GRAPHPFN: A PRIOR-DATA FITTED GRAPH FOUNDATION MODEL

Dmitry Eremeev
HSE University, Yandex Research
eremeev-d@yandex-team.ru

Oleg Platonov
HSE University, Yandex Research
olegplatonov@yandex-team.ru

Gleb Bazhenov
HSE University, Yandex Research
gv-bazhenov@yandex-team.ru

Artem Babenko
Yandex Research, HSE University
arbabenko@yandex-team.ru

Liudmila Prokhorenkova
Yandex Research
ostroumova-la@yandex-team.ru

ABSTRACT

Graph foundation models face several fundamental challenges including transferability across datasets and data scarcity, which calls into question the very feasibility of graph foundation models. However, despite similar challenges, the tabular domain has recently witnessed the emergence of the first successful foundation models such as TabPFNv2 and LimiX. Many of these models are based on the prior-data fitted networks (PFN) framework, in which models are pretrained on carefully designed synthetic datasets to make predictions in an in-context learning setting. Recently, G2T-FM has made the first step towards adopting PFNs for graphs, yet it is limited to hand-crafted features and was never pretrained on graph data. In this work, we make the next step by proposing GraphPFN, a PFN-based model designed and pretrained specifically for graph node-level tasks. Following the PFN framework, we first design a prior distribution of synthetic attributed graphs by using a novel combination of multi-level stochastic block models and a preferential attachment process for structure generation and graph-aware structured causal models for attribute generation. Then, we augment the tabular foundation model LimiX with attention-based graph neighborhood aggregation layers and train it on synthetic graphs sampled from our prior. On diverse real-world graph datasets with node-level tasks, GraphPFN shows strong in-context learning performance and achieves state-of-the-art results after finetuning, outperforming both G2T-FM and task-specific GNNs trained from scratch on most datasets. More broadly, GraphPFN shows the potential of PFN-based models for building graph foundation models. Our code is available at https://github.com/yandex-research/graphpfn.

1 INTRODUCTION

Foundation models have significantly advanced the state of the art in natural language processing and computer vision by learning transferable representations from large unannotated datasets. Notable examples, such as BERT (Devlin et al., 2019) and GPT-3 (Brown et al., 2020) in NLP, or ViT (Dosovitskiy et al., 2020) and CLIP (Radford et al., 2021) in vision, have fundamentally changed how models are built by reducing the dependence on task-specific models and large labeled datasets. Inspired by these successes, there is growing interest in extending the foundation models methodology to other modalities, including graphs.

However, developing graph foundation models (GFMs) is much more challenging. Unlike text and images, graph data does not constitute *a single domain*. Instead, graphs are used to represent data from *different domains*, e.g., social networks (both virtual and real-world), information networks,

transportation networks, co-purchasing networks, various physical, biological, or engineering systems, or even networks of abstract concepts. As a consequence, both the structure of a graph and its attributes (features and labels) may vary significantly across graph datasets and tasks. Moreover, the amount and diversity of the available graph data are significantly lower compared to those in computer vision and natural language processing. Taken together, these challenges call into question the very feasibility of graph foundation models.

However, despite facing similar challenges, tabular machine learning has recently witnessed the emergence of the first successful tabular foundation models (TFMs) such as TabPFNv2 (Hollmann et al., 2025) or LimiX (Zhang et al., 2025a). Many of these models are based on the framework of prior-data fitted networks (PFNs, Müller et al., 2022; Hollmann et al., 2023). PFNs are pretrained on diverse synthetic datasets drawn from a *prior* to approximate Bayesian inference and can make predictions via in-context learning. Strong performance in the tabular domain suggests that PFNs offer a promising path towards building foundation models for both tabular and graph domains.

The recent works G2T-FM (Eremeev et al., 2025), TAG (Hayler et al., 2025), and TabPFN-GN (Choi et al., 2025) have made the first step towards adopting PFNs for graph tasks by utilizing foundation models for tabular data to create graph foundation models for node-level tasks. For this, they augment node features with graph-based information such as neighborhood-aggregated features or Laplacian positional encodings. This allows for transforming a graph node-level prediction problem into a tabular prediction problem and applying an existing tabular foundation model to this task. The resulting approach shows strong performance, but such models still depend heavily on hand-crafted features and lack large-scale pretraining on diverse graph data. As a result, they are limited in their ability to capture complex graph patterns.

In this work, we make the next step by proposing GraphPFN, a PFN-based model designed and pretrained specifically for graph node-level tasks. Following the PFN framework, we pretrain GraphPFN on synthetic datasets drawn from a carefully designed graph prior. For generating graph structures, we propose an approach that combines multiple stochastic block models and augments them with a preferential-attachment process. We then generate graph-structure-dependent node attributes for our graphs by augmenting tabular structured causal models (SCMs, Hollmann et al., 2023; Qu et al., 2025) typical for tabular PFNs with message-passing mechanisms at random SCM nodes. This method allows us to efficiently generate millions of realistic and diverse synthetic graph datasets. Then, we initialize GraphPFN from the tabular foundation model LimiX (Zhang et al., 2025a) and add an attention-based message-passing layer to each block of LimiX. This allows the model to learn complex graph-specific patterns while retaining its ability to handle diverse features and labels inherited from LimiX.

Our experiments show that on diverse real-world graph node-level prediction datasets, GraphPFN achieves strong in-context learning performance, competitive with the best current models — well-tuned traditional GNNs (Kipf & Welling, 2017; Hamilton et al., 2017; Veličković et al., 2018; Shi et al., 2021) with improved architectures (Platonov et al., 2023b) and recent TFM-based GFMs G2T-FM (Eremeev et al., 2025) and TAG (Hayler et al., 2025). Furthermore, after finetuning, GraphPFN outperforms all other approaches, setting a new state of the art for the considered datasets.

Overall, our main contributions can be summarized as follows:

- We propose GraphPFN, which is, to the best of our knowledge, the first publicly available PFN-based model designed and pretrained specifically for graph node-level tasks.
- We introduce a novel graph prior for the efficient generation of realistic synthetic attributed graphs.
- We demonstrate that the finetuned GraphPFN outperforms both strong traditional GNN baselines and existing graph foundation models.

2 Related work

2.1 Prior-data fitted networks

Prior-data fitted networks (PFNs) were first introduced by Müller et al. (2022). The main idea behind PFNs is to train models that can make predictions on previously unseen datasets in a single forward pass. These models leverage in-context learning (ICL): rather than updating model parameters for

each new dataset, they use the context provided at inference time to adapt their predictions without additional training.

In the PFN framework, the input to the model consists of two parts: a set of training samples with their labels, called the *context*, and a set of test samples without labels, called the *query*. During a single forward pass, the model uses the context to make predictions for the query samples, thus performing ICL. In practice, PFNs are commonly implemented as Transformers with a specific attention mask (Hollmann et al., 2023; 2025): context (training) samples attend to all other context samples, while query (test) samples are only allowed to attend to context samples and not to each other. This structure ensures that predictions for each query are based solely on the training data.

PFNs are trained via pretraining on a large collection of synthetic datasets. To achieve this, one specifies a *prior* over supervised datasets, and the model is trained to perform ICL as described above, predicting labels for query samples given context samples. As shown by Müller et al. (2022), this procedure trains the network to approximate the posterior predictive distribution under the chosen prior, which provides the main theoretical motivation for the approach.

2.2 Tabular foundation models

In their pioneering work TabPFN (Hollmann et al., 2023) and its successor TabPFNv2 (Hollmann et al., 2025), the authors proposed to utilize the framework of prior-data fitted networks to create tabular foundation models and showed that such models can achieve strong results, competitive with other approaches (Erickson et al., 2025). Nowadays, TFMs have become an active research area, with several new methods released recently (Zhang et al., 2025a;b; Grinsztajn et al., 2025; Qu et al., 2026). Some methods focus on scalability (Qu et al., 2025) or faster inference (Mueller et al., 2025), while others emphasize training on real-world datasets rather than synthetic tasks (Ma et al., 2025). Together, these works broaden the design space of tabular foundation models by trading off data sources, computational efficiency, and scalability.

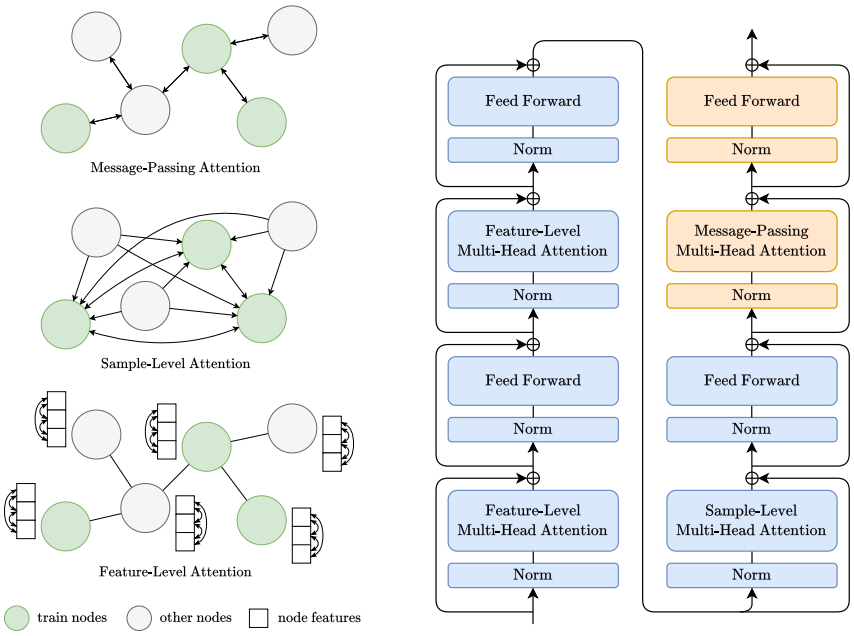
2.3 Graph foundation models

Similar to foundation models for tabular data, graph foundation models face the challenge of handling datasets from diverse domains. A particularly difficult, yet crucial, aspect is managing the wide variety of node attributes (features and labels) present in different graphs. Early GFMs did not fully address this issue. They often relied on dimensionality reduction techniques such as principal component analysis or singular value decomposition (Xia & Huang, 2024; Zhao et al., 2024; Wang et al., 2025; Yu et al., 2025), or restricted their focus to graphs where node attributes are all textual (Wang et al., 2024; He et al., 2025; Liu et al., 2024).

More recent works, such as G2T-FM (Eremeev et al., 2025), TAG (Hayler et al., 2025), and TabPFN-GN (Choi et al., 2025), have explored leveraging tabular foundation models to better address feature diversity in graph datasets. These approaches incorporate hand-crafted features, for example, neighborhood-aggregated features or Laplacian positional encodings, to effectively convert graph information into tabular features. Empirical results show that these methods achieve strong results, always significantly outperforming prior GFMs (Xia et al., 2024; Xia & Huang, 2024; Zhao et al., 2024; Finkelshtein et al., 2025; Zhao et al., 2025) and frequently outperforming well-tuned GNNs trained from scratch (Kipf & Welling, 2017; Hamilton et al., 2017; Veličković et al., 2018; Shi et al., 2021) with improved architectures (Platonov et al., 2023b), supporting the utility of employing tabular foundation models as a basis for learning on graph data.

3 GraphPFN

GraphPFN is a foundation model designed for in-context learning on graph-structured data. Inspired by recent advances in prior-data fitted networks (PFNs) for tabular data (Hollmann et al., 2025), GraphPFN extends these ideas to graphs by augmenting a tabular foundation model with attention-based message-passing adapters. This design allows GraphPFN to reuse strong feature modeling from tabular pretraining while capturing complex graph-specific patterns.



(a) Different attention mechanisms in GraphPFN. (b) Base neural block in GraphPFN architecture.

Figure 1: An illustration of GraphPFN architecture.

3.1 Architecture

Our model architecture extends the tabular foundation model LimiX (Zhang et al., 2025a) by adding attention-based message-passing layers to each of its transformer blocks. This strategy is inspired by recent findings that tabular foundation models can already learn patterns relevant for various graph tasks (Eremeev et al., 2025; Hayler et al., 2025; Choi et al., 2025). By initializing our model with a pretrained tabular foundation model instead of training from scratch, we leverage these learned representations, which significantly reduces computational costs while still achieving strong results.

Below, we first summarize the common architecture of a tabular foundation model (Hollmann et al., 2025; Zhang et al., 2025a), considering LimiX as the specific example, to clarify how GraphPFN represents samples (nodes) and features, and how attention flows in the base model. We then describe how the graph adapters modify this flow to leverage the graph topology.

LimiX LimiX (Zhang et al., 2025a) is a transformer-style foundation model for tabular data that departs from the common design of representing each sample with a single fixed-length embedding. Instead, it uses a multi-token representation: for every sample, each feature contributes one token,[1] yielding a token grid with one axis for features and one for samples. This design naturally handles a variable number of heterogeneous features in different datasets without changing and retraining the model.

A LimiX transformer block contains three attention layers, each followed by an element-wise feed-forward network (FFN). Two layers are feature-level multi-head attention (MHA) modules that operate within a sample, allowing all feature tokens of the same sample to attend to each other. The third is a sample-level MHA that operates within a feature across samples, allowing tokens corresponding to the same feature to exchange information across the dataset. The feature-level MHAs enable rich interactions among features within each sample, while the sample-level MHA supports in-context learning by transferring information across samples for the same feature.

Attention masking at the sample level follows the standard PFN protocol: training (context) samples attend to all other training samples, and test (query) samples attend only to training samples. Thus,

[1] In the current implementation, two features are grouped into one token, but we omit this detail for clarity.

information can flow from train to test but not from test to train or between test samples. Feature-level attention within a sample is unmasked.

Graph adapters To inject graph structure information without disrupting LimiX's tokenization, we add a message-passing adapter that implements scaled dot product attention between neighboring nodes at the end of every LimiX transformer block (see Figure 1 for an illustration). Note that we use the scaled dot product attention that is identical to the original Transformer attention (Vaswani et al., 2017) except for being restricted to 1-hop graph neighborhoods. Intuitively, our message-passing adapter performs a second, graph-structure-aware round of sample-level attention: tokens may exchange information only along the observed edges. In contrast to the global sample-level attention of LimiX (which is masked by the PFN protocol), the graph adapter is masked by the adjacency and therefore routes information locally, from each node to its neighbors. Because we keep the per-feature token representation intact, the adapter runs over the sample axis for each feature token independently, using the same graph mask across features.

This design complements the global PFN-style attention by adding a local channel that is common in graph learning. We process the entire graph jointly and the graph adapter allows bidirectional exchange between labeled and unlabeled nodes along edges. Similar to the classic GNNs, these message-passing layers allow the model to capture complex graph dependencies that cannot be captured by hand-crafted features.

