A Fair Evaluation of Graph Foundation Models for Node Property Prediction

Oleg Platonov [1 2] **Gleb Bazhenov** [1 2] **Dmitry Eremeev** [1 2] **Liudmila Prokhorenkova** [2]

Abstract

Due to the wide use of graph-structured data in different fields of industry and science, the development of Graph Foundation Models (GFMs) has recently attracted a lot of attention. While many different types of models are called GFMs, particular interest has been paid to GFMs designed for node property prediction tasks, which is one of the most popular settings in Graph ML with lots of real-world applications from fraud detection in financial and social networks to recommendation systems for e-commerce and user-generated content platforms. While a number of GFMs for this task have been recently proposed, the field has not converged to a unified evaluation setting, and different works evaluate their models in widely different ways, preventing reliable comparison of GFMs with each other and with other types of models. In this work, we conduct a fair and rigorous reevaluation of 9 recent GFMs for node property prediction, comparing them to strong Graph Neural Network (GNN) baselines. We find that, among these GFMs, only the most recent ones based on the Prior-data Fitted Networks paradigm outperform well-tuned GNNs in predictive performance, although at a higher inference cost.

1. Introduction

Graphs are a natural way to represent many types of data, and thus Graph Machine Learning (i.e., applying machine learning to graph-structured data) has recently gained a lot of attention. In the past decade, message-passing Graph Neural Networks (GNNs) (Duvenaud et al., 2015; Kipf & Welling, 2017; Gilmer et al., 2017; Hamilton et al., 2017) have emerged as a particularly effective approach to many tasks in Graph Machine Learning and become the default model type in the field. One of the most common tasks in Graph Machine Learning, and one in which GNNs have been particularly successful, is node property prediction, i.e., predicting unknown labels of nodes in a large graph from node features, graph structure, and known node labels. This task attracts a lot of attention as it has many impactful real-world applications: large-scale industrial use cases of node property prediction include applying GNNs to social, financial, co-purchasing, content-similarity, and transportation networks to solve tasks like fraud detection, churn prediction, content or product search and recommendation, click-through rate prediction, estimated time of arrival prediction, and many others (Ying et al., 2018; Wang et al., 2019a; Chen et al., 2021; Derrow-Pinion et al., 2021; Fey et al., 2023; De Nadai et al., 2024; Borisyuk et al., 2024; Zhao et al., 2025; Bazhenov et al., 2025).

GNNs are typically trained from scratch for each specific task. However, recently some machine learning fields have achieved success in developing Foundation Models — large multi-purpose pretrained models capable of solving many different tasks. Thus, while classic GNNs currently remain the dominant approach to Graph Machine Learning, there have recently been many attempts at developing Graph Foundation Models (GFMs). While different types of GFMs can support different tasks and data domains, due to high interest in node property prediction applications, a number of GFMs have been designed specifically for node property prediction across various datasets (Liu et al., 2024; Li et al., 2024; Xia et al., 2024; Xia & Huang, 2024; Zhao et al., 2024; Yu et al., 2025; Wang et al., 2025; He et al., 2025; Finkelshtein et al., 2025; Lachi et al., 2025; Eremeev et al., 2025; 2026; Choi et al., 2025; 2026; Hayler et al., 2025). However, despite many GFMs for node property prediction being developed, the community has not currently adopted a unified evaluation setting for these models and different works perform evaluation in widely different ways. This prevents reliable comparison of GFMs with each other and with other model types. Further, many evaluation setups in the literature have such issues as selecting narrow and unrepresentative sets of datasets and using weak baselines.

To obtain a reliable assessment of the current state of the field, we conduct a large empirical study that reevaluates 9 recent GFMs for node property prediction in a more reliable and realistic setting. We find that the predictive performance of different models differs drastically, and most of them

[1]HSE University [2]Yandex Research. Correspondence to: Oleg Platonov <olegplatonov@yandex-team.ru>.

The Workshop on Graph Foundation Models at the 43rd International Conference on Machine Learning (ICML 2026), Seoul, South Korea. Copyright 2026 by the author(s).

cannot outperform properly tuned GNNs. Only several recent GFMs based on the Prior-data Fitted Networks (PFNs) paradigm (Müller et al., 2022) (this paradigm first gained success in developing Foundation Models for tabular data (Hollmann et al., 2023; 2025)) achieve strong results and are able to outperform GNNs, although these models also come with significantly higher computational costs of inference.

2. Background

The term Foundation Model refers to a machine learning model that can be applied to a wide range of tasks with no or minimal adaptation (Bommasani et al., 2021). Such models are commonly pretrained on large amounts of data and then applied to downstream tasks with relatively light fine-tuning or even without any fine-tuning at all (e.g., in the in-context learning regime). Foundation Models have achieved great success in the fields of Natural Language Processing, Computer Vision, and Audio Processing, transforming these fields and becoming default approaches in them. This has led to many attempts to design Foundation Models for other types of data, including graph-structured data. However, we note that graph-structured data differs significantly from text, image, video, or audio data in that it can arguably be better viewed not as a single data domain, but rather a general way to represent data from different domains. Indeed, there is not much in common between graphs representing social networks, road networks, databases, ontologies, molecules, connectomes, protein-protein interaction networks, and gene regulatory networks (and this is just a small set of examples of data types commonly represented as graphs). The practical tasks that these graphs are used to solve are also very different and include predicting properties of graph components (nodes, edges, or more complex graph substructures) or of entire graphs, generating graphs or their components, and even predicting interactions between multiple graphs (e.g., molecules). Further, even a single task type can have very different meanings and represent entirely different real-world problems in different graphs (e.g., predicting whether a user is a fraudster in a social network and whether an accident is likely on a road segment in a transportation network are both node property prediction tasks) and even in the same graph (e.g., predicting whether a user is a fraudster in a social network and estimating the likelihood of a user leaving the platform, i.e., churn prediction, in the same social network). This has led to the natural result that, while many models currently proposed in the literature are called GFMs, they all target different types of graphs and/or different tasks. For example, considering data domain, there are GFMs meant only for molecular graphs (Beaini et al., 2024), only for knowledge graphs (Galkin et al., 2024), or only for road graphs (Wang et al., 2023). Further, considering task type, there are GFMs meant only for node property prediction (Eremeev et al., 2025; 2026), only for graph property prediction (Frasca et al., 2024), or only for link prediction (Dong et al., 2024). We thus argue that the term GFM is not well-defined in itself and encourage researchers to explicitly clarify the scope of applicability of their model when calling it a GFM.

In this work, we focus on GFMs for node property prediction. Specifically, in node property prediction, given a graph and labels of some of the nodes, the model should predict labels for the rest of the nodes. The labels can be either categorical (node classification) or continuous (node regression). This task appears in many real-world applications, and many GFMs for this task have been proposed. The key challenge for GFMs dealing with node property prediction is that graphs can come from vastly different domains (e.g., social networks, co-purchasing networks, road networks). This implies that the graphs can have different node feature spaces and node target spaces, i.e., have features and targets with different dimensions, distributions, meanings, and relations to each other. Thus, a successful GFM for node property prediction must be able to effectively adapt to graphs with different node feature and target spaces. The GFMs being proposed in the literature differ significantly in their mechanisms for this adaptation — we discuss some of these mechanisms in Appendix A. However, we note that the field of tabular deep learning faces the same challenge of having different feature and target spaces for different datasets, and in this field several successful Foundation Models (Hollmann et al., 2023; 2025; Qu et al., 2025; 2026; Zhang et al., 2025) have recently been created following the Prior-data Fitted Networks (PFNs) paradigm (Müller et al., 2022). This paradigm trains Transformer-based (Vaswani et al., 2017) models on a large number of synthetic datasets to learn to process the entire dataset and adapt to the new feature and target spaces in a single forward pass and thus make predictions in the in-context learning regime (we describe the PFNs paradigm in more detail in Appendix B). Several recent works (Eremeev et al., 2025; 2026; Choi et al., 2025; 2026; Hayler et al., 2025) have adapted the PFNs paradigm to GFMs for node property prediction.

3. Experiments

3.1. Datasets

Recently, there has been a lot of criticism of datasets commonly used in Graph ML due to their questionable practical relevance, narrow domain coverage, bugs in the data collection process, and other issues (Platonov et al., 2023b; Li et al., 2023; Bechler-Speicher et al., 2025; Bazhenov et al., 2025) (we provide more details on these issues in Appendix D). Thus, for our experiments, we use 10 datasets from the recently introduced GraphLand benchmark (Bazhenov et al., 2025) that represents real-world applications of node property prediction and was designed to address the issues of prior datasets. We use the `RL` data splits. To aggregate results across multiple datasets, we compute

average model rank and average normalized score (a normalized score is calculated by linearly transforming metrics obtained on a specific dataset so that the worst model achieves a value of 0 and the best model achieves a value of 1).

3.2. Models

Graph Neural Networks As noted in the broader ML literature (Lipton & Steinhardt, 2018), baseline tuning often receives less attention than tuning of newly proposed models, and Graph ML research is not immune to this tendency. It is often reported that newer methods outperform classic GNNs, however, a number of works over the years have performed more fair reevaluations of classic GNNs (Shchur et al., 2018; Errica et al., 2020; Platonov et al., 2023b; Tönshoff et al., 2023; Luo et al., 2024; 2025; Bazhenov et al., 2025) in different settings and found that their performance can be substantially improved by performing proper hyperparameter search and adding standard deep learning building blocks such as skip-connections (He et al., 2016) and normalizations (Ioffe & Szegedy, 2015; Ba et al., 2016). In particular, Platonov et al. (2023b) proposed stronger GNN variants with architectural improvements, and Luo et al. (2024) recently provided a relatively simple but strong evaluation protocol for node classification with GNNs including both hyperparameter search and architectural improvements. In our work, to provide a fair model comparison, we use two versions of strong GNNs with improved backbones — those from Platonov et al. (2023b) and those from Luo et al. (2024). We conduct extensive hyperparameter search for them: 100 trials with the TPE algorithm (Bergstra et al., 2011; Watanabe, 2023). More specifically, we evaluate the improved versions of 4 GNN models: GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2018), and Local Graph Transformer (LGT) (Shi et al., 2021; Platonov et al., 2023b; Platonov & Prokhorenkova, 2026). We provide more details on these models in Appendix C.

Graph Foundation Models We evaluate the following 9 GFMs: OpenGraph (Xia et al., 2024), AnyGraph (Xia & Huang, 2024), GCOPE (Zhao et al., 2024), SAMGPT (Yu et al., 2025), MDGFM (Wang et al., 2025), TS-GNN (Finkelshtein et al., 2025), G2T-FM (Eremeev et al., 2025), TAG (Hayler et al., 2025), GraphPFN (Eremeev et al., 2026). We use their open-source official implementations for all models. G2T-FM, TAG, and GraphPFN are based on the PFNs paradigm. GraphPFN uses a custom graph-native architecture, while G2T-FM and TAG directly use unmodified Tabular Foundation Models as backbones. For G2T-FM, we evaluate the LimiX-16M (Zhang et al., 2025) backbone, which the authors of G2T-FM recommend as the best performing one. For TAG, we evaluate both the TabPFNv2 (Hollmann et al., 2025) and LimiX-16M (Zhang et al., 2025) backbones, which achieve the best results in the original TAG paper. GCOPE, SAMGPT, and MDGFM only support the fine-tuning (FT) regime. OpenGraph, AnyGraph, TS-GNN, and all PFN-based GFMs support in-context learning (ICL), and the official implementations of G2T-FM and GraphPFN also support fine-tuning. We evaluate each model using all the supported regimes. Note that some of the considered models do not support node regression — we evaluate these models only on node classification datasets.

3.3. Results: Predictive Performance

We describe the details of our experimental setup in Appendix E. Our experimental results are provided in Table 1. Based on them, we make the following observations:

- The results clearly separate GFMs into two groups, which correspond to PFN-based models and non-PFN-based models. Non-PFN-based GFMs always underperform well-tuned classic GNNs on the considered datasets, often by a substantial margin. In contrast, PFN-based GFMs typically outperform GNNs: they achieve the best results on most of the datasets, with classic GNNs getting in top-3 only on 2 out of the 10 datasets (and even on these 2 datasets, the first place belongs to GraphPFN).
- Comparing PFN-based GFMs and their different regimes and backbones to each other, we see that fine-tuning almost always improves the performance of these models (with the only exception being G2T-LimiX on `tolokers-2`), and these improvements are quite substantial. But even in the ICL regime, they always outperform all other GFMs, and also outperform all GNNs on the majority of datasets. The LimiX backbone is better than the TabPFNv2 backbone for TAG. In the ICL regime, when using the same LimiX backbone, G2T-FM and TAG (which both augment node features with graph-based information and then pass them directly to the tabular backbone) achieve close results (the same average rank, and G2T-FM slightly leads in the average normalized score). However, on 6 out of the 10 datasets, they are outperformed by GraphPFN (which directly incorporates graph message passing in the PFN architecture and thus natively supports graph-structured data) in the ICL regime. Further, in the fine-tuning regime, the lead of GraphPFN over G2T-FM significantly increases.
- GraphPFN in the fine-tuning regime is the best model overall, achieving the best results on all the 10 datasets (often outperforming all other models by a substantial margin).

3.4. Results: Computational Efficiency

In this subsection, we discuss the computational costs of GNNs and PFN-based GFMs. First, it is important to note that these models operate in very different regimes, making a comparison of their computational costs not straightforward. GNNs follow the classic paradigm of requiring relatively long training (hundreds to thousands of forward and backward passes) to optimize model parameters for each specific dataset and then performing relatively fast inference (a single forward pass) to make predictions for this dataset.

Table 1. Experimental results. Accuracy, AP, and R^2 are reported for mult. class., bin. class., and regression datasets, respectively. Average rank (lower is better) and average normalized score (higher is better) are computed either only across classification datasets (*rank (cls)* and *score (cls)*) or across all datasets (*rank (all)* and *score (all)*). The *Crit.* and *Class.* prefixes refer to improved GNNs from Platonov et al. (2023b) and Luo et al. (2024), respectively (by the first words of the paper titles). The best results are highlighted: first, second, third.

	mult. class.	bin. class.			regression									
	hm-categories	tolokers-2	city-reviews	artnet-exp	hm-prices	avazu-ctr	city-roads-M	city-roads-L	twitch-views	artnet-views	rank (cls) ↓	rank (all) ↓	score (cls) ↑	score (all) ↑
Crit.-GraphSAGE	64.03 ± 1.87	53.80 ± 0.62	77.54 ± 0.15	48.13 ± 0.58	75.02 ± 0.09	32.11 ± 0.49	59.69 ± 0.54	53.31 ± 0.41	71.37 ± 0.53	54.44 ± 0.20	11.00	9.00	83.17	24.48
Class.-GraphSAGE	69.09 ± 0.52	54.82 ± 0.46	78.40 ± 0.14	46.42 ± 0.38	73.39 ± 0.32	31.53 ± 0.19	60.15 ± 0.58	54.73 ± 0.25	73.52 ± 0.12	55.05 ± 0.08	10.00	7.90	85.09	29.47
Crit.-GCN	69.23 ± 0.12	53.52 ± 0.61	76.59 ± 0.35	45.46 ± 0.74	71.25 ± 0.57	31.48 ± 0.16	59.41 ± 0.40	52.06 ± 0.78	77.60 ± 0.06	57.53 ± 0.31	12.00	9.20	82.68	23.82
Class.-GCN	70.00 ± 0.66	54.68 ± 0.40	78.53 ± 0.08	48.15 ± 0.29	71.68 ± 0.50	30.98 ± 0.33	60.14 ± 0.16	55.34 ± 0.20	78.02 ± 0.13	58.87 ± 0.32	8.00	6.60	86.46	37.73
Crit.-GAT	73.46 ± 0.27	58.94 ± 0.91	78.42 ± 0.18	49.43 ± 0.19	75.43 ± 0.22	32.43 ± 0.36	59.68 ± 0.42	53.75 ± 0.49	75.26 ± 0.30	55.00 ± 0.26	5.25	5.90	91.54	42.97
Class.-GAT	70.05 ± 0.89	57.62 ± 0.85	78.40 ± 0.10	48.25 ± 0.39	67.17 ± 0.10	30.53 ± 0.31	60.22 ± 0.47	55.19 ± 0.40	73.63 ± 1.22	56.40 ± 0.26	7.00	7.20	88.61	30.47
Crit.-LGT	72.15 ± 0.71	56.20 ± 0.32	78.03 ± 0.13	49.15 ± 0.67	73.89 ± 0.47	31.44 ± 0.57	59.55 ± 2.02	54.12 ± 0.29	76.22 ± 0.08	53.89 ± 0.21	7.50	7.70	88.71	35.16
Class.-LGT	62.54 ± 0.36	56.94 ± 0.27	78.38 ± 0.27	48.04 ± 0.45	67.95 ± 0.28	30.09 ± 0.20	59.25 ± 0.26	54.71 ± 0.27	67.95 ± 0.18	54.24 ± 1.24	10.00	10.20	85.28	17.22
OpenGraph (ICL)	9.88 ± 0.76	40.43 ± 1.15	58.54 ± 1.60	15.41 ± 0.71	N/A	N/A	N/A	N/A	N/A	N/A	17.25	N/A	24.00	N/A
AnyGraph (ICL)	14.15 ± 1