	All	Hom.	Het.	All	Hom.	Het.
NodePFN's prior	68.2	71.6	63.0	70.2	73.0	65.9
GADRPFN w/ GraphPFN agg.	70.5	73.4	65.9	72.7	75.0	69.2
GADRPFN prior	72.0	75.5	66.5	75.3	78.0	71.1

The gap grows with pretraining. On our network the gap between the two priors grows over pretraining, mostly on the heterophilous block (Appendix G.3), and reaches 5.1% at the end (Table 2).

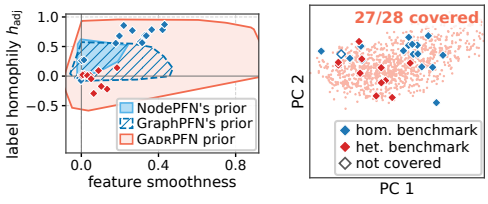
Table 3: **Our network with one part changed.** Mean accuracy over the 28 benchmarks on the GADRPFN prior.

Network ablation	All
GADRPFN (full)	75.3
w/ GCN layer (NodePFN's)	74.6
w/o ADR layer	71.6

The network needs a graph module. Without a graph module the accuracy of our network drops from 75.3 to 71.6 (Table 3), so smoothing the features at inference does not replace it. The ADR layer outperforms NodePFN's GCN layer (74.6) and alone contains the prior's terms (Table 4).

4.4 WHAT DOES EACH PRIOR GENERATE?

Each prior is represented by 1,000 synthetic tasks. GraphPFN's prior here is GraphPFN's own prior, sampled with the code and pretraining configuration that GraphPFN released. It differs from the row of Table 2 that uses GraphPFN's aggregation in our prior. Besides the feature smoothness and the label homophily of Figure 4a, Figure 4b and Table 15 use four more statistics of each graph. They are its mean degree, its number of classes, its class imbalance and how closely each feature follows its average over the neighbours (Appendix H). Following Xie et al. (2026) and Naeem et al. (2020), Table 15 measures how close five sets of 200 draws are to the benchmarks and how many they cover (Appendix H).



(a) Region of each prior. (b) Coverage.

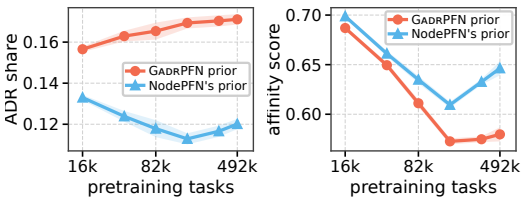
Figure 4: **What each prior generates.** (a) The region of 99% of each prior's draws. (b) GADRPFN draws and the benchmarks on two principal components of the six statistics.

Half the benchmarks lie outside both existing priors. In Figure 4a neither existing region reaches below an adjusted homophily of -0.1, and NodePFN's stays below a feature smoothness of 0.25. In $1{,}000$ draws of either prior, at most one task in a hundred has features as smooth along the edges as the coauthorship graphs, and none has labels that run against the edges as on the WebKB graphs. The fourteen benchmarks outside both are Texas, Cornell and Wisconsin, whose labels run against the edges, and eleven of the twelve homophilous graphs with h_{adj} above 0.55. The region of the GADRPFN prior contains all 28, and in six statistics its draws cover 27 (Figure 4b).

The GADRPFN prior comes closest on most measures. In Table 15 it has the smallest distribution gap in six statistics, 0.47 against 0.79 for GraphPFN's prior and 1.22 for NodePFN's, and covers the most benchmarks in both spaces. Its draws lie nearest to the homophilous benchmarks in both spaces. On the heterophilous ones they are nearest in the two statistics and comparable to GraphPFN's prior in six, 0.90 against 0.95. These measures characterise the priors but do not by themselves explain Table 2, since similarity in task statistics only partly explains accuracy (Türkmen et al., 2026; Zhao et al., 2026).

4.5 WHAT DOES THE NETWORK LEARN FROM THE PRIOR?

The network relies more on its graph layer. We track the networks on NodePFN's prior and on ours at six checkpoints on six probe graphs (Appendix I). The share of the residual update written by the ADR branch of equation (3) grows from 0.157 to 0.171 on our prior and falls from 0.133 to 0.120 on NodePFN's prior (Figure 5a). It is higher on our prior on every graph and checkpoint and ends 1.4 times as high, mostly at the expense of attention.



(a) Use of the ADR layer. (b) Non-linearity.

Figure 5: **What the network learns from the prior.** (a) Share of the residual update written by the ADR branch. (b) Affinity score of the activations. Bands show within-graph standard errors.

The network remains more non-linear. How non-linear a pretrained network is reflects its expressive power and depends on its pretraining data, and more diverse synthetic data makes it more non-linear (Bouniot et al., 2025; Xie et al., 2026). Our prior covers more of the benchmarks (§4.4). The affinity score of Bouniot et al. (2025) is 1 for an affine activation and lower the more non-linear it is. After $163{,}840$ tasks the score of the network on NodePFN's prior rises from 0.61 to 0.65, so it computes closer to a linear map of its input, while its accuracy gains less than 1%. The network on our prior stays near 0.58 while its accuracy rises (Figure 5b). This is consistent with the more varied tasks of our prior.

5 CONCLUDING REMARKS

We studied how the prior of a graph PFN shapes how the graph relates to node features. GADRPFN generates each task with graph neural diffusion, in which a node state on a random graph evolves under an advection–diffusion–reaction equation for a random time. Trained from scratch on far less data than models built on tabular foundation models, it has the best mean accuracy and average rank on homophilous and heterophilous benchmarks. With the network, the recipe and the budget fixed, replacing NodePFN's prior with ours increases the mean accuracy of both networks, and on our prior the network relies more on its graph layer and remains more non-linear.

Limitations and scope. Our scope is in-context node classification on graphs treated as undirected, without the fine-tuning that GraphPFN and TAG also support, and graphs with edge features, edge types or time stamps are outside the prior. Row attention is quadratic in the number of nodes.

Future work. Accuracy is still rising at the end of our pretraining (Appendix G.3), so scaling the pretraining and the network is the next step. The prior could also be extended beyond node classification to node regression, link prediction and graph-level tasks, and to directed and dynamic graphs. The approach may also apply wherever a prior must specify how a structure shapes the data.

AI USE STATEMENT

We used the large-language-model assistant Claude (Anthropic, 2026) and the machine translation service DeepL Translator (DeepL SE, 2026), and the authors made the final decisions. Claude gave feedback on the analyses and on the interpretation of their results, and it helped write the code that plots the figures, summarise the related work that the authors had found and cited, and check the reported numbers against the result files. The authors interpreted the results and revised the figures. They conceived the idea of GADRPFN, designed its prior, network and training method, generated the synthetic data, prepared and processed the datasets, chose the inference settings and ran the experiments. They also set the structure of the manuscript and its terminology, used Claude and DeepL to translate and refine the text, and then revised it.

ETHICS STATEMENT

GADRPFN is pretrained only on synthetic graphs, so no real or personal data enters its weights. We evaluate it on public benchmarks used in prior work, whose sources and preprocessing are given in Appendices C and F.2. Some of these benchmarks are networks of users, and we use them as released and only to measure accuracy. The study collects no new data, involves no human subjects and reports no predictions about individuals. We are not aware of other concerns under the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

The inference code of GADRPFN, the checkpoint of Table 1 and the per-dataset inference settings behind its GADRPFN column are at `https://anonymous.4open.science/r/GadrPFN/`. One script runs the 28 benchmarks from one configuration file and reports the mean and standard deviation of each. Every split is drawn from a fixed seed as in Appendix C, and every dataset is public, with its source named in the repository. The sampler of §3.1 with its coefficient mixtures is in Appendix A, the pretraining recipe in §3.4 and Appendix A, the split protocol and the source of every copied baseline cell in Appendix C, the per-dataset inference hyperparameters of Table 1 and their search ranges in Appendix D, and the readings and budgets of the controlled comparisons in Appendix B. The pretraining code, the sampler and the scripts behind the other tables will be released upon acceptance.

REFERENCES

Anthropic. Claude. https://www.anthropic.com/claude, 2026.

Gleb Bazhenov, Oleg Platonov, and Liudmila Prokhorenkova. GraphLand: Evaluating graph machine learning models on diverse industrial data. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025.

Maya Bechler-Speicher, Yoel Gottlieb, Andrey Isakov, David Abensur, Ami Tavory, Daniel Haimovich, Ido Guy, and Udi Weinsberg. Billion-scale graph foundation models. *arXiv preprint arXiv:2602.04768*, 2026.

Quentin Bouniot, Ievgen Redko, Anton Mallasto, Charlotte Laclau, Oliver Struckmeier, Karol Arndt, Markus Heinonen, Ville Kyrki, and Samuel Kaski. From Alexnet to transformers: Measuring the non-linearity of deep neural networks with affine optimal transport. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pp. 25250–25260, 2025.

Benjamin Paul Chamberlain, James Rowbottom, Maria I. Gorinova, Michael M. Bronstein, Stefan Webb, and Emanuele Rossi. GRAND: Graph neural diffusion. In *ICML*, 2021.

Haibo Chen, Xin Wang, Zeyang Zhang, Haoyang Li, Ling Feng, and Wenwu Zhu. AutoGFM: Automated graph foundation model with adaptive architecture customization. In *International Conference on Machine Learning (ICML)*, 2025.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized PageRank graph neural network. In *ICLR*, 2021.

Jeongwhan Choi, Seoyoung Hong, Noseong Park, and Sung-Bae Cho. GREAD: Graph neural reaction-diffusion networks. In *ICML*, 2023.

Jeongwhan Choi, Woosung Kang, Minseo Kim, Jongwoo Kim, and Noseong Park. Can TabPFN compete with GNNs for node classification via graph tabularization? *arXiv preprint arXiv:2512.08798*, 2025.

Jeongwhan Choi, Jongwoo Kim, Woosung Kang, and Noseong Park. Learning posterior predictive distributions for node classification from synthetic graph priors. In *ICLR*, 2026.

DeepL SE. DeepL Translator. https://www.deepl.com/translator, 2026.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

Yash Deshpande, Andrea Montanari, Elchanan Mossel, and Subhabrata Sen. Contextual stochastic block models. In *Advances in Neural Information Processing Systems*, 2018.

Francesco Di Giovanni, James Rowbottom, Benjamin P. Chamberlain, Thomas Markovich, and Michael M. Bronstein. Understanding convolution on graphs via energies. *Transactions on Machine Learning Research*, 2023.

Samuel Dooley, Gurnoor Singh Khurana, Chirag Mohapatra, Siddartha V. Naidu, and Colin White. ForecastPFN: Synthetically-trained zero-shot forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Moshe Eliasof, Eldad Haber, and Eran Treister. Feature transportation improves graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 11874–11882, 2024.

Moshe Eliasof, Krishna Sri Ipsit Mantri, Beatrice Bevilacqua, Bruno Ribeiro, and Carola-Bibiane Schönlieb. Bridging input feature spaces towards graph foundation models. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2605.04834.

Dmitry Eremeev, Oleg Platonov, Gleb Bazhenov, Artem Babenko, and Liudmila Prokhorenkova. GraphPFN: A prior-data fitted graph foundation model. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2509.21489.

Ben Finkelshtein, İsmail İlkan Ceylan, Michael Bronstein, and Ron Levie. Equivariance everywhere all at once: A recipe for graph foundation models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized PageRank. In *ICLR*, 2019.

Benjamin Gutteridge, Michael Bronstein, and Xiaowen Dong. Can graph foundation models generalize over architecture? In *ICLR 2026 Workshop on Geometry-grounded Representation Learning and Generative Modeling (GRaM)*, 2026. arXiv:2603.22984.

Andi Han, Dai Shi, Lequan Lin, and Junbin Gao. From continuous dynamics to graph neural networks: Neural diffusion and beyond. *Transactions on Machine Learning Research*, 2024.

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

Kai Helli, David Schnurr, Noah Hollmann, Samuel Müller, and Frank Hutter. Drift-resilient TabPFN: In-context learning temporal distribution shifts on tabular data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *ICLR*, 2023.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637:319–326, 2025.

Yifan Hou, Jian Zhang, James Cheng, Kaili Ma, Richard T. B. Ma, Hongzhi Chen, and Ming-Chang Yang. Measuring and improving the use of graph information in graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2020.

Yan Jiang, Ruihong Qiu, and Zi Huang. GFMate: Empowering graph foundation models with test-time prompt tuning. In *International Conference on Machine Learning (ICML)*, 2026.

Jeffrey Lai, Anthony Bao, and William Gilpin. Panda: A pretrained forecast model for chaotic dynamics. In *International Conference on Learning Representations (ICLR)*, 2026.

Dooho Lee and Jaemin Yoo. Node4All: Learning node representation beyond datasets. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2026.

Dooho Lee, Myeong Kong, Minho Jeong, and Jaemin Yoo. View space: Learning representation across arbitrary graphs. In *International Conference on Machine Learning (ICML)*, 2026.

Haoyang Li, Haibo Chen, Xin Wang, and Wenwu Zhu. Out-of-distribution generalization in graph foundation models. *arXiv preprint arXiv:2601.21067*, 2026.

Yuhan Li, Peisong Wang, Xiao Zhu, Aochuan Chen, Haiyun Jiang, Deng Cai, Victor Wai Kin Chan, and Jia Li. GLBench: A comprehensive benchmark for graph with large language models. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024.

Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. One for all: Towards training one graph model for all classification tasks. In *ICLR*, 2024.

Weishuo Ma, Yanbo Wang, Xiyuan Wang, Lei Zou, and Muhan Zhang. GILT: An LLM-free, tuning-free graph foundational model for in-context learning. In *ICML 2026 Workshop on Graph Foundation Models*, 2026.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do Bayesian inference. In *ICLR*, 2022.

Samuel Müller, Matthias Feurer, Noah Hollmann, and Frank Hutter. PFNs4BO: In-context learning for Bayesian optimization. In *International Conference on Machine Learning (ICML)*, 2023.

Samuel Müller, Arik Reuter, Noah Hollmann, David Rügamer, and Frank Hutter. Position: The future of Bayesian prediction is prior-fitted. In *International Conference on Machine Learning (ICML)*, 2025.

Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *International Conference on Machine Learning (ICML)*, pp. 7176–7185, 2020.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-GCN: Geometric graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2020.

Oleg Platonov, Denis Kuznedelev, Artem Babenko, and Liudmila Prokhorenkova. Characterizing graph datasets for node classification: Homophily-heterophily dichotomy and beyond. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023a.

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

Oleg Platonov, Gleb Bazhenov, Dmitry Eremeev, and Liudmila Prokhorenkova. A fair evaluation of graph foundation models for node property prediction. *ICML 2026 Workshop on Graph Foundation Models*, 2026. arXiv:2606.24509.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *ICML*, 2025.

T. Konstantin Rusch, Benjamin P. Chamberlain, James Rowbottom, Siddhartha Mishra, and Michael M. Bronstein. Graph-coupled oscillator networks. In *ICML*, 2022.

Li Sun, Zhenhao Huang, Suyang Zhou, Qiqi Wan, Hao Peng, and Philip S. Yu. RiemannGFM: Learning a graph foundation model from Riemannian geometry. In *Proceedings of the ACM Web Conference (WWW)*, 2025.

Li Sun, Zhenhao Huang, Silei Chen, Lanxu Yang, Junda Ye, Sen Su, and Philip S. Yu. Multi-domain Riemannian graph gluing for building graph foundation models. In *International Conference on Learning Representations (ICLR)*, 2026.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. GraphGPT: Graph instruction tuning for large language models. In *SIGIR*, 2024.

Zeynep Türkmen, Kürşat Kaya, Alexander Pfefferle, and Frank Hutter. Towards evaluating data priors for tabular foundation models. *arXiv preprint arXiv:2606.29241*, 2026.

Kai Wang, Siqiang Luo, Caihua Shan, and Yifei Shen. Towards graph foundation models: Training on knowledge graphs enables transferability to general graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025a.

Yuelin Wang, Kai Yi, Xinliang Liu, Yu Guang Wang, and Shi Jin. ACMP: Allen-Cahn message passing with attractive and repulsive forces for graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2023.

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

Qitian Wu, Chenxiao Yang, Kaipeng Zeng, and Michael M. Bronstein. Supercharging graph transformers with advective diffusion. In *International Conference on Machine Learning (ICML)*, 2025.

Lianghao Xia and Chao Huang. AnyGraph: Graph foundation model in the wild. In *Findings of the Association for Computational Linguistics: ACL*, 2026.

Shifeng Xie, Vasilii Feofanov, Jianfeng Zhang, Themis Palpanas, and Ievgen Redko. CauKer: Classification time series foundation models can be pretrained on synthetic data. In *International Conference on Learning Representations (ICLR)*, 2026.

Omer Yom-Tov and Avigdor Gal. Neural message passing on structural interaction graphs for fully-inductive graph neural networks. *arXiv preprint arXiv:2608.08567*, 2026.

Xingtong Yu, Chang Zhou, Yuan Fang, and Xinming Zhang. Text-free multi-domain graph pre-training: Toward graph foundation models. *arXiv preprint arXiv:2405.13934*, 2024.

Xingxuan Zhang, Gang Ren, Han Yu, Hao Yuan, Hui Wang, Jiansheng Li, Jiayun Wu, Lang Mo, Li Mao, Mingchao Hao, Ningbo Dai, Renzhe Xu, Shuyang Li, Tianyang Zhang, Yue He, Yuanrui Wang, Yunjia Zhang, Zijing Xu, Dongzhe Li, Fang Gao, Hao Zou, Jiandong Liu, Jiashuo Liu, Jiawei Xu, Kaijie Cheng, Kehan Li, Linjun Zhou, Qing Li, Shaohua Fan, Xiaoyu Lin, Xinyan Han, Xuanyue Li, Yan Lu, Yuan Xue, Yuanyuan Jiang, Zimu Wang, Zhenlei Wang, and Peng Cui. LimiX: Unleashing structured-data modeling capability for generalist intelligence. *arXiv preprint arXiv:2509.03505*, 2025a.

Xiyuan Zhang, Danielle C. Maddix, Junming Yin, Nick Erickson, Abdul Fatir Ansari, Boran Han, Shuai Zhang, Leman Akoglu, Christos Faloutsos, Michael W. Mahoney, Cuixiong Hu, Huzefa Rangwala, George Karypis, and Yuyang Wang. Mitra: Mixed synthetic priors for enhancing tabular foundation models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b.

He Zhao, Ryan Thompson, Daniel M. Steinberg, Ashfaqur Rahman, Edwin V. Bonilla, and Cheng Soon Ong. From synthetic priors to model behavior: Structural coverage in tabular foundation models. *arXiv preprint arXiv:2609.06912*, 2026.

Jianan Zhao, Zhaocheng Zhu, Mikhail Galkin, Hesham Mostafa, Michael Bronstein, and Jian Tang. Fully-inductive node classification on arbitrary graphs. In *ICLR*, 2025.

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

Wei Zhuo and Siqiang Luo. Modality-free graph in-context alignment. In *International Conference on Learning Representations (ICLR)*, 2026.

Chenyi Zi, Haihong Zhao, Xiangguo Sun, Yiqing Lin, Hong Cheng, and Jia Li. ProG: A graph prompt learning benchmark. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024.

CONTENTS OF THE APPENDIX

A Implementation details **15**

A.1 The prior . . . 15

A.2 The network . . . 16

A.3 Pretraining and inference . . . 16

B Relation to the existing priors **16**

C Experimental setup in detail **17**

D Inference hyperparameters of Table 1 **18**

E Baselines outside the protocol of Table 1 **19**

F Further comparisons **22**

F.1 Few-shot node classification under the ProG protocol . . . 22

F.2 Text-attributed graphs . . . 24

F.3 Large-scale node classification . . . 25

G Cost and scaling **26**

G.1 Inference cost . . . 26

G.2 Beyond the pretraining context length . . . 27

G.3 Pretraining length . . . 28

H What each prior generates **29**

I What each prior teaches the network **30**

A IMPLEMENTATION DETAILS

A.1 THE PRIOR

① **Graph and state.** Both orders of §3.1 are used, the graph-first order in most tasks. Most graphs come from degree-corrected or plain block models and the rest from other random graph generators, with a mean degree between 2 and 40. The edge homophily is drawn per task, and some block models connect each community mainly to one partner community. In most tasks the communities shift the causes of their nodes, and the causal model may average the state along the graph between its layers.

② **Graph neural diffusion for a time T.** Table 4 lists the terms of the flow in the prior and in the ADR layer. Each coefficient is drawn per task from a fixed mixture that includes zero, so a task uses only some of the terms. A negative diffusion coefficient sharpens the state against its neighbours, as the repulsive forces of graph neural diffusion do (Wang et al., 2023; Di Giovanni et al., 2023). With inertia the flow becomes a damped wave (Rusch et al., 2022). In most tasks only part of the columns of the state flows, as GraphPFN aggregates only a share of its channels.

③ **Observation.** The separate label time applies to the labels cut from the target in the graph-first order. These labels are smoothed or sharpened for a time $T_y \sim U(0, T_{\max})$, or left unchanged.

Table 4: **The terms of equation (1) in the prior and in the ADR layer.** The prior draws each coefficient per task, and the ADR layer learns it per channel.

Term	Prior	ADR layer
diffusion $-\alpha LZ$	α positive, negative or zero	learned α, A_{att} in place of $\hat{A}$
advection κVZ	$\kappa \geq 0$, random node potential	learned κ, learned node potential
reaction $\beta r(Z)$	Allen–Cahn, Zeldovich or band-pass form	Allen–Cahn and band-pass forms
restart $\gamma(Z_0 - Z)$	$\gamma \geq 0$	learned γ
inertia	in some tasks	learned momentum
time	$T \sim U(0, 1)$	unit time, two steps

A.2 The network

Rows and attention. As in NodePFN, each node is represented by a row that embeds its features and, for context nodes, its label. Self-attention lets each query row attend to the context rows, and the network has 32.5 M parameters (Appendix G.1).

Heads. Two classification heads map each query row to a distribution over the classes, and their outputs are averaged with a weight set per dataset at inference (Appendix D).

ADR layer. The two integration steps carry a learned momentum, which corresponds to the inertia of the prior. The diffusivity A_{att} is an edge softmax over the neighbours of each node and starts as the mean over them, and the velocity field of advection comes from a learned node potential. The coefficients are learned per channel (Table 4).

A.3 Pretraining and inference

Runs train for 60 epochs of $1{,}024$ optimiser steps with eight tasks per step unless stated otherwise, so the final model is pretrained on $491{,}520$ synthetic tasks. Pretraining takes about two hours on one NVIDIA RTX A6000 GPU with 48 GB of memory.

B Relation to the existing priors

Table 5: **The three priors in the steps of §3.1.**

	NodePFN's prior	GraphPFN's prior	GADRPFN prior
① Order	labels first	graph first	either
① Graph	homophilous block model	block model with attachment	block model
① State	independent of the graph	channels partly aggregated	averaged on the graph
② Graph neural diffusion	none ($T = 0$)	one step per layer	every term, time T
③ Labels	with the features	with the features	separate label time T_y

GraphPFN's aggregations in the flow. In every layer of its causal model, GraphPFN's prior aggregates a random share p of the channels over the neighbours with one of five aggregations (GCN, mean, min, max or attention), on a graph from a multi-level block model with preferential attachment. Only the GCN aggregation corresponds to the flow, as one step of diffusion alone without self loops (§3.3), and the mean is the same step under row normalisation. Min, max and attention are not linear in the state, and none of the five is by itself a sharpening, transport or reaction term, although the causal model can combine their outputs with those of its other channels. The share p determines how often the state is aggregated, whereas the time and the coefficients of the flow determine how far it moves.

Our prior with GraphPFN's aggregation. This row of Table 2 replaces the diffusion of our prior with these aggregations. Every layer of the causal model aggregates a share $p \in \{0, 0.1, \ldots, 1\}$ of its channels with one of the five aggregations. The averaging and smoothing of ①, the flow of ②

and the separate label time of ③ are disabled. Like GraphPFN's prior, the causal model also takes the log-degree and the PageRank of each node as inputs, here in three quarters of the tasks. The graphs, the labels, the causal model and the task sizes remain those of our prior. We replace the diffusion rather than train on GraphPFN's whole prior, so that the comparison does not also change the graph generator and the causal model, which §4.4 compares separately.

Budgets and evaluation settings. Each prior pretrains both networks at the budget of that network. NodePFN's network uses NodePFN's inputs and training recipe for 30 epochs, and ours uses the recipe of Appendix A.3 for 60. The entry of NodePFN's prior on NodePFN's network is the released checkpoint of Table 1. The two existing priors are evaluated under NodePFN's released settings on both networks and the GADRPFN prior under the per-dataset settings of Table 1, so that its entry on our network is the model of Table 1 and the first row of Table 3.

Checks on the code. At $T = T_y = 0$ the sampler returns the same tasks as the sampler without the flow, bit for bit on 40 seeds. One coarse Euler step with $|\alpha|T \leq 1$ matches the averaging layer of ①, column scales included, to 5×10^{-7}, which is the identity of §3.3. The averaging layer uses $\hat{A} = \tilde{D}^{-1/2}(A + I)\tilde{D}^{-1/2}$, and GraphPFN's GCN aggregation uses $D^{-1/2}AD^{-1/2}$ without self loops, as does the row of Table 2 with its aggregation.

C EXPERIMENTAL SETUP IN DETAIL

Benchmarks. The 28 graphs of Table 1 are the 23 graphs of NodePFN's Table 1 (Choi et al., 2026) and five further graphs of the GraphAny benchmark (Zhao et al., 2025) (Wiki, BlogCatalog, LastFM Asia, FullCora and Roman-Empire)[2]. Chameleon and Squirrel are the filtered graphs of Platonov et al. (2023b), which come from the same release as their heterophily suite and Roman-Empire[3]. The large graph ogbn-arxiv of GraphAny's benchmark is evaluated in Appendix F.3.

Splits. We use the public splits of a graph when they are provided, as GraphAny does. These are the standard split of Cora, Citeseer and Pubmed, the ten splits of Pei et al. (2020) for WebKB and Actor, the ten splits of Platonov et al. (2023b) for their suite, Roman-Empire and the filtered Chameleon and Squirrel, and the official splits of WikiCS. On the other graphs a seeded draw labels 20 nodes per class and splits the rest evenly into validation and test nodes. Each number is the mean over five runs, on the first five public splits when a graph provides several, on five seeded draws when it provides none, and on the single public split of Cora, Citeseer and Pubmed.

NodePFN's released settings. NodePFN releases, together with its checkpoint, a per-dataset TSVD width, number of feature-smoothing steps and ensemble size. NodePFN in Table 1, the existing priors in Table 2 and the appendix analyses that name these settings use them unchanged.

Baselines. The marks of Table 1 give the source of every cell.

- † Copied under the same protocol. The four GraphAny models come from NodePFN's Table 1 and, on the five further graphs, from the GraphAny paper. TS-Mean (Finkelshtein et al., 2025)[4] and TAG (Hayler et al., 2025) come from their shared table, whose official splits match ours, although it has no Cora and uses the unfiltered Chameleon and Squirrel. GraphAny (Cora) and GraphAny (Wisconsin) are trained on Cora and Wisconsin, respectively.
- ¶ From Table 16 of Lee et al. (2026), which reproduces the GraphAny models, TS-Mean, TS-GAT and TAG on the filtered Chameleon and Squirrel under our splits, where the other copied cells come from the unfiltered graphs, and reports TAG on Cora, which TAG's table omits.
- § Cora is the training graph of TS-Mean, so its paper reports no held-out number there. The cell is our run of the authors' released code and WikiCS-trained model, as in their Table 2.
- * GraphPFN (Eremeev et al., 2026)[5] runs in its in-context mode on the raw feature matrix. Its tabular backbone gives each node one token per pair of features, so on the eight graphs with

[2] `https://github.com/DeepGraphLearning/GraphAny` (MIT).
[3] `https://github.com/yandex-research/heterophilous-graphs` (MIT).
[4] `https://github.com/benfinkelshtein/EquivarianceEverywhere` (MIT).
[5] `https://github.com/yandex-research/graphpfn` (Apache-2.0).

wide features (Amazon-computer, Coauthor-CS, Coauthor-physics, Questions, Wiki, BlogCatalog, LastFM Asia, FullCora) the raw run exceeds a 48 GB GPU, and these cells use PCA to 64 components, as its paper does for high-dimensional text features (its Appendix D).

- NodePFN (Choi et al., 2026)[6] is the released checkpoint under its released settings. Under our splits it averages 69.8 on the 23 graphs of its paper against the reported 71.2, with most of the gap on Air-Brazil (63.1 against 75.4), whose reported number comes from a different split.
- ◦ FullCora has 70 classes, more than the 20 class slots of the NodePFN and GADRPFN checkpoints. Both read it through random partitions of its classes into the 20 class slots.

Statistical comparison. Table 6 tests the difference between GADRPFN and each baseline in Table 1 with the two-sided Wilcoxon signed-rank test on the 28 paired accuracies and a Holm adjustment over the eight baselines. A Friedman test over the nine columns rejects equal ranks ($\chi^2 = 100.5$, $p < 10^{-17}$). In the critical difference diagram of Figure 6 (Demšar, 2006), a bar joins models that the Wilcoxon–Holm procedure used by Helli et al. (2024) does not separate at 0.05 over all 36 pairs, and no bar joins GADRPFN to another model.

Table 6: **Paired tests on Table 1.** Benchmarks on which GADRPFN outperforms, ties with or trails each baseline at the precision of Table 1, the mean gain in accuracy, and the Wilcoxon signed-rank p-value over the 28 paired accuracies with a Holm adjustment over the eight baselines.

Baseline	Benchmarks	Ahead / tied / behind	Mean gain	p (Holm)
TAG	28	25 / 1 / 2	+3.3	< 0.001
GraphPFN	28	23 / 1 / 4	+3.6	< 0.001
NodePFN	28	28 / 0 / 0	+7.1	< 0.001
GraphAny (Wisconsin)	28	28 / 0 / 0	+9.9	< 0.001
GraphAny (Products)	28	28 / 0 / 0	+10.0	< 0.001
TS-Mean	28	28 / 0 / 0	+10.0	< 0.001
GraphAny (Arxiv)	28	28 / 0 / 0	+10.1	< 0.001
GraphAny (Cora)	28	28 / 0 / 0	+10.5	< 0.001

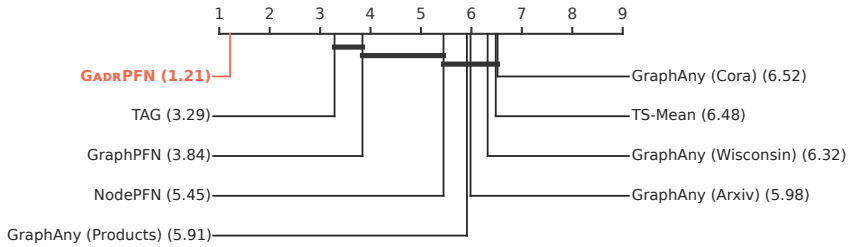


Figure 6: **Critical difference diagram** of the nine in-context columns of Table 1 over the 28 benchmarks. The axis is the average rank, and a bar joins models that the pairwise Wilcoxon test with a Holm adjustment over all pairs does not separate at 0.05.

D INFERENCE HYPERPARAMETERS OF TABLE 1

Table 7 lists, per dataset, the inference hyperparameters of the GADRPFN column of Table 1. The search covers the reduction (none, PCA or truncated SVD), widths of 5 to 300 components, 0 to 8 smoothing steps, 1 to 512 ensemble members and MLP-head weights from 0 to 1. With a single ensemble member and the other settings unchanged, the mean accuracy is 73.8.

[6] https://github.com/jeongwhanchoi/NodePFN (Apache-2.0).

Table 7: Inference hyperparameters of the GADRPFN column of Table 1. Width is the number of components kept by the reduction, Smoothing is the number of feature-smoothing steps on the observed graph, and MLP weight is the weight of the MLP head when the two classification heads are averaged.

Dataset	Reduction	Width	Smoothing	Ensemble	MLP weight
homo.					
Air-Brazil	none	–	0	32	0.75
Air-Europe	none	–	2	8	0.25
Air-USA	none	–	8	32	0.25
Cora	PCA	15	1	8	0.5
Citeseer	TSVD	15	0	8	0.5
Pubmed	TSVD	15	0	32	0.75
WikiCS	TSVD	25	0	32	0.75
Amazon-photo	TSVD	15	0	32	0.75
Amazon-comp.	TSVD	25	0	4	0.25
DBLP	TSVD	15	1	32	0.75
Coauthor-CS	TSVD	25	0	32	0.25
Coauthor-phys.	TSVD	25	1	4	0.75
Deezer	none	–	2	8	0.25
Wiki	TSVD	50	0	64	0.75
BlogCatalog	TSVD	15	0	8	0.5
LastFM Asia	TSVD	50	1	64	0.25
FullCora$^{\circ}$	TSVD	50	2	8	0.5
hetero.					
Cornell	TSVD	25	0	8	0.5
Texas	TSVD	20	0	32	0.75
Wisconsin	TSVD	25	0	32	0.25
Chameleon	TSVD	50	0	64	0.5
Actor	TSVD	16	0	120	0.14
Minesweeper	none	–	1	28	0.74
Tolokers	none	–	0	8	0
Amazon-ratings	TSVD	50	2	8	0.25
Questions	TSVD	50	0	8	0.75
Squirrel	TSVD	120	1	512	0
Roman-Empire	TSVD	50	0	32	0

E BASELINES OUTSIDE THE PROTOCOL OF TABLE 1

Graph foundation models. Table 1 includes the models that classify all 28 graphs from their labelled nodes, without node text and, except for GraphAny on its own training graph, without training on them. Graph foundation models pretrain on collections of real graphs. Some classify a new graph from its labelled nodes without tuning (Ma et al., 2026; Zhuo & Luo, 2026), some tune prompts at test time or pretrain across domains without text (Jiang et al., 2026; Yu et al., 2024), some align the feature spaces of different graphs (Eliasof et al., 2026; Lee & Yoo, 2026; Xia & Huang, 2026), and some process node text with a language model (Liu et al., 2024; Tang et al., 2024). Other work studies which architecture (Chen et al., 2025; Gutteridge et al., 2026), which geometry (Sun et al., 2025; 2026) or what scale (Bechler-Speicher et al., 2026) transfers across graphs, and how far pretraining transfers (Wang et al., 2025a;b; Li et al., 2026). These models are compared under shared protocols of their own (Platonov et al., 2026; Bazhenov et al., 2025), and Appendix F.2 compares GADRPFN with the methods that use node text.

Models trained on the target graph. Table 8 adds the supervised MLP, GCN and GAT of NodePFN's Table 1, RGVT (Lee et al., 2026), whose encoder is fixed but whose predictor and recurrent depth are fitted on each graph's training and validation nodes, and Sigil-nc (Yom-Tov & Gal, 2026), which fits an MLP on each target graph's training labels on top of a Cora-pretrained encoder of structural interaction graphs. GADRPFN trains on none of these graphs and has the best mean accuracy and average rank on both blocks.

Table 8: GADRPFN against models trained on the target graph. MLP, GCN and GAT are as reported in NodePFN's Table 1 (Choi et al., 2026), RGVT is from Lee et al. (2026) and Sigil-nc from Yom-Tov & Gal (2026), and the last two are blank where their papers do not report our version of a graph. Means and ranks are computed over the graphs reported in each column, and paired GADRPFN is the mean of GADRPFN on those graphs.

Dataset	MLP†	GCN†	GAT†	RGVT†	Sigil-nc†	GADRPFN
homo.						
Air-Brazil	23.1 ± 5.8	42.3 ± 8.0	57.7 ± 14.8	75.4 ± 6.4	39.2 ± 5.0	$\mathbf{80.8 \pm 7.7}$
Air-Europe	21.2 ± 2.3	41.9 ± 3.6	32.5 ± 8.4	57.1 ± 3.5	38.5 ± 6.4	$\mathbf{60.8 \pm 3.6}$
Air-USA	22.9 ± 1.5	46.5 ± 1.8	48.5 ± 4.2	57.5 ± 1.3	48.2 ± 3.4	$\mathbf{63.3 \pm 1.8}$
Cora	48.4 ± 0.6	81.4 ± 0.7	81.7 ± 1.4	81.7 ± 1.1	–	$\mathbf{82.2 \pm 0.2}$
Citeseer	44.4 ± 0.4	63.4 ± 0.6	69.1 ± 1.6	70.1 ± 1.5	53.7 ± 2.2	$\mathbf{70.8 \pm 0.4}$
Pubmed	69.5 ± 1.8	76.6 ± 0.3	77.3 ± 0.6	78.2 ± 0.8	77.6 ± 1.1	$\mathbf{80.8 \pm 0.3}$
WikiCS	72.7 ± 0.4	79.1 ± 0.5	79.3 ± 0.2	78.9 ± 0.6	79.5 ± 0.4	$\mathbf{80.8 \pm 0.4}$
Amazon-photo	68.2 ± 0.9	91.9 ± 0.8	91.9 ± 1.1	92.3 ± 0.4	84.1 ± 1.8	$\mathbf{92.8 \pm 0.6}$
Amazon-comp.	58.3 ± 3.0	85.8 ± 0.9	$\mathbf{87.0 \pm 0.5}$	84.1 ± 1.8	72.8 ± 2.6	86.9 ± 0.8
DBLP	56.3 ± 0.6	73.0 ± 2.2	73.9 ± 1.4	78.1 ± 1.2	75.1 ± 0.6	$\mathbf{78.9 \pm 1.2}$
Coauthor-CS	85.9 ± 0.9	91.8 ± 0.7	88.5 ± 0.8	$\mathbf{92.3 \pm 0.4}$	90.1 ± 0.4	$\mathbf{92.3 \pm 0.2}$
Coauthor-phys.	87.4 ± 2.0	$\mathbf{93.9 \pm 0.4}$	93.0 ± 0.9	93.2 ± 0.5	92.8 ± 0.4	93.1 ± 0.2
Deezer	54.2 ± 2.1	53.7 ± 2.3	56.0 ± 3.8	$\mathbf{56.2 \pm 1.4}$	–	55.2 ± 0.6
Wiki	63.8 ± 1.8	70.1 ± 1.5	59.6 ± 4.2	72.3 ± 4.7	68.4 ± 2.0	$\mathbf{75.0 \pm 1.2}$
BlogCatalog	64.1 ± 1.9	71.5 ± 2.6	62.0 ± 6.0	81.4 ± 2.8	68.8 ± 1.1	$\mathbf{86.4 \pm 0.4}$
LastFM Asia	57.4 ± 0.9	81.9 ± 1.1	84.7 ± 0.6	–	–	$\mathbf{85.9 \pm 0.4}$
FullCora°	33.5 ± 0.6	61.1 ± 0.2	59.0 ± 0.6	$\mathbf{63.3 \pm 0.8}$	57.1 ± 0.4	60.5 ± 0.3
hetero.						
Cornell	67.6 ± 5.1	35.1 ± 6.5	35.1 ± 3.5	74.0 ± 2.4	60.5 ± 4.5	$\mathbf{77.8 \pm 1.1}$
Texas	48.6 ± 4.0	51.4 ± 2.7	54.0 ± 2.4	78.9 ± 2.3	62.7 ± 6.5	$\mathbf{85.9 \pm 4.0}$
Wisconsin	66.7 ± 3.5	37.2 ± 1.6	52.9 ± 3.1	78.0 ± 4.5	71.8 ± 8.3	$\mathbf{89.8 \pm 3.8}$
Chameleon	38.9 ± 2.2	41.3 ± 3.0	39.8 ± 2.1	44.7 ± 2.2	–	$\mathbf{50.4 \pm 4.2}$
Actor	34.0 ± 0.8	28.6 ± 0.7	27.3 ± 0.2	32.7 ± 1.3	36.5 ± 0.5	$\mathbf{37.9 \pm 1.0}$
Minesweeper	80.0 ± 0.0	81.1 ± 0.4	80.1 ± 0.0	80.0 ± 0.0	85.0 ± 0.7	$\mathbf{85.8 \pm 0.2}$
Tolokers	78.2 ± 0.0	79.9 ± 0.1	78.5 ± 0.6	78.3 ± 0.3	80.7 ± 0.7	$\mathbf{82.4 \pm 0.6}$
Amazon-ratings	47.9 ± 0.5	47.4 ± 0.3	47.2 ± 0.4	45.6 ± 1.2	48.5 ± 2.4	$\mathbf{50.6 \pm 0.7}$
Questions	$\mathbf{97.3 \pm 0.1}$	97.2 ± 0.0	97.1 ± 0.0	97.1 ± 0.1	97.1 ± 0.1	$\mathbf{97.3 \pm 0.0}$
Squirrel	35.5 ± 1.0	38.7 ± 1.8	38.8 ± 2.4	41.8 ± 2.1	–	$\mathbf{48.2 \pm 2.6}$
Roman-Empire	65.8 ± 0.3	45.1 ± 0.4	43.9 ± 0.5	70.5 ± 0.7	$\mathbf{76.8 \pm 0.3}$	75.7 ± 0.5
Mean	56.85	63.89	64.15	71.62	68.07	**75.31**
Paired GADRPFN	75.31	75.31	75.31	74.91	77.68	–
Rank	5.04	3.82	4.02	2.72	3.57	**1.32**

Other models reported on some graphs. Table 9 compares the three PFNs with MeanGNN, which is trained on each graph, TS-GAT (Finkelshtein et al., 2025) and the Cora-trained zero-shot model GOBLIN (Gutteridge et al., 2026), and leaves a cell blank where a paper does not report our version of the graph. Over the graphs each of them reports, GADRPFN outperforms MeanGNN by 10.0% on average (25 graphs), TS-GAT by 9.8% (27) and GOBLIN by 9.1% (23).

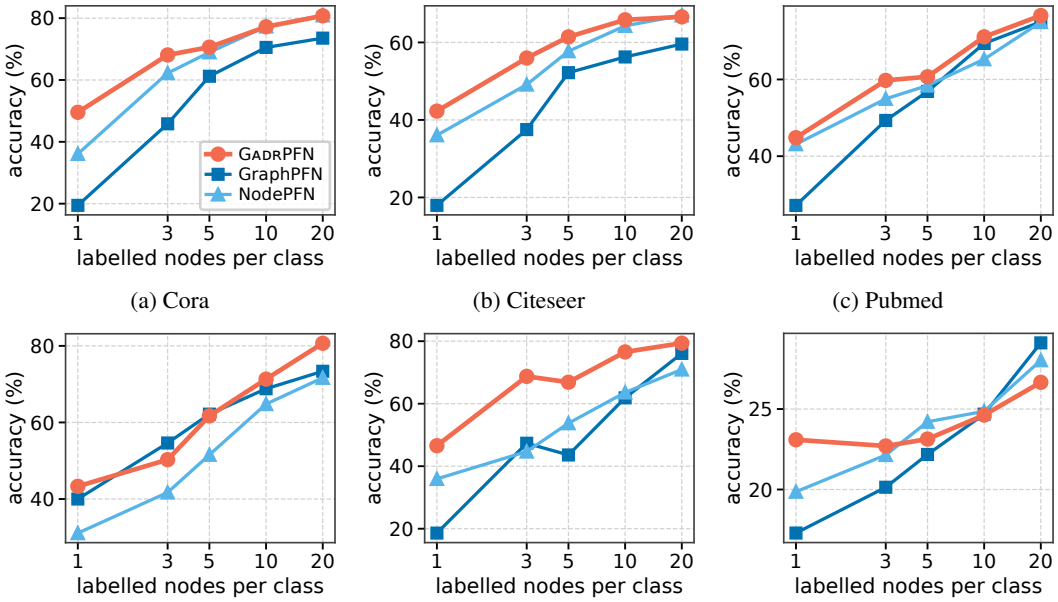
Table 9: The three PFNs against other models reported on some graphs. MeanGNN, which is trained on each graph, and TS-GAT are from Finkelshtein et al. (2025), the ¶ cells from Lee et al. (2026) and GOBLIN from Gutteridge et al. (2026), and cells are blank where our version of a graph is not reported. Means and ranks are computed over the graphs reported in each column, and paired GADRPFN is the mean of GADRPFN on those graphs.

Dataset	MeanGNN†	TS-GAT†	GOBLIN†	NodePFN	GraphPFN	GADRPFN
homo.						
Air-Brazil	32.3 ± 7.5	36.9 ± 10.0	50.8 ± 9.6	63.1 ± 6.2	73.8 ± 8.2	$\mathbf{80.8 \pm 7.7}$
Air-Europe	39.1 ± 6.4	38.4 ± 6.3	41.2 ± 4.0	53.0 ± 4.8	58.6 ± 2.9	$\mathbf{60.8 \pm 3.6}$
Air-USA	43.9 ± 1.2	41.7 ± 2.6	51.8 ± 4.4	60.6 ± 1.6	62.0 ± 0.8	$\mathbf{63.3 \pm 1.8}$
Cora	–	–	–	82.1 ± 0.3	76.1 ± 0.7	$\mathbf{82.2 \pm 0.2}$
Citeseer	65.1 ± 1.3	68.7 ± 0.5	65.3 ± 0.0	67.9 ± 0.3	62.3 ± 1.6	$\mathbf{70.8 \pm 0.4}$
Pubmed	74.6 ± 0.1	75.5 ± 0.9	74.7 ± 0.0	78.6 ± 0.3	78.0 ± 1.6	$\mathbf{80.8 \pm 0.3}$
WikiCS	72.0 ± 1.7	73.2 ± 2.5	63.4 ± 0.4	76.0 ± 0.8	78.6 ± 0.7	$\mathbf{80.8 \pm 0.4}$
Amazon-photo	89.0 ± 1.1	89.7 ± 1.2	88.9 ± 1.3	89.5 ± 0.8	89.5 ± 0.5	$\mathbf{92.8 \pm 0.6}$
Amazon-comp.	73.9 ± 0.9	80.2 ± 2.4	80.7 ± 2.4	79.5 ± 1.0	$84.1 \pm 0.9^*$	$\mathbf{86.9 \pm 0.8}$
DBLP	65.0 ± 2.4	64.3 ± 3.8	69.7 ± 2.7	76.5 ± 1.5	67.6 ± 0.7	$\mathbf{78.9 \pm 1.2}$
Coauthor-CS	80.9 ± 0.7	91.1 ± 0.4	89.6 ± 1.1	90.9 ± 0.5	$91.7 \pm 0.2^*$	$\mathbf{92.3 \pm 0.2}$
Coauthor-phys.	79.0 ± 1.1	92.2 ± 0.6	89.6 ± 1.6	92.3 ± 0.8	$91.5 \pm 1.0^*$	$\mathbf{93.1 \pm 0.2}$
Deezer	54.9 ± 1.8	52.9 ± 2.9	–	51.2 ± 1.8	53.7 ± 1.5	$\mathbf{55.2 \pm 0.6}$
Wiki	74.2 ± 0.9	72.5 ± 1.6	64.1 ± 3.8	66.4 ± 1.6	$\mathbf{79.5 \pm 1.1}^*$	75.0 ± 1.2
BlogCatalog	84.5 ± 0.7	78.4 ± 5.2	75.7 ± 2.6	63.4 ± 2.4	$82.6 \pm 0.7^*$	$\mathbf{86.4 \pm 0.4}$
LastFM Asia	77.2 ± 1.0	76.6 ± 1.4	–	77.9 ± 2.1	$79.2 \pm 1.4^*$	$\mathbf{85.9 \pm 0.4}$
FullCora°	55.9 ± 1.0	52.8 ± 0.9	55.3 ± 1.0	47.5 ± 3.6	$56.7 \pm 1.1^*$	$\mathbf{60.5 \pm 0.3}$
hetero.						
Cornell	63.8 ± 1.5	71.3 ± 4.5	67.6 ± 0.0	71.9 ± 2.8	73.0 ± 3.8	$\mathbf{77.8 \pm 1.1}$
Texas	76.8 ± 1.5	74.0 ± 4.1	78.4 ± 0.0	74.6 ± 6.5	69.5 ± 21.8	$\mathbf{85.9 \pm 4.0}$
Wisconsin	74.1 ± 12.2	65.9 ± 9.9	82.3 ± 0.0	80.0 ± 6.1	82.7 ± 4.7	$\mathbf{89.8 \pm 3.8}$
Chameleon	–	36.4 ± 8.3¶	–	41.2 ± 2.4	46.7 ± 2.5	$\mathbf{50.4 \pm 4.2}$
Actor	32.0 ± 0.3	28.8 ± 2.1	25.4 ± 0.1	33.2 ± 0.3	$\mathbf{38.0 \pm 1.3}$	37.9 ± 1.0
Minesweeper	84.1 ± 0.2	80.7 ± 0.8	81.0 ± 0.0	80.7 ± 0.3	83.6 ± 0.5	$\mathbf{85.8 \pm 0.2}$
Tolokers	78.6 ± 0.7	78.0 ± 0.2	78.7 ± 0.0	78.6 ± 0.2	$\mathbf{83.9 \pm 0.6}$	82.4 ± 0.6
Amazon-ratings	40.7 ± 0.1	41.9 ± 0.5	44.4 ± 0.0	44.4 ± 0.4	48.7 ± 0.4	$\mathbf{50.6 \pm 0.7}$
Questions	97.2 ± 0.1	97.0 ± 0.0	96.3 ± 0.0	97.0 ± 0.0	$\mathbf{97.3 \pm 0.0}^*$	$\mathbf{97.3 \pm 0.0}$
Squirrel	–	35.8 ± 3.3¶	–	43.2 ± 1.0	$\mathbf{48.7 \pm 1.3}$	48.2 ± 2.6
Roman-Empire	69.4 ± 0.7	67.8 ± 0.6	62.5 ± 0.0	47.7 ± 1.8	69.6 ± 0.6	$\mathbf{75.7 \pm 0.5}$
Mean	67.12	65.29	68.58	68.17	71.69	**75.31**
Paired GADRPFN	77.11	75.05	77.68	75.31	75.31	–
Rank	4.42	4.44	4.41	3.73	2.50	**1.16**

F FURTHER COMPARISONS

Each subsection follows the protocol of one benchmark, copies its baseline numbers (marked †) and adds our runs of the three PFNs. NodePFN is the released checkpoint and GADRPFN the model of Table 1.

F.1 FEW-SHOT NODE CLASSIFICATION UNDER THE PROG PROTOCOL



(a) Cora (b) Citeseer (c) Pubmed

(d) Wisconsin (e) Texas (f) Actor

Figure 7: **Accuracy against labelled nodes per class on each graph.** Mean over five draws under the ProG split protocol for the three models that classify without training on the target graph. Figure 3a is their mean over the six graphs.

Protocol. The ProG benchmark (Zi et al., 2024)[7] evaluates its graph-prompt family with $k \in \{1, 3, 5\}$ labelled nodes per class. A random 90% of the nodes forms the test set (70% on Wisconsin and Texas, which ProG's code uses when 90% of the nodes is fewer than $1{,}000$), there is no validation set, and results are averaged over five draws. We draw the training nodes and the test nodes disjointly and add $k \in \{10, 20\}$.

Baselines. Table 10 copies ProG's node-classification results on the three citation graphs (its Tables 2, 7 and 9). Supervised is a GCN trained on the k shots, Pre-train & Fine-tune is a self-supervised GNN fine-tuned on them, and the other rows are its prompt-tuning methods. ProG reports Pre-train & Fine-tune and each prompt method as the best of six pretraining variants. All but Supervised are pretrained on the target graph, and all are fitted on the k shots by gradient descent, whereas the three PFNs receive the shots as context rows. We do not copy ProG's values on Wisconsin, Texas, Actor and ogbn-arxiv, where several cells exceed the highest fully supervised accuracies reported for these graphs.

Result. With one labelled node per class GADRPFN outperforms NodePFN and GraphPFN on all six graphs (Table 11) and the supervised GCN on all three citation graphs (Table 10).

[7] `https://github.com/sheldonresearch/ProG` (MIT).

Table 10: Few-shot node classification on the citation graphs under the ProG protocol (Zi et al., 2024). Rows marked † are copied from ProG's Tables 2, 7 and 9, where each cell except Supervised is the best of six pretraining variants. The PFN rows are our runs over five draws without training on the target graph. Bold marks the best per column and shot and underline the runner-up.

Method	Cora	Citeseer	Pubmed
1-shot			
Supervised†	26.6 ± 5.5	21.8 ± 7.3	39.4 ± 16.3
Pre-train & Fine-tune†	52.6 ± 1.7	35.0 ± 4.4	46.7 ± 14.9
GPPT†	43.1 ± 9.4	37.3 ± 6.2	$\underline{48.3 \pm 17.7}$
All-in-one†	52.4 ± 10.2	40.4 ± 2.8	45.2 ± 6.5
Gprompt†	$\mathbf{56.7 \pm 11.2}$	$\underline{53.2 \pm 10.9}$	39.7 ± 15.3
GPF†	38.6 ± 5.4	31.2 ± 8.1	$\mathbf{50.0 \pm 8.9}$
GPF-plus†	$\underline{55.8 \pm 10.3}$	$\mathbf{59.7 \pm 11.9}$	46.6 ± 19.0
NodePFN	36.1 ± 2.9	36.1 ± 5.4	43.1 ± 10.2
GraphPFN	19.5 ± 6.6	18.0 ± 6.0	27.2 ± 6.7
GADRPFN (ours)	49.6 ± 5.7	42.3 ± 6.4	44.8 ± 14.1
3-shot			
Supervised†	37.8 ± 9.2	35.2 ± 6.9	57.3 ± 4.6
Pre-train & Fine-tune†	52.0 ± 2.8	45.1 ± 2.1	65.4 ± 3.0
GPPT†	43.8 ± 6.1	42.3 ± 8.3	67.4 ± 3.0
All-in-one†	48.1 ± 4.8	48.1 ± 8.2	65.8 ± 5.8
Gprompt†	$\underline{63.8 \pm 5.8}$	$\underline{60.0 \pm 6.2}$	66.7 ± 3.5
GPF†	34.8 ± 19.8	25.9 ± 12.3	$\mathbf{71.2 \pm 2.8}$
GPF-plus†	56.4 ± 5.4	$\mathbf{72.5 \pm 5.6}$	$\underline{70.8 \pm 4.0}$
NodePFN	62.2 ± 5.3	49.1 ± 5.5	54.9 ± 7.2
GraphPFN	45.8 ± 5.8	37.5 ± 6.3	49.3 ± 11.5
GADRPFN (ours)	$\mathbf{68.1 \pm 3.3}$	56.0 ± 4.2	59.8 ± 8.1
5-shot			
Supervised†	50.2 ± 8.4	41.2 ± 6.3	67.9 ± 2.2
Pre-train & Fine-tune†	62.7 ± 3.5	39.5 ± 3.5	$\mathbf{70.9 \pm 4.9}$
GPPT†	52.0 ± 3.4	45.8 ± 7.4	67.0 ± 3.7
All-in-one†	30.4 ± 13.5	27.9 ± 10.6	46.2 ± 15.8
Gprompt†	$\underline{69.0 \pm 3.6}$	$\underline{66.1 \pm 1.6}$	67.9 ± 2.1
GPF†	35.4 ± 1.0	25.1 ± 3.0	69.0 ± 4.0
GPF-plus†	63.3 ± 4.7	$\mathbf{75.7 \pm 2.2}$	$\underline{69.6 \pm 4.3}$
NodePFN	68.9 ± 4.7	57.7 ± 3.5	58.4 ± 6.4
GraphPFN	61.2 ± 6.6	52.2 ± 4.9	56.9 ± 2.2
GADRPFN (ours)	$\mathbf{70.6 \pm 5.2}$	61.4 ± 1.6	60.7 ± 7.0

Table 11: The three PFNs under the ProG protocol on six graphs with 1 to 20 labelled nodes per class. Each cell is the mean over five draws. Bold marks the best per column and shot and underline the runner-up.

Method	Cora	Citeseer	Pubmed	Wisconsin	Texas	Actor
1-shot						
NodePFN	36.1 ± 2.9	36.1 ± 5.4	43.1 ± 10.2	31.1 ± 7.6	35.9 ± 14.9	19.9 ± 1.8
GraphPFN	19.5 ± 6.6	18.0 ± 6.0	27.2 ± 6.7	40.0 ± 17.7	18.6 ± 8.2	17.3 ± 4.1
GADRPFN (ours)	$\mathbf{49.6 \pm 5.7}$	$\mathbf{42.3 \pm 6.4}$	$\mathbf{44.8 \pm 14.1}$	$\mathbf{43.3 \pm 14.0}$	$\mathbf{46.6 \pm 13.7}$	$\mathbf{23.1 \pm 1.5}$
3-shot						
NodePFN	62.2 ± 5.3	49.1 ± 5.5	54.9 ± 7.2	41.7 ± 8.3	44.7 ± 9.0	22.1 ± 2.1
GraphPFN	45.8 ± 5.8	37.5 ± 6.3	49.3 ± 11.5	$\mathbf{54.6 \pm 11.3}$	47.3 ± 17.3	20.1 ± 1.7
GADRPFN (ours)	$\mathbf{68.1 \pm 3.3}$	$\mathbf{56.0 \pm 4.2}$	$\mathbf{59.8 \pm 8.1}$	50.3 ± 10.9	$\mathbf{68.8 \pm 4.3}$	$\mathbf{22.7 \pm 1.1}$
5-shot						
NodePFN	68.9 ± 4.7	57.7 ± 3.5	58.4 ± 6.4	51.5 ± 7.3	53.8 ± 6.4	$\mathbf{24.2 \pm 1.7}$
GraphPFN	61.2 ± 6.6	52.2 ± 4.9	56.9 ± 2.2	$\mathbf{62.2 \pm 3.7}$	43.6 ± 23.2	22.2 ± 1.0
GADRPFN (ours)	$\mathbf{70.6 \pm 5.2}$	$\mathbf{61.4 \pm 1.6}$	$\mathbf{60.7 \pm 7.0}$	61.7 ± 2.8	$\mathbf{66.9 \pm 4.7}$	23.1 ± 2.6
10-shot						
NodePFN	$\mathbf{77.3 \pm 0.9}$	64.3 ± 1.3	65.3 ± 5.4	64.8 ± 3.9	63.6 ± 7.1	$\mathbf{24.9 \pm 1.4}$
GraphPFN	70.5 ± 2.2	56.3 ± 5.2	69.4 ± 2.1	68.8 ± 5.4	61.9 ± 20.0	24.7 ± 2.2
GADRPFN (ours)	77.2 ± 0.7	$\mathbf{65.8 \pm 1.6}$	$\mathbf{71.1 \pm 6.0}$	$\mathbf{71.3 \pm 5.0}$	$\mathbf{76.6 \pm 6.0}$	24.6 ± 2.1
20-shot						
NodePFN	$\mathbf{80.9 \pm 1.7}$	$\mathbf{67.0 \pm 0.4}$	74.9 ± 2.0	71.7 ± 4.6	70.9 ± 7.1	28.0 ± 2.2
GraphPFN	73.5 ± 2.9	59.6 ± 2.1	75.0 ± 2.2	73.4 ± 2.9	76.1 ± 3.3	$\mathbf{29.1 \pm 1.5}$
GADRPFN (ours)	80.8 ± 1.2	66.6 ± 0.7	$\mathbf{76.7 \pm 2.1}$	$\mathbf{80.7 \pm 2.8}$	$\mathbf{79.4 \pm 4.5}$	26.7 ± 1.5

F.2 TEXT-ATTRIBUTED GRAPHS

Protocol. GLBench (Li et al., 2024)[8] evaluates methods that process the node texts with a language model. Table 12 copies the accuracy columns of its Table 4 and adds our own runs of NodePFN, GraphPFN and GADRPFN on the same files and masks. The supervised split of Cora, Citeseer and Pubmed is the same twenty labelled nodes per class as ours, with 500 validation nodes and the rest as test, WikiCS uses its official split, and Reddit and Instagram use 10% of the nodes for training.

Features. GADRPFN encodes the node texts with the frozen sentence encoder bge-base (110M parameters), whose 768-dimensional CLS vector is the node's feature vector. Only this vector is given to the network, which does not see the text and is not fine-tuned, and the encoder is far smaller than the LLaMA2-7B model that six of the ten LLM-based methods run, while GIANT, OFA and GLEM use the 66M-parameter Sentence-BERT and PATTON the 110M-parameter BERT (GLBench's Table 6). The GNN rows of GLBench use GLBench's 128-dimensional BERT-tiny vectors, and Table 12 runs the three PFNs on both feature sets.

Arxiv. Arxiv (169k nodes) exceeds the checkpoint in both ways of Appendix F.3, since it has 40 classes and a 91k-node labelled set, and its cell is marked °. GADRPFN uses all 91k labelled nodes as context, just as the supervised rows train on them. NodePFN runs out of memory beyond 32,768 context rows and uses a class-proportional sample of that size, and the table gives GADRPFN at that context in parentheses beside its full-context cell. GraphPFN processes the whole graph at once and runs out of memory on Arxiv.

Result. Among the methods of Table 12, GADRPFN is the most accurate on Cora, Citeseer, WikiCS and Reddit and the runner-up on Pubmed. On Arxiv it is within 1% of GCN, GAT and GraphSAGE trained on the same 91k labels. With the BERT-tiny vectors that the GNN rows use, GADRPFN averages below GCN and GAT, so its lead over the GNN rows relies on the stronger encoder, whereas its lead over NodePFN and GraphPFN holds with either feature set.

[8] `https://github.com/NineAbyss/GLBench` (MIT).

Table 12: Accuracy under the supervised setting of GLBench (Li et al., 2024) (its Table 4) with our runs of NodePFN, GraphPFN and GADRPFN on the same masks. ° marks Arxiv, decoded through random 20-slot class partitions with all its 91k labelled nodes as context for GADRPFN and 32,768 of them for NodePFN. The value in parentheses is GADRPFN with the same 32,768 context rows as NodePFN. Avg. and Rank are computed over the rows complete on all seven graphs. Bold marks the best per column and underline the runner-up.

Method	Cora	Citeseer	Pubmed	Arxiv	WikiCS	Reddit	Instagram	Avg.	Rank
GNN									
GCN	<u>82.1</u>	69.8	79.1	72.2	80.3	63.2	65.8	73.2	6.6
GAT	80.3	68.8	76.9	71.8	79.7	62.0	65.4	72.1	9.7
GraphSAGE	79.9	68.2	76.8	71.9	79.9	58.5	65.1	71.5	11.2
PLM									
Sent-BERT (22M)	69.7	68.4	65.9	72.8	77.1	57.3	63.1	67.8	15.5
BERT (110M)	69.7	67.8	63.7	72.3	78.5	58.4	63.8	67.7	15.2
RoBERTa (355M)	69.7	68.2	71.2	72.9	78.7	57.2	63.6	68.8	14.6
LLM as enhancer									
GIANT	81.0	65.8	76.9	72.0	80.5	64.7	66.0	72.4	8.6
TAPE	81.0	66.1	79.9	73.0	82.3	60.7	65.8	72.7	7.7
OFA	75.2	<u>73.0</u>	75.6	73.2	77.3	64.9	60.9	71.5	10.3
ENGINE	81.5	72.2	74.7	<u>75.0</u>	81.2	63.2	**67.6**	73.6	5.1
LLM as predictor									
InstructGLM	69.1	51.9	71.3	39.1	45.7	55.8	57.9	55.8	20.3
GraphText	76.2	59.4	74.6	49.5	67.3	61.9	62.6	64.5	16.3
GraphAdapter	72.8	69.6	72.8	74.5	70.8	61.2	67.4	69.9	10.3
LLaGA	74.4	55.7	52.5	72.8	73.9	67.2	62.9	65.6	14.6
LLM as aligner									
GLEM-GNN	<u>82.1</u>	71.2	81.7	**76.4**	<u>82.4</u>	59.6	66.1	<u>74.2</u>	<u>4.4</u>
GLEM-LLM	73.8	68.8	79.2	74.0	80.2	58.0	65.0	71.3	9.9
PATTON	70.5	63.6	**84.3**	70.7	80.8	59.4	64.3	70.5	11.9
PFNs with BERT-tiny features									
NodePFN	77.6	62.4	68.6	57.2°	76.3	66.5	64.3	67.6	14.4
GraphPFN	69.3	61.8	58.5	N/A	77.7	69.7	65.7	–	–
GADRPFN	79.1	66.8	71.6	69.9° (68.5)	80.0	<u>71.3</u>	65.6	72.0	10.7
PFNs with bge-base features									
NodePFN	81.7	69.4	76.6	60.8°	78.6	66.7	63.7	71.1	10.2
GraphPFN	78.5	72.3	73.1	N/A	81.9	71.1	<u>67.5</u>	–	–
GADRPFN	**82.4**	**73.3**	<u>81.8</u>	71.5° (70.8)	**83.1**	**71.4**	67.1	**75.8**	**3.4**

F.3 LARGE-SCALE NODE CLASSIFICATION

Protocol. ogbn-arxiv has 169k nodes, 40 classes and, under the OGB split that GraphAny's benchmark adopts, 91k labelled nodes, which the in-context models use as their context. Table 13 copies the models that classify the graph without training on it, namely the GraphAny models trained on other graphs (Zhao et al., 2025, Table 5), TS-Mean and TS-GAT (Finkelshtein et al., 2025, Table 1) and TAG (Hayler et al., 2025, Table 1). It adds the three PFNs, with GADRPFN and NodePFN under the partition decoding below and GraphPFN under its own. GraphPFN needs at least five ensemble members for 40 classes, and exceeds a 48 GB GPU with a single one, since each member processes the 169k nodes at once. The lower block gives the models trained on the graph's 91k labels, namely GraphAny trained on ogbn-arxiv, the GCN and GAT of GraphAny's table, MeanGNN, which TS-GNN trains on each graph, and the RGVT and Sigil-nc encoders with an MLP fitted on the target graph (Lee et al., 2026; Yom-Tov & Gal, 2026). RGVT is itself pretrained on ogbn-arxiv.

Settings. The graph exceeds the pretraining tasks in two ways. Its 40 classes are decoded through random partitions into the 20 class slots, as TabPFN and TAG do above their class limits. The whole 91k-node context is used in full (Appendix G.2). The features are reduced to 25 components and smoothed for two steps, with an ensemble of eight, and NodePFN uses the same decoding.

Table 13: ogbn-arxiv under GraphAny's protocol (OGB split, 91k labelled nodes as context). The upper two blocks classify the graph without training on it and are ranked together, and the lower block trains on it. † marks copied numbers. Bold marks the best and underline the runner-up.

Method	The 91k labels are its	Accuracy
No training on the graph		
GraphAny (Products)†	context	58.6 ± 0.1
GraphAny (Wisconsin)†	context	57.8 ± 0.6
GraphAny (Cora)†	context	58.6 ± 0.1
TS-Mean†	context	56.3 ± 2.6
TS-GAT†	context	53.1 ± 2.0
TAG†	context	$\underline{66.7 \pm 0.2}$
PFNs, 40 *classes decoded through sixteen* 20*-slot partitions*		
GadrPFN (ours)	context	$\mathbf{71.0 \pm 0.1}$
NodePFN	context	64.5 ± 2.9
GraphPFN	context	out of memory (48 GB)
Trained on the graph (reference)		
GraphAny (Arxiv)†	training set	58.7 ± 0.2
MeanGNN†	training set	50.9 ± 0.4
GCN†	training set	71.7 ± 0.3
GAT†	training set	73.7 ± 0.1
RGVT†	training set	70.3 ± 0.3
Sigil-nc†	training set	71.9 ± 0.1

G Cost and scaling

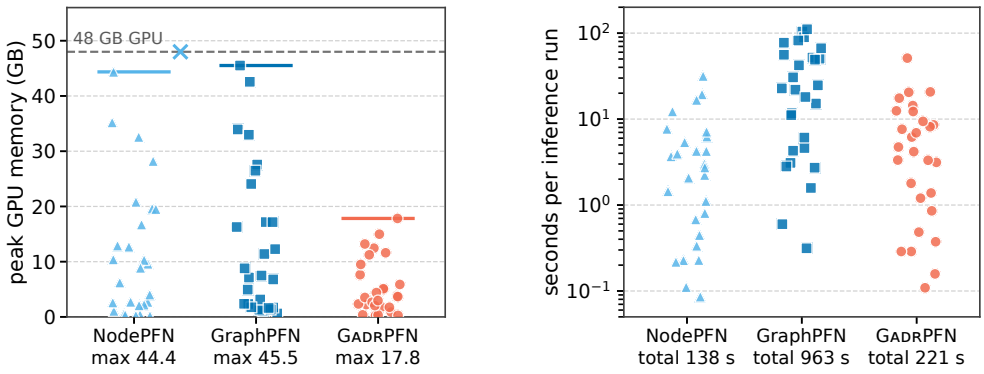
This appendix reports the inference cost and how accuracy changes with the labelled context and the amount of pretraining.

G.1 Inference cost

Setup. Table 14 and Figure 8 report, for the three PFNs of Table 1, the time and the peak memory of one inference run on each of the 28 benchmarks. NodePFN and GadrPFN run their checkpoints of Table 1 under NodePFN's released settings and GraphPFN under its own, all on one 48 GB A6000. A run comprises the graph normalisation, the feature smoothing and TSVD (GraphPFN uses its own preprocessing), the fit and the forward passes of the ensemble over every query node, excluding checkpoint loading. The time is the mean over five runs, and the memory is PyTorch's peak allocation over them, including the weights. None of the three trains on the graph, so this is their whole cost.

Model sizes. The three models are of similar size. NodePFN has 29.0 M parameters, GraphPFN 20.1 M (a frozen LimiX backbone of 16.5 M with 3.6 M of graph adapters and MLPs) and GadrPFN 32.5 M.

Result. GadrPFN processes all 28 graphs on one GPU with at most 17.8 GB, while NodePFN and GraphPFN exceed 24 GB on four and seven graphs, respectively, and NodePFN's released model, which builds dense per-graph operators, exceeds the GPU on WikiCS, where its cell of Table 1 was computed on CPU (Figure 8a). Over the 28 graphs it requires 221 seconds, against 138 for NodePFN and 963 for GraphPFN, and it is faster than GraphPFN on 24 of them (Figure 8b). GraphPFN takes the raw feature matrix as input, so on the graphs marked * it runs with the PCA-64 preprocessing that its paper uses, as in Table 1.



(a) Peak GPU memory.

(b) Time per inference run.

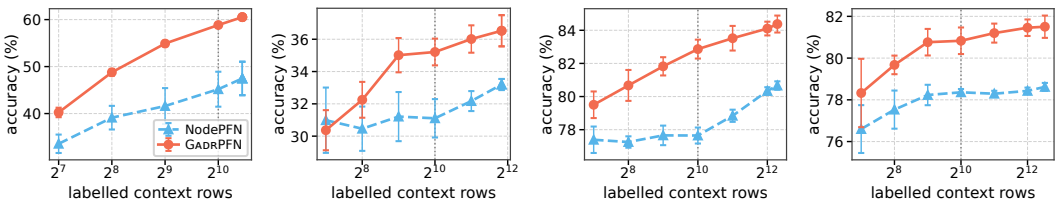
Figure 8: **Inference cost per benchmark.** One dot per benchmark of Table 14 for each PFN. (a) The bar marks the largest peak memory, which is printed under the name. The dashed line is the memory of the GPU, and the cross marks NodePFN on WikiCS, which exceeds it. (b) The total over the 28 graphs is printed under the name.

Table 14: Inference cost on the benchmarks of Table 1 under NodePFN's released settings. Time is in seconds per inference run and peak GPU memory in GB on one 48 GB A6000. Context is the number of labelled nodes per split. * marks GraphPFN with PCA-64 (raw features exceed the GPU), ° the 70 classes decoded through class partitions, and "–" a run that exceeds the GPU. The last row sums the times and takes the maximum memory over the graphs a model runs on.

Dataset	Nodes	Edges	Context	NodePFN		GraphPFN		GADRPFN	
				Time (s)	VRAM (GB)	Time (s)	VRAM (GB)	Time (s)	VRAM (GB)
Air-Brazil	131	1,003	80	0.2	0.4	0.3	0.6	0.4	0.4
Air-Europe	399	5,993	80	0.2	0.5	0.6	0.7	0.3	0.5
Air-USA	1,190	13,599	80	0.3	1.1	1.6	1.1	0.5	0.9
Cora	2,708	5,278	140	0.7	2.6	22.8	11.4	1.2	2.2
Citeseer	3,327	4,552	120	0.8	2.7	51.9	24.1	1.4	2.3
Pubmed	19,717	44,324	60	3.7	19.6	104	45.5	7.6	12.4
WikiCS	11,701	215,603	580	–	–	50.4	17.2	14.4	13.2
Amazon-photo	7,650	119,081	160	4.2	32.6	77.3	27.6	8.6	7.6
Amazon-comp.	13,752	245,861	200	2.2	2.2	15.2*	4.9	3.3	1.8
DBLP	17,716	52,867	80	4.2	20.8	56.0	26.5	8.1	11.2
Coauthor-CS	18,333	81,894	300	7.7	28.2	22.0*	6.8	12.2	11.6
Coauthor-phys.	34,493	247,962	100	19.4	9.6	42.2*	12.3	20.7	3.5
Deezer	28,281	92,752	40	6.3	35.2	66.6	42.6	12.5	17.8
Wiki	2,405	11,596	340	1.1	4.1	4.6*	1.3	1.8	2.7
BlogCatalog	5,196	171,743	120	12.3	44.4	11.7*	2.3	17.5	9.5
LastFM Asia	7,624	27,806	360	7.1	10.3	11.1*	3.2	9.4	5.0
FullCora°	19,793	63,421	1,400	31.7	6.2	24.7*	7.1	51.2	3.7
Cornell	183	277	87	0.1	0.2	3.1	1.7	0.1	0.2
Texas	183	279	87	0.1	0.3	2.8	1.7	0.2	0.3
Wisconsin	251	450	120	0.2	0.4	4.3	2.4	0.3	0.3
Chameleon	890	8,854	440	0.4	2.7	18.1	7.5	0.9	2.0
Actor	7,600	26,659	3,648	3.0	10.4	89.6	33.9	4.7	5.1
Minesweeper	10,000	39,402	5,000	4.0	12.9	2.7	1.3	6.2	5.9
Tolokers	11,758	519,000	5,879	2.1	16.7	6.1	1.6	4.2	3.7
Amazon-ratings	24,492	93,050	12,246	5.4	8.9	30.6*	8.8	6.9	4.4
Questions	48,921	153,540	24,460	2.7	2.1	82.1*	17.2	3.1	1.8
Squirrel	2,223	46,998	1,055	1.4	12.7	49.1	16.3	3.3	3.0
Roman-Empire	22,662	32,927	11,331	16.7	19.5	111	33.0	20.5	15.0
All 28 graphs, total / max				138	44.4	963	45.5	221	17.8

G.2 BEYOND THE PRETRAINING CONTEXT LENGTH

Setup. Every pretraining task has at most 1,024 nodes, so a context of 1,024 labelled rows is the longest the network has seen. Eight benchmarks of Table 1 exceed it, from Squirrel (1,055 labelled rows per split) and FullCora (1,400) to Roman-Empire (11,331), Amazon-ratings (12,246) and Questions (24,460). Figure 9 caps the context of these graphs at 128 to 4,096 rows, drawn



(a) FullCora (b) Actor (c) Minesweeper (d) Tolokers

(e) Amazon-ratings (f) Questions (g) Squirrel (h) Roman-Empire

Figure 9: **Accuracy against the number of labelled context rows.** The eight benchmarks whose labelled set exceeds the pretraining task size, with the mean and standard deviation over five runs under NodePFN's released settings. The dotted line is the pretraining task size of 1,024 nodes.

class-proportionally per split, and at the full labelled set, for GADRPFN and the released NodePFN under NodePFN's released settings.

Result. Accuracy does not fall beyond the pretraining length on any graph, and for GADRPFN it rises faster than for NodePFN on graphs whose structure carries information about the labels. From 1,024 rows to the full labelled set GADRPFN gains 9.6% on Roman-Empire, 7.8% on Amazon-ratings and 1.5% on Minesweeper, against 4.5%, 5.9% and 3.0% for NodePFN. Its lead over NodePFN widens with the context, from 8% at 128 rows to 28% at the full set on Roman-Empire and from 7% to 13% on FullCora. The gains of Table 1 on the graphs with many labels are obtained with contexts longer than any seen in pretraining.

G.3 PRETRAINING LENGTH

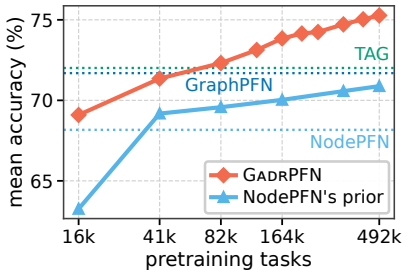


Figure 10: **Accuracy over pretraining.** The network of Table 1 on each prior, at the saved checkpoints of one run per prior. Mean accuracy on the 28 benchmarks under the settings selected for GADRPFN in Table 1, with its baselines dotted.

Setup. The run behind Table 1 pretrains on 491,520 synthetic tasks. Figure 10 evaluates every saved checkpoint of that run on the 28 benchmarks under the per-dataset settings of its Table 1 column and plots the accuracy against the number of tasks seen, so that the last checkpoint gives that table. To separate the prior from the amount of pretraining, the same network is pretrained on NodePFN's prior with the same recipe (the run behind the NodePFN's-prior row of Table 2) and evaluated under the same settings at six of its checkpoints.

Result. Accuracy rises with the number of tasks throughout the run and has not flattened at 491,520, where the mean is 75.3. After 16,384 tasks (a thirtieth of the run) the model reaches 69.1 and passes the released NodePFN, and after 81,920 it reaches 72.3 and passes GraphPFN and TAG (the strongest baseline of Table 1) with a sixth of the pretraining that the reported model uses. The gap between the two priors doubles over the run (§4.3). On the heterophilous block, after 40,960 tasks the network on NodePFN's prior gains 0.5% and GADRPFN gains 5.3%. §4.5 shows what changes in the network over the same stretch.

H WHAT EACH PRIOR GENERATES

Two statistics. Figure 4(a) positions each benchmark by two statistics computed on its full graph. We measure the feature smoothness of Hou et al. (2020) by $1 - \mathcal{E}(X)$ instead of their λ_f, with $\mathcal{E}(X) = \mathrm{tr}(X^\top \bar{L} X)/\mathrm{tr}(X^\top X)$, which is the Dirichlet energy of the standardised features on the normalised Laplacian $\bar{L}$ without self loops. Adjusted label homophily (Platonov et al., 2023a) is $h_{\mathrm{adj}} = (h - \sum_k \bar{p}_k^2)/(1 - \sum_k \bar{p}_k^2)$, where h is the edge homophily of y on the undirected graph without self loops and $\bar{p}_k$ the share of edge endpoints in class k. It is 0 for random labels regardless of the number and the sizes of the classes and 1 for perfect agreement. The region of each prior in Figure 4(a) is the convex hull of its 1,000 synthetic graphs after the 0.5% tails of each coordinate are removed.

Six statistics. Figure 11 and the lower block of Table 15 describe a graph by six statistics of the graph with its features and labels. They are the edge label homophily, the feature smoothness above, the logarithm of the mean degree, the number of classes, the logarithm of the class imbalance (largest over smallest class) and the feature-structure coupling, which is the mean over feature columns of $|\mathrm{corr}(x, \hat{A}x)|$. Each is standardised by its mean and standard deviation over the 28 benchmarks. Every prior contributes its first 1,000 synthetic graphs. Ours and NodePFN's have 256 to 1,024 nodes as in pretraining. GraphPFN's graphs are sampled with the released prior code and pretraining configuration of GraphPFN, restricted to its classification tasks, and have about 1,000 to 5,000 nodes and a mean degree of up to a few hundred, as in its pretraining.

Measures. Table 15 adapts the global and local measures of Xie et al. (2026, Appendix D) to graphs, with one whole graph and its features and labels as one sample. The distribution gap is the sliced Wasserstein distance SWD_2 between the synthetic graphs and the 28 benchmarks, which is the square root of the mean squared 2-Wasserstein distance between their projections onto 512 random unit directions. The nearest-graph distance is the distance from each benchmark to its closest synthetic graph, averaged over each block. It replaces CauKer's class-wise nearest-neighbour agreement, which requires labels shared by synthetic and real samples. A benchmark is covered when its nearest synthetic graph is closer than its nearest other benchmark, which is the coverage of Naeem et al. (2020) with one neighbour. The table averages five disjoint draws of 200 synthetic graphs, so its coverage counts lie below those of the 1,000 draws in Figure 11.

Counts behind §4.4. On the two statistics, the region of NodePFN's prior contains 8 of the 28 benchmarks, that of GraphPFN's prior 14 and that of the GADRPFN prior all 28. The graphs of NodePFN's prior lie near the vertical axis and fall below $h_{\mathrm{adj}} = 0$ only through sampling noise. Every benchmark lies within 0.07 of a synthetic graph of the GADRPFN prior, against up to 0.27 for NodePFN's and GraphPFN's, whose draws lie farthest from the coauthorship and co-purchase graphs, LastFM Asia and the WebKB graphs. On the six statistics the GADRPFN prior covers 16 of the 17 homophilous benchmarks and 27 of all 28, against 6 and 15 for NodePFN's prior and 6 and 16 for GraphPFN's (Figure 11). Most homophilous benchmarks, whose features are smooth and whose labels follow the graph, lie outside the draws of each existing prior.

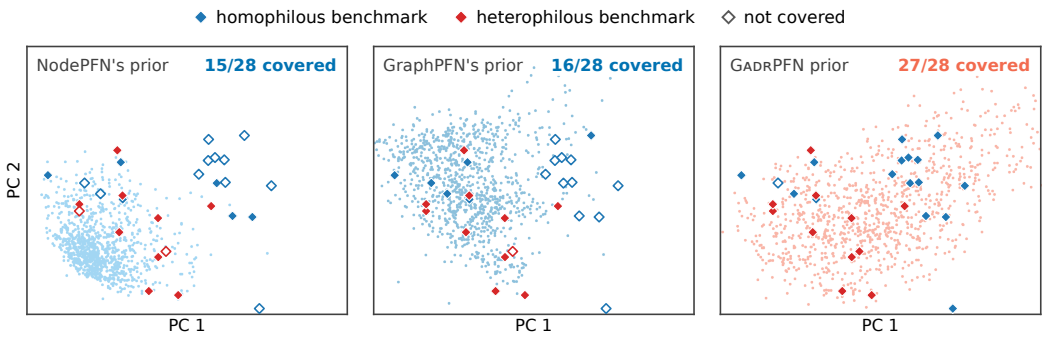


Figure 11: **Coverage of the benchmarks by each prior.** Two principal components of the six statistics, with 1,000 synthetic graphs per prior. The right panel is Figure 4(b). A benchmark is filled when a synthetic graph lies closer to it than any other benchmark.

Table 15: **How close the synthetic graphs of each prior are to the 28 benchmarks.** Distances in the space of graph statistics, with the best prior in bold.

	NodePFN's prior	GraphPFN's prior	GADRPFN prior
Two statistics of Figure 4			
Distribution gap: SWD_2 (↓)	0.269 ±0.008	0.307 ±0.009	**0.136 ±0.004**
Nearest synthetic graph, homophilous (↓)	0.151 ±0.037	0.188 ±0.072	**0.040 ±0.006**
Nearest synthetic graph, heterophilous (↓)	0.085 ±0.003	0.054 ±0.005	**0.033 ±0.005**
Coverage: benchmarks covered (↑)	10.4 ±0.8	12.0 ±1.4	**19.2 ±1.2**
Six graph statistics (standardised)			
Distribution gap: SWD_2 (↓)	1.22 ±0.02	0.79 ±0.02	**0.47 ±0.05**
Nearest synthetic graph, homophilous (↓)	1.98 ±0.26	1.60 ±0.11	**1.17 ±0.06**
Nearest synthetic graph, heterophilous (↓)	1.26 ±0.06	**0.90 ±0.06**	0.95 ±0.06
Coverage: benchmarks covered (↑)	7.8 ±1.9	14.4 ±1.0	**18.8 ±3.3**

I WHAT EACH PRIOR TEACHES THE NETWORK

This appendix gives the settings of the two probes of §4.5 and what they show beyond Figure 5. Both probes examine the network of our architecture pretrained on NodePFN's prior and on ours (the two runs of Figure 10).

Probe graphs. The six probe graphs are Cora, Citeseer and Coauthor-CS from the homophilous block and Chameleon, Roman-Empire and Amazon-ratings from the heterophilous one. Each graph is evaluated at the checkpoints after 2, 5, 10, 20, 40 and 60 epochs, in one forward pass on its first split, with 50 feature components, no smoothing and one ensemble member.

Share of the ADR branch. For each of blocks 2 to 11 the probe computes the norm of each residual branch's output relative to that of its input, averaged over nodes, and the share of the graph branch among the three, and then averages over blocks and graphs, with a band of one within-graph standard error. On twelve graphs, which add Pubmed, Amazon-photo, Squirrel, Actor, Texas and Minesweeper to the six, the share on the GADRPFN prior is higher on every graph and also 1.4 times that on NodePFN's prior at the end.

Non-linearity. On a fixed sample of 2,048 nodes, the affinity score of Bouniot et al. (2025) compares the input and output of each block's activation, matching the official code to four decimals. Figure 5b reports its median over the twelve blocks, averaged over the six probe graphs.