Each adapter is implemented as a sparse, multi-head attention module with the adjacency as its mask, where two nodes attend to each other if and only if an edge connects them. Similar to other attention modules, it is followed by a feed-forward network independently applied to each token. Both the FFN and the message-passing layer are wrapped with residual connections (He et al., 2016) and layer normalization (Ba et al., 2016), mirroring the structure of the LimiX blocks for stable optimization.

3.2 Pretraining

GraphPFN was pretrained under the PFN framework (Müller et al., 2022; Hollmann et al., 2023; 2025) by continuing training from the LimiX checkpoint. In total, we used $2{,}240{,}000$ synthetic datasets generated according to the prior described below. Pretraining ran for approximately 7 days on 8 NVIDIA A100 80GB GPUs. Each GPU processed one synthetic dataset per step with 20 gradient accumulation steps, resulting in 160 datasets per optimizer step.

Inspired by TabICL (Qu et al., 2025), we pretrained GraphPFN in two stages to optimize training efficiency by gradually increasing the size of synthetic datasets. Specifically, Stage 1 ran for $10{,}000$ optimizer steps with sample sizes drawn log-uniformly from $1{,}000$ to $2{,}000$ and took approximately 2 days. Stage 2 ran for $4{,}000$ optimizer steps with sample sizes drawn log-uniformly from $8{,}000$ to $10{,}000$ and took approximately 5 days.

We optimized the model using AdamW (Loshchilov & Hutter, 2019) with weight decay 0.1. We used a cosine annealing learning rate schedule (Loshchilov & Hutter, 2017) with linear warmup over the first 10% of training steps and a base learning rate of $3 \cdot 10^{-4}$. To improve training stability and preserve the feature modeling capabilities of LimiX, we froze all model layers except the graph adapters. Only these components were updated during pretraining. To further stabilize training, we applied an exponential moving average (EMA) to the weights throughout pretraining, with decay 0.98 in Stage 1 and 0.95 in Stage 2.

Objective We optimize a joint objective that combines the PFN supervised loss with the masked graph modeling (MGM) loss (Li et al., 2023). For the supervised PFN term, we sample a random set of context nodes for each dataset, compute predictions for all other nodes, and minimize the supervised loss on them. We use the cross-entropy loss for classification tasks and the mean squared error loss for regression tasks.

To encourage more graph-aware representations, inspired by the success of GNN self-supervised pretraining, we add the masked graph modeling term from Li et al. (2023). Specifically, we randomly sample a fraction $p = 0.1$ of edges as positive samples, remove these edges from the input graph, and uniformly sample an equal number of unconnected node pairs as negative examples. We then train the model with the cross-entropy loss to distinguish between positive and negative edges. For this, we apply an additional MLP head to the pointwise multiplication of the target embeddings from the

last layer for the source and destination nodes of each sampled edge. The total loss is the sum of the supervised loss and the MGM loss, with a coefficient of 0.1 applied to the MGM term.

Learning curves To monitor pretraining progress, we periodically evaluated GraphPFN in the ICL setting on several GraphLand datasets. We observe that the first pretraining stage already yields strong improvements over the initialization, while the second stage further improves performance consistently across datasets. The corresponding learning curves are shown in Appendix C.

4 GRAPH PRIOR

As discussed above, GraphPFN is based on the prior-data fitted networks (PFNs) framework (Müller et al., 2022). In this approach, the model is pretrained on a large number of synthetic graph datasets sampled from a chosen prior distribution. Since the pretrained model aims to approximate the posterior predictive distribution, it is crucial to design a diverse, high-quality, and realistic prior. In this section, we describe the prior used for pretraining GraphPFN. First, we explain our method for generating realistic graph structures. Then, we describe how we use these graphs to generate node attributes and targets.

4.1 STRUCTURE GENERATION

Our main aim is to generate graph structures similar to real-world graphs. After examining graphs from a range of graph machine learning datasets, we find that most of them exhibit strong clustering (community) structure, which in general is a common feature of real-world graphs (Girvan & Newman, 2002). Thus, as the basis for our graph generation process, we use the degree-corrected stochastic block model (SBM) (Karrer & Newman, 2011), which can generate graphs with community structure. However, we find that graphs generated from the degree-corrected SBM exhibit clusters that appear overly regular and well-defined, resembling separated spheres. In contrast, clusters in real-world graphs are often more irregular, with more complex shapes and frequent overlaps. Thus, to obtain graph structures similar to real-world ones, we design a novel method that combines multiple SBMs. First, we generate several *first-level* graphs from SBMs with different parameters. Then, we generate a *second-level* graph from another SBM, such that this second-level graph has the number of nodes equal to the total number of nodes across all first-level graphs. We then randomly assign each node from the first-level graphs to a unique node in the second-level graph, thus essentially constructing a bijection f between first-level and second-level graph nodes. Then, we transfer each edge from the first-level graphs to the second-level graph by creating a new edge in the second-level graph between the corresponding nodes, i.e., if there was an edge between nodes u and v in the first-level graphs, then we create an edge between nodes $f(u)$ and $f(v)$ in the second-level graph. The obtained second-level graph with additional edges combines multiple graphs generated from different SBMs and exhibits clusters of nodes with complex shapes and overlaps that we aimed to capture.

Further, we observe that most graphs from graph benchmarks exhibit a core-periphery structure, where the core is composed of multiple relatively dense clusters, but there are also many low-degree peripheral nodes. Such a structure is known to be common in real-world networks (Zhang et al., 2015). While our method of combining multiple graphs generated from SBMs produces realistic node clusters, it produces a relatively small number of peripheral nodes. Thus, we augment the approach discussed above with a preferential attachment (PA) process (Price, 1965; 1976; Albert & Barabási, 2002). Specifically, we use the graph obtained thus far as the initialization for the PA process. We sequentially add low-degree nodes and connect them to the previous ones with probabilities proportional to their degrees. The initial degree of each new node is chosen randomly.

Our method has many hyperparameters such as the number and size of blocks for SBMs or their degree sequences. Similar to prior works on PFNs (Hollmann et al., 2023; 2025; Qu et al., 2025; 2026), we define probability distributions for each hyperparameter and sample new hyperparameters for each synthetic graph, which allows us to generate diverse graphs. At the same time, we can easily set bounds for sizes, densities, or maximum degrees of the generated graphs, allowing us to ensure that the graphs fit the desired constraints.

We provide example visualizations of several graphs generated by our process in Appendix B.

Table 1: Characteristics of real-world graphs from the GraphLand benchmark.

	artnet-exp	artnet-views	avazu-ctr	city-reviews	city-roads-M	hm-prices	tolokers-2	twitch-views
Avg. degree	11.12	11.12	288.04	15.66	3.75	460.92	88.28	80.87
Clustering coefficient	0.03	0.03	0.24	0.26	0.00	0.27	0.23	0.02
Avg. pairwise distance	4.34	4.43	3.72	4.78	129.89	2.39	2.84	2.84
Homophily	0.28	–	–	0.69	–	–	0.10	–
Assortativity	–	0.19	0.18	–	0.74	0.12	–	-0.41
Degree power-law R^2	0.70	0.70	0.90	0.92	0.52	0.84	0.83	0.97

Table 2: Summary of distribution of characteristics of synthetic graphs for the first pretraining stage.

	Mean	Std	Min	Q25	Median	Q75	Max
Avg. degree	125.64	130.27	2.39	10.24	87.27	206.33	479.03
Clustering coefficient	0.3	0.23	0.0	0.08	0.28	0.49	0.88
Avg. pairwise distance	2.92	1.35	1.57	2.01	2.28	3.39	9.76
Homophily	0.06	0.16	−0.48	−0.01	0.0	0.08	0.96
Assortativity	0.05	0.13	−0.21	−0.0	0.0	0.07	0.7
Degree power-law R^2	0.68	0.21	0.11	0.51	0.7	0.86	0.99

4.2 Attribute generation

We generate features and targets for synthetic graphs with a neural structural causal model (SCM) that extends the MLP-based SCM of Qu et al. (2025); Hollmann et al. (2023). As a starting point, we follow the TabICL protocol: we sample an MLP architecture (number of layers, dimension of hidden layers, activation function) and its weights at random, draw random inputs, propagate them through the network, and then designate a random subset of neurons as observed features and another random neuron as the target, leaving the rest as latent variables. This yields a broad family of causal mechanisms in which features and targets can depend on each other and on latent confounders. We refer to Qu et al. (2025); Hollmann et al. (2023) for further details.

To make attributes also depend on the graph structure, we extend this SCM in two complementary ways. First, we introduce a mixture of MLP and GNN neurons. For each dataset, we sample a mixing probability $p \in \{0.0, 0.1, \ldots, 0.9, 1.0\}$ and a GNN type from $\{\mathrm{GCN}, \mathrm{SAGE}_{\mathrm{mean}}, \mathrm{SAGE}_{\mathrm{max}}, \mathrm{SAGE}_{\mathrm{min}}, \mathrm{GT}\}$. At every hidden layer, we compute the outputs of both an MLP transformation and a GNN layer. Each neuron is then independently assigned to be MLP-type or GNN-type, with probabilities $1 - p$ and p, respectively, and its value is taken from the corresponding transformation. This mechanism controls how strongly the generated variables depend on the graph. Second, with a 0.5 probability, we augment the random inputs with Laplacian positional encodings (LapPE) (Dwivedi et al., 2020; Belkin & Niyogi, 2001) of dimension uniformly sampled from $\{1, \ldots, 32\}$ to further integrate graph structure into the data generation process.

Together, the mixed MLP/GNN neurons and optional LapPE inject graph information into the SCM while remaining close to the tabular prior. When $p = 0$ and LapPE is not used, the procedure reduces to the TabICL-style tabular SCM, while a larger value of p and the inclusion of LapPE increase the influence of graph structure on both features and targets.

4.3 Numerical analysis of graph characteristics

To further assess how closely our synthetic datasets match the properties of real-world graphs, we perform a numerical analysis of several graph characteristics. Specifically, we sample $1{,}000$ synthetic datasets from the prior used in the first pretraining stage and compute the following statistics.

First, we report basic structural measures, including the average degree, the clustering coefficient, and an estimate of the average pairwise distance. Second, to quantify the tendency of adjacent nodes to have similar targets, we compute unbiased homophily (Mironov & Prokhorenkova, 2024) for classification datasets and assortativity (Newman, 2003) for regression datasets. Finally, to evaluate how well the graphs follow the power-law degree distribution, we compute the cumulative degree distribution: for each degree value k, we estimate $P_k = \mathbb{P}(\deg(u) \geq k)$. We then fit a linear regression to the pairs $(\log k, \log P_k)$ and report the corresponding R^2.

Table 2 summarizes the distributions of these characteristics for synthetic datasets, while Table 1 reports the same statistics for real datasets. In most cases, the distribution of characteristics in the synthetic datasets covers that of the real-world datasets.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Datasets In terms of dataset selection, we follow the experimental setup of G2T-FM (Eremeev et al., 2025). We evaluate two collections of datasets: (i) real-world datasets from the recently proposed GraphLand benchmark (Bazhenov et al., 2025); and (ii) some of the classic graph datasets. Together, these datasets cover node classification and regression, come from diverse application domains, include both homophilous and non-homophilous graphs,[2] and span a range of densities and other graph structural properties. Table 7 lists the datasets used in our study and summarizes their statistics. For all datasets, we use 10%/10%/80% train/validation/test splits. Due to current limitations of TFMs, we restrict our study to small- and medium-scale datasets and exclude classification tasks with more than 10 classes.[3]

In our evaluation, we run all experiments 10 times and report the mean and standard deviation of the model performance. We report average precision for binary classification tasks, accuracy for multiclass classification tasks, and R^2 for regression tasks. For all metrics, higher is better.

Methods In addition to the proposed GraphPFN, we evaluate the following methods:

- **LightGBM** (Ke et al., 2017), a strong tabular baseline, augmented with neighborhood feature aggregation (NFA) (Bazhenov et al., 2025) to incorporate information about the graph structure.
- **Classic GNNs**: GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2018), neighborhood-attention Graph Transformer (GT) (Shi et al., 2021). Following Platonov et al. (2023b), we augment these models with residual connections (He et al., 2016), layer normalization (Ba et al., 2016), and MLP blocks, which have been shown to substantially improve the performance of classic GNNs (Luo et al., 2024; 2025). We perform extensive hyperparameter tuning for these models.
- **Prior GFMs:** OpenGraph (Xia et al., 2024), AnyGraph (Xia & Huang, 2024), GCOPE (Zhao et al., 2024), and TS-GNN (Finkelshtein et al., 2025), which are not based on the PFN framework.
- **Recent PFN-based GFMs:** G2T-FM (Eremeev et al., 2025) and TAG (Hayler et al., 2025). To the best of our knowledge, these are the strongest publicly available graph foundation models for node-level tasks with arbitrary features.

We evaluate GraphPFN in both in-context learning (ICL) and full finetuning (FT) settings. Other models are evaluated in ICL, FT, or both settings based on their official implementations. For G2T-FM and GraphPFN in the ICL setting, we additionally report ensembling (E) results obtained by averaging the predictions over 10 forward passes with different random seeds. Since TAG inherently employs ensembling, we also mark it with "(ICL, E)", despite the exact ensembling implementation being different from G2T-FM and GraphPFN. For the FT setting, we finetune the entire model, following Eremeev et al. (2025); Rubachev et al. (2025). Inspired by Abboud et al. (2020); Sato et al. (2021); Kanatsoulis et al. (2025), we additionally augment the input of GraphPFN with 8 random features during each forward pass in all experiments in this section. For further details on the experimental setup, please refer to Appendix D.

5.2 EXPERIMENTAL RESULTS

Tables 3 and 4 present the results of our experiments. Below, we summarize and discuss our key observations. Additional experimental results are provided in Appendix C.

[2] A graph is called homophilous if its edges tend to connect nodes with similar labels, see Newman (2003); Platonov et al. (2023a); Mironov & Prokhorenkova (2024) for details.

[3] In principle, TFMs can handle more than 10 classes via schemes such as error-correcting output codes, as proposed in Hollmann et al. (2025). For simplicity, we focus on datasets with at most 10 classes that are natively supported by existing TFMs.

Table 3: Evaluation results on the GraphLand datasets under the RL (Random Low) data split. We report average precision for binary classification tasks and R^2 for regression tasks. We also report average rank over classification datasets in the "AR (cls)" column and over all datasets in the "AR (all)" column. During the computation of "AR (all)" we exclude all methods that do not support regression tasks, so that ranks are computed only over tables with no missing entries.

	artnet-exp	city-reviews	tolokers-2	artnet-views	avazu-ctr	city-roads-M	hm-prices	twitch-views	AR (cls)	AR (all)
LightGBM + NFA	46.13 ± 0.04	78.53 ± 0.01	56.34 ± 0.06	56.10 ± 0.02	31.71 ± 0.01	61.18 ± 0.03	70.84 ± 0.04	60.14 ± 0.01	**11.0**	**9.75**
GCN	44.86 ± 0.36	77.81 ± 0.15	56.27 ± 0.31	56.03 ± 0.25	32.00 ± 0.16	58.82 ± 0.25	68.02 ± 0.42	75.51 ± 0.05	**13.33**	**10.25**
GraphSAGE	45.14 ± 0.36	78.17 ± 0.10	54.43 ± 0.34	49.32 ± 0.91	31.44 ± 0.16	59.44 ± 0.27	70.00 ± 0.74	66.29 ± 0.32	**12.33**	**11.25**
GAT	45.06 ± 0.52	77.74 ± 0.21	57.41 ± 0.85	53.60 ± 0.24	32.63 ± 0.17	59.86 ± 0.20	72.07 ± 1.22	72.89 ± 0.27	**12.67**	**9.5**
GT	46.41 ± 0.71	77.34 ± 0.21	56.98 ± 0.55	53.37 ± 0.46	31.11 ± 0.49	59.55 ± 0.28	69.44 ± 0.94	72.13 ± 0.13	**12.67**	**10.88**
OpenGraph (ICL)	15.16 ± 0.83	59.09 ± 0.72	40.38 ± 1.13	–	–	–	–	–	**17.0**	–
AnyGraph (ICL)	12.84 ± 0.93	63.71 ± 1.45	28.75 ± 3.56	–	–	–	–	–	**18.33**	–
TS-GNN (ICL)	20.44 ± 1.05	43.46 ± 5.17	38.54 ± 0.94	–	–	–	–	–	**17.33**	–
G2T-LimiX (ICL)	48.44 ± 0.23	77.81 ± 0.49	61.45 ± 0.33	60.94 ± 0.11	32.39 ± 0.13	64.53 ± 0.09	74.96 ± 0.07	71.08 ± 0.07	**7.33**	**7.5**
GraphPFN (ICL)	50.85 ± 0.31	79.89 ± 0.10	60.33 ± 0.60	61.77 ± 0.23	30.41 ± 0.18	63.55 ± 0.29	75.38 ± 0.26	73.27 ± 0.13	**5.67**	**7.12**
G2T-LimiX (ICL, E)	48.52 ± 0.29	77.97 ± 0.53	61.58 ± 0.32	60.96 ± 0.10	32.41 ± 0.14	64.69 ± 0.05	74.97 ± 0.06	71.30 ± 0.07	**6.33**	**6.5**
TAG-TabPFNv2 (ICL, E)	47.87 ± 0.39	77.38 ± 0.29	59.33 ± 1.00	–	–	–	–	–	**11.33**	–
TAG-LimiX (ICL, E)	50.19 ± 0.46	78.87 ± 0.13	59.46 ± 1.05	–	–	–	–	–	**7.0**	–
GraphPFN (ICL, E)	51.49 ± 0.10	80.22 ± 0.04	60.98 ± 0.25	62.07 ± 0.04	30.46 ± 0.14	64.29 ± 0.12	75.82 ± 0.06	73.81 ± 0.04	**3.0**	**5.25**
GCOPE (FT)	14.92 ± 1.56	67.16 ± 0.98	28.81 ± 1.28	–	–	–	–	–	**17.33**	–
G2T-LimiX (FT)	49.89 ± 0.18	80.14 ± 0.06	61.28 ± 0.70	62.08 ± 0.12	34.02 ± 0.27	65.82 ± 0.18	76.27 ± 0.24	73.69 ± 0.35	**5.0**	**4.25**
GraphPFN (FT)	50.64 ± 1.27	80.49 ± 0.16	60.90 ± 0.98	63.90 ± 0.12	34.34 ± 0.50	66.04 ± 0.41	79.26 ± 0.43	78.17 ± 0.13	**4.0**	**2.75**
G2T-LimiX (FT, E)	49.96 ± 0.09	80.15 ± 0.07	60.71 ± 0.57	62.11 ± 0.06	33.86 ± 0.33	65.96 ± 0.11	76.30 ± 0.19	73.62 ± 0.49	**5.67**	**4.38**
GraphPFN (FT, E)	51.97 ± 0.40	80.89 ± 0.05	60.92 ± 0.56	65.12 ± 0.11	35.05 ± 0.23	67.27 ± 0.14	80.13 ± 0.37	78.50 ± 0.36	**2.33**	**1.5**

Observation 1. *GraphPFN shows strong ICL performance, outperforming GNNs and other GFMs on the GraphLand benchmark, while matching them on classic graph datasets.*

Specifically, when compared with GNNs and other GFMs evaluated in an ICL regime, GraphPFN (ICL) achieves the best average rank on GraphLand datasets and only loses to GAT on classic datasets. Importantly, unlike G2T-FM, which also achieves strong ICL performance, GraphPFN does not require potentially heavy preprocessing like the computation of Laplacian positional encodings or PageRank.

Observation 2. *Ensembling further boosts the ICL performance of GraphPFN, in particular allowing it to outperform GNNs on average on both collections of datasets.*

Specifically, GraphPFN (ICL, E) achieves better average rank than GNNs and G2T-LimiX (ICL, E) on both collections of datasets. When compared to TAG, GraphPFN outperforms it on all considered GraphLand datasets, and performs on par on classic datasets, while requiring no potentially heavy preprocessing.

Observation 3. *Finetuning allows GraphPFN to achieve the best results on 10 out of 13 datasets, and the gains over the second-best method are often substantial.*

Specifically, the finetuned and ensembled GraphPFN achieves the best performance among all considered methods on all datasets except tolokers-2, amazon-ratings, and pubmed. Moreover, GraphPFN often yields substantial gains over the second-best method, for example, of at least one percentage point on the artnet-exp, artnet-views, avazu-ctr, city-roads-M, hm-prices, twitch-views, and questions datasets.

The performance of finetuned GraphPFN without ensembling is also strong. Specifically, it achieves the second-best result across all considered methods on 7 out of 13 datasets, only losing to GraphPFN (FT, E). It also achieves better average ranks than all other methods except those evaluated in the (FT, E) regime on both collections of datasets.

We hypothesize that this strong performance stems from GraphPFN's ability to capture complex graph patterns via message passing, in contrast to G2T-FM and TAG, which rely on hand-crafted graph-based features. At the same time, the pretraining procedure appears to be critical. When we replace the pretrained graph adapters with randomly initialized ones and finetune on a downstream dataset, performance drops substantially; see Appendix C for details.

Table 4: Evaluation results on classic graph datasets under a random 10%/10%/80% train/val/test split. We report average precision for binary classification tasks and accuracy for multiclass classification tasks. We also report average rank (AR) in the last column.

	amazon-ratings	facebook	pubmed	questions	wiki-cs	AR
GCN	41.43 ± 0.49	91.26 ± 0.21	85.46 ± 0.19	15.42 ± 0.67	81.74 ± 0.21	**11.4**
GraphSAGE	40.07 ± 0.53	91.12 ± 0.22	86.04 ± 0.27	16.55 ± 0.64	81.50 ± 0.27	**11.6**
GAT	40.67 ± 0.55	92.61 ± 0.21	84.81 ± 0.24	16.75 ± 0.67	82.25 ± 0.27	**9.2**
GT	41.56 ± 0.40	91.71 ± 0.22	84.95 ± 0.19	14.03 ± 0.90	82.54 ± 0.21	**10.4**
OpenGraph (ICL)	29.36 ± 1.24	75.27 ± 5.05	70.30 ± 2.67	3.77 ± 0.65	75.66 ± 0.39	**16.8**
AnyGraph (ICL)	33.49 ± 3.44	61.17 ± 8.64	65.31 ± 6.26	4.27 ± 0.66	65.17 ± 2.51	**17.0**
TS-GNN (ICL)	43.00 ± 0.13	77.87 ± 2.73	64.41 ± 5.11	5.00 ± 0.48	46.25 ± 9.77	**15.6**
G2T-LimiX (ICL)	44.20 ± 0.26	91.30 ± 0.23	88.93 ± 0.46	15.32 ± 0.81	80.12 ± 0.23	**10.2**
GraphPFN (ICL)	42.09 ± 2.06	90.44 ± 0.37	89.79 ± 0.35	18.27 ± 2.24	79.58 ± 0.40	**10.0**
G2T-LimiX (ICL, E)	44.65 ± 0.18	91.66 ± 0.05	89.19 ± 0.18	15.46 ± 0.93	81.09 ± 0.09	**8.8**
TAG-TabPFNv2 (ICL, E)	43.97 ± 0.52	93.11 ± 0.17	87.80 ± 0.20	15.07 ± 1.32	82.59 ± 0.12	**7.6**
TAG-LimiX (ICL, E)	44.89 ± 0.15	92.85 ± 0.14	88.09 ± 0.18	15.53 ± 1.99	82.64 ± 0.17	**5.8**
GraphPFN (ICL, E)	44.52 ± 0.27	91.70 ± 0.21	90.36 ± 0.13	20.79 ± 0.49	81.69 ± 0.15	**6.4**
GCOPE (FT)	39.90 ± 0.43	85.08 ± 0.17	79.35 ± 0.70	6.59 ± 0.43	59.13 ± 1.20	**15.6**
G2T-LimiX (FT)	44.00 ± 0.92	92.17 ± 0.22	90.57 ± 0.13	20.16 ± 0.56	81.90 ± 0.43	**6.0**
GraphPFN (FT)	45.34 ± 0.35	92.91 ± 0.22	89.79 ± 0.35	21.85 ± 1.04	81.91 ± 0.44	**4.0**
G2T-LimiX (FT, E)	46.05 ± 0.23	92.53 ± 0.13	90.88 ± 0.14	20.83 ± 0.44	83.11 ± 0.39	**2.6**
GraphPFN (FT, E)	45.89 ± 0.53	93.94 ± 0.12	90.36 ± 0.13	22.76 ± 0.55	83.53 ± 0.25	**1.6**

Table 5: Results of pretraining GraphPFN on simpler graph models.

	artnet-exp	artnet-views	avazu-ctr	city-reviews	city-roads-M	hm-prices	tolokers-2	twitch-views
Erdős–Rényi	46.92	56.17	25.09	78.60	63.75	66.52	45.26	55.92
Degree-corrected SBM	47.27	60.87	25.68	78.71	59.60	69.86	58.29	62.69
Preferential attachment	48.16	58.56	28.09	76.88	59.14	69.16	55.53	61.78
Ours	50.61	60.05	28.31	79.46	61.65	73.37	59.04	64.23

5.3 Pretraining on simpler random graphs

To assess the utility of the proposed graph generation procedure, we conduct an ablation study in which we replace it during pretraining with simpler random graph models. Specifically, we consider three alternatives: the Erdős–Rényi model (Erdős & Rényi, 1959), the degree-corrected SBM (Karrer & Newman, 2011), and the preferential attachment model (Albert & Barabási, 2002) with varying initial degrees. Due to the high computational cost, we run only the first stage of pretraining for all three alternatives. For a fair comparison, we evaluate them against an intermediate GraphPFN checkpoint taken after completing the first pretraining stage.

The results, presented in Table 5, indicate that our proposed procedure yields superior performance in most cases. The only exceptions are the Erdős–Rényi model on the `city-roads-M` dataset and the degree-corrected SBM on `artnet-views`. For `artnet-views`, the difference is small and can plausibly be attributed to stochasticity in the pretraining process. For `city-roads-M`, we hypothesize that the outcome reflects a mismatch between the structural biases of our procedure, which primarily produces graphs resembling information, social, and web networks, and road networks, which typically lack a pronounced community or core-periphery structure. In this setting, the Erdős–Rényi model, which does not enforce these properties, may be more suitable. Conversely, on most of the datasets, the results for the Erdős–Rényi model are worse than for the more complex approaches, which indicates that a complex and diverse prior is important for GraphPFN pretraining.

6 Conclusion

In this work, we propose GraphPFN, a prior-data fitted graph foundation model for node-level tasks. Following the PFN framework, GraphPFN is pretrained on synthetic datasets drawn from a carefully designed prior over attributed graphs and makes predictions in the in-context learning setting. Our experiments show that GraphPFN achieves strong results: even in the ICL setting, it often outperforms classic GNNs and prior ICL GFMs, while after finetuning it consistently reaches state-of-the-art performance, often by substantial margins. Overall, our results suggest that pretraining graph foundation models on synthetic datasets drawn from a well-designed prior is a promising direction for building more generalizable GFMs. Despite promising results, the current implementation of GraphPFN has several limitations, which we discuss in Appendix A.

REFERENCES

Ralph Abboud, Ismail Ilkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization. *arXiv preprint arXiv:2010.01179*, 2020.

Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Muhammed Fatih Balin and Ümit Çatalyürek. Layer-neighbor sampling—defusing neighborhood explosion in GNNs. *Advances in Neural Information Processing Systems*, 36:25819–25836, 2023.

Marc Barthélemy. Spatial networks. *Physics reports*, 499:1–101, 2011.

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

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in Neural Information Processing Systems*, 14, 2001.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 257–266, 2019.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

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

VP Dwivedi, CK Joshi, T Laurent, Y Bengio, and X Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

Paul Erdős and Alfréd Rényi. On random graphs I. *Publ. math. debrecen*, 6(290-297):18, 1959.

Dmitry Eremeev, Gleb Bazhenov, Oleg Platonov, Artem Babenko, and Liudmila Prokhorenkova. Turning tabular foundation models into graph foundation models. *arXiv preprint arXiv:2508.20906*, 2025.

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. Tabarena: A living benchmark for machine learning on tabular data. *arXiv preprint arXiv:2506.16791*, 2025.

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

Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, Mihir Manium, Rosen Yu, Felix Jablonski, Shi Bin Hoo, Anurag Garg, Jake Robertson, Magnus Bühler, Vladyslav Moroshan, Lennart Purucker, Clara Cornu, Lilly Charlotte Wehrhahn, Alessandro Bonetto, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-2.5: Advancing the state of the art in tabular foundation models. *arXiv preprint arXiv:2511.08667*, 2025.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30, 2017.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Yufei He, Yuan Sui, Xiaoxin He, and Bryan Hooi. UniGraph: Learning a unified cross-domain foundation model for text-attributed graphs. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 448–459, 2025.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations*, 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(8045):319–326, 2025.

Charilaos Kanatsoulis, Evelyn Choi, Stefanie Jegelka, Jure Leskovec, and Alejandro Ribeiro. Learning efficient positional encodings with graph neural networks. In *International Conference on Learning Representations*, 2025.

Brian Karrer and Mark EJ Newman. Stochastic blockmodels and community structure in networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 83(1):016107, 2011.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 30, 2017.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Jintang Li, Ruofan Wu, Wangbin Sun, Liang Chen, Sheng Tian, Liang Zhu, Changhua Meng, Zibin Zheng, and Weiqiang Wang. What's behind the mask: Understanding masked graph modeling for graph autoencoders. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023.

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 *International Conference on Learning Representations*, 2024.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

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

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Can classic GNNs be strong baselines for graph-level tasks? Simple architectures meet excellence. In *International Conference on Machine Learning*. PMLR, 2025.

Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Hamidreza Kamkari, Alex Labach, Jesse C. Cresswell, Keyvan Golestan, Guangwei Yu, Maksims Volkovs, and Anthony L. Caterini. TabDPT: Scaling tabular foundation models on real data. *Advances in Neural Information Processing Systems*, 2025.

Mikhail Mironov and Liudmila Prokhorenkova. Revisiting graph homophily measures. *Learning on Graphs Conference*, 2024.

Andreas C Mueller, Carlo A Curino, and Raghu Ramakrishnan. MotherNet: Fast training and inference via hyper-network transformers. In *International Conference on Learning Representations*, 2025.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022.

Mark EJ Newman. Mixing patterns in networks. *Physical Review E*, 67(2), 2003.

Oleg Platonov, Denis Kuznedelev, Artem Babenko, and Liudmila Prokhorenkova. Characterizing graph datasets for node classification: Homophily-heterophily dichotomy and beyond. *Advances in Neural Information Processing Systems*, 36:523–548, 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*, 2023b.

Derek De Solla Price. A general theory of bibliometric and other cumulative advantage processes. *Journal of the American society for Information science*, 27(5):292–306, 1976.

Derek J De Solla Price. Networks of scientific papers: The pattern of bibliographic references indicates the nature of the scientific research front. *Science*, 149(3683):510–515, 1965.

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

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. *arXiv preprint arXiv:2602.11139*, 2026.

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

Ivan Rubachev, Akim Kotelnikov, Nikolay Kartashev, and Artem Babenko. On finetuning tabular foundation models. *arXiv preprint arXiv:2506.08982*, 2025.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM international conference on data mining (SDM)*, pp. 333–341. SIAM, 2021.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Shuo Wang, Bokui Wang, Zhixiang Shen, Boyan Deng, and Zhao Kang. Multi-domain graph foundation models: Robust knowledge transfer via topology alignment. In *International Conference on Machine Learning*, 2025.

Zehong Wang, Zheyuan Zhang, Nitesh Chawla, Chuxu Zhang, and Yanfang Ye. GFT: Graph foundation model with transferable tree vocabulary. *Advances in Neural Information Processing Systems*, 37:107403–107443, 2024.

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

Lianghao Xia, Ben Kao, and Chao Huang. OpenGraph: Towards open graph foundation models. *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

Xingtong Yu, Zechuan Gong, Chang Zhou, Yuan Fang, and Hui Zhang. SAMGPT: Text-free graph foundation model for multi-domain pre-training and cross-domain adaptation. In *Proceedings of the ACM on Web Conference 2025*, pp. 1142–1153, 2025.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graph-SAINT: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020.

Hanqing Zeng, Muhan Zhang, Yinglong Xia, Ajitesh Srivastava, Andrey Malevich, Rajgopal Kannan, Viktor Prasanna, Long Jin, and Ren Chen. Decoupling the depth and scope of graph neural networks. *Advances in Neural Information Processing Systems*, 34:19665–19679, 2021.

Xiao Zhang, Travis Martin, and Mark EJ Newman. Identification of core-periphery structure in networks. *Physical Review E*, 91(3):032803, 2015.

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, et al. Mitra: Mixed synthetic priors for enhancing tabular foundation models. *Advances in Neural Information Processing Systems*, 2025b.

Haihong Zhao, Aochuan Chen, Xiangguo Sun, Hong Cheng, and Jia Li. All in one and one for all: A simple yet effective method towards cross-domain graph pretraining. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4443–4454, 2024.

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

Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. Layer-dependent importance sampling for training deep and large graph convolutional networks. *Advances in Neural Information Processing Systems*, 32, 2019.

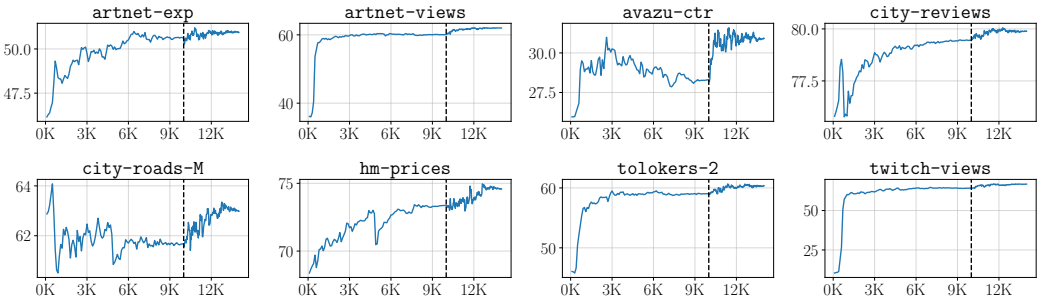


Figure 2: In-context learning performance of intermediate checkpoints of GraphPFN. The x-axis represents the number of steps, and the y-axis represents the metric on the test set. We also add vertical line at 10K steps to separate two pretraining stages.

A Limitations and future work

Despite promising results, the current implementation of GraphPFN has several limitations that may stimulate future research:

- GraphPFN is difficult to scale to very large datasets since its current implementation requires processing the entire dataset at once, which leads to significant memory consumption. Developing more scalable graph foundation models can be a promising direction for future work. For example, one can utilize more memory-efficient TFMs (like TabICL (Qu et al., 2025; 2026)) as backbones or combine GraphPFN with sampling methods (Hamilton et al., 2017; Zeng et al., 2020; Zou et al., 2019; Chiang et al., 2019; Zeng et al., 2021; Balin & Çatalyürek, 2023).
- The proposed graph prior is focused on social and information networks and does not cover graphs from specific domains such as traffic networks. Extending the prior with more diverse random graph models (e.g., geometric graphs (Barthélemy, 2011)) can further improve the results of GraphPFN and make its performance more robust.
- GraphPFN inherits some limitations from its tabular backbone LimiX. For example, GraphPFN does not natively support more than 10 classes in multiclass classification. This limitation can be alleviated with further development of TFMs or by using approaches such as error-correcting output codes, as proposed in Hollmann et al. (2025).
- Currently, GraphPFN is limited to node-level prediction tasks and cannot handle link prediction or graph-level tasks (e.g., graph classification or regression). While we have taken a first step towards link-level tasks by adding a masked graph modeling head to GraphPFN during pretraining, the model still heavily relies on the presence of node-level labels to make predictions, so we cannot directly apply GraphPFN to link prediction tasks.

B Synthetic graph examples

We design our prior to generate graphs that are both realistic and diverse. In Figure 3, we provide examples of our synthetic graphs. Note that these graphs tend to exhibit both community structure and core-periphery structure. Our graph generation process allows us to easily control various graph properties such as graph size or density.

C Additional results

Learning curves To monitor the progress of pretraining, we evaluated the in-context learning performance of GraphPFN on several datasets from the GraphLand benchmark (Bazhenov et al., 2025) (see Section 5.1 for the evaluation procedure) every 100 steps during the first stage and every 20 steps during the second stage. We emphasize that these evaluations were used only for a post-hoc analysis of training progress: test set performance was not used for early stopping or any other

Table 6: Additional comparison of finetuned GraphPFN with finetuned LimiX with randomly initialized graph adapters (GA).

	artnet-exp	city-reviews	tolokers-2	artnet-views	avazu-ctr	city-roads-M	hm-prices	twitch-views
GraphPFN (FT)	50.64 ± 1.27	80.49 ± 0.16	60.90 ± 0.98	63.90 ± 0.12	34.34 ± 0.50	66.04 ± 0.41	79.26 ± 0.43	78.17 ± 0.13
Limix + GA (FT)	46.19 ± 0.21	78.94 ± 0.35	48.55 ± 1.77	53.84 ± 0.59	26.78 ± 0.16	63.41 ± 0.59	69.44 ± 0.48	74.01 ± 1.41

Table 7: The key statistics of the considered graph datasets.

name	# nodes	# edges	# features	mean degree	task	# classes	homophily	feature type
artnet-exp	50,405	280,348	75	11.1	cls.	2	no	tabular
artnet-views	50,405	280,348	50	11.1	reg.	–	no	tabular
avazu-ctr	76,269	10,984,077	260	288.0	reg.	–	no	tabular
city-reviews	148,801	1,165,415	37	15.7	cls.	2	yes	tabular
city-roads-M	57,073	107,104	26	3.8	reg.	–	yes	tabular
hm-prices	46,563	10,730,995	41	460.9	reg.	–	no	tabular
tolokers-2	11,758	519,000	16	88.3	cls.	2	no	tabular
twitch-views	168,114	6,797,557	4	80.9	reg.	–	no	tabular
amazon-ratings	24,492	93,050	300	7.6	cls.	5	no	text-based
facebook	22,470	170,823	128	15.2	cls.	4	yes	text-based
pubmed	19,717	44,324	500	4.5	cls.	3	yes	text-based
questions	48,921	153,540	301	6.3	cls.	2	no	text-based
wiki-cs	11,701	215,603	300	36.9	cls.	10	yes	text-based

form of model selection. Figure 2 shows that the first stage enables GraphPFN to quickly reach reasonable performance on most datasets. On datasets such as artnet-views, tolokers-2, and twitch-views, where the performance gap between the graph-agnostic LimiX and GraphPFN is largest, most of the gap is closed within the first 1-3K gradient steps. Nevertheless, the second stage consistently improves performance across all datasets and is essential for achieving strong final performance of GraphPFN.

LimiX + GA Since GraphPFN achieves strong performance in the finetuning setting, one may hypothesize that the performance comes solely from the powerful graph adapters, but not from the pretraining procedure. To test this hypothesis, we consider the following model. We start with LimiX and add graph adapters, so the architecture exactly matches that of GraphPFN. But instead of using pretrained weights for graph adapters, we employ random weights. In order to stabilize training, we initialized the last layers in all graph adapters with zeros, ensuring that the random initialization does not break the model. After that, we finetune this model following exactly the same protocol as GraphPFN. The results of this model and a comparison with GraphPFN are presented in Table 6. One can see that using random adapters instead of pretrained ones leads to significant drops in performance, supporting the importance of pretraining for achieving the strong performance of the finetuned GraphPFN.

D Additional details on experimental setup

Data Table 7 presents key statistics of datasets used in our main experiments. Importantly, the selected datasets cover diverse domains, spanning both homophilic and heterophilic structures, diverse feature types (both tabular and text-based) and tasks (both classification and regression).

PCA Since some datasets (specifically, amazon-ratings and questions) with text-embedding features have relatively high feature dimensionality, which prevented us from directly finetuning on these datasets, we applied PCA to reduce the feature dimensionality to 64. We did not apply PCA to the pubmed dataset, since, in our preliminary experiments, doing so led to degraded performance. As a result, GraphPFN ran out of memory during finetuning on pubmed, and we therefore report its in-context learning (ICL) performance for this dataset.

Ensembling For G2T-FM and GraphPFN, we report test-time ensembling results. Specifically, we evaluate the same model under 10 different random seeds and average the resulting predictions. The diversity among ensemble members is induced by several sources of stochasticity: (1) internal randomness of the TFM model (e.g., random positional encodings), (2) random shuffling of labels and input features, (3) PEARL in G2T-FM, and (4) random features in GraphPFN. In G2T-FM, we do not apply feature shuffling, because it degraded performance on some datasets in our preliminary experiments. When ensembling is combined with finetuning, we apply ensembling at every evaluation

step, including validation evaluation for early stopping and the final test evaluation. However, during optimization we use a single randomly sampled ensemble member to compute predictions for each optimizer step.

E LLM USAGE

LLMs have been used for proofreading and minor stylistic editing of the paper; the authors are responsible for all the content.

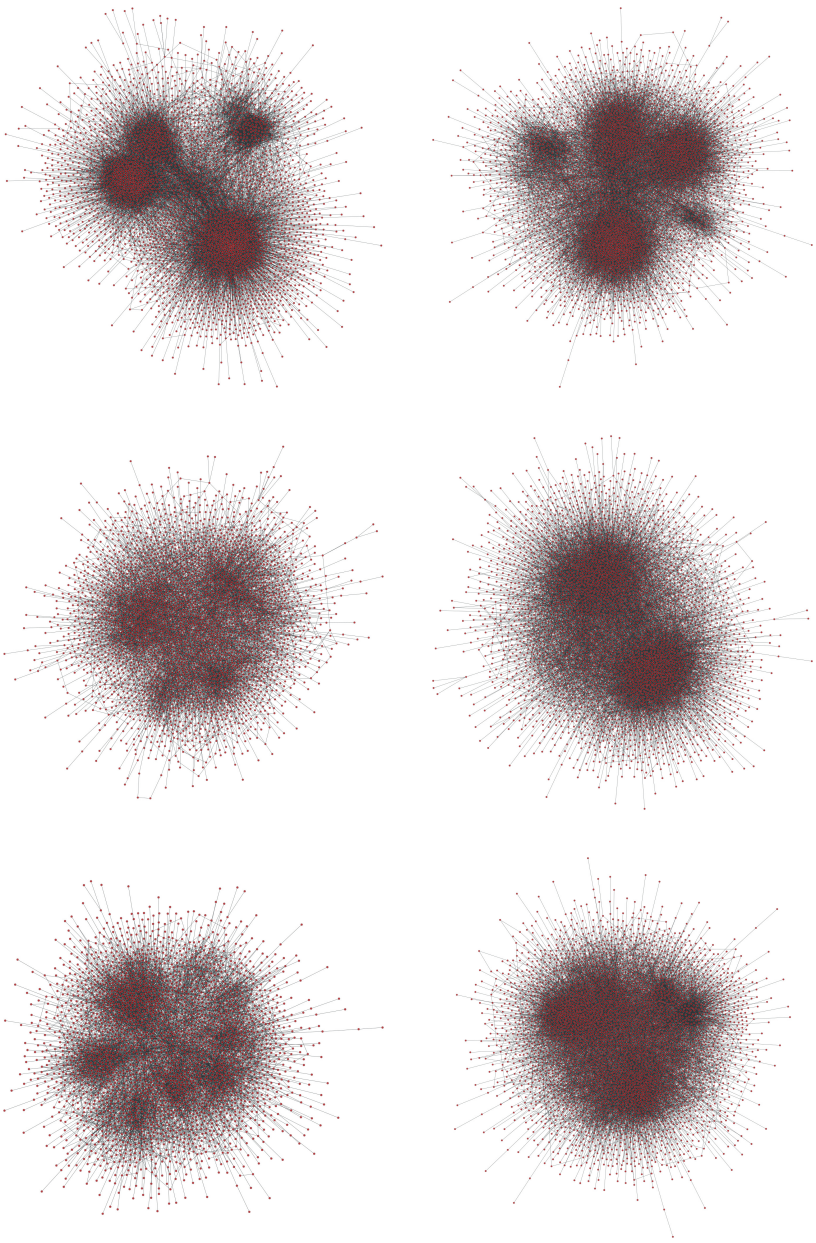


Figure 3: Example visualizations of graphs from our prior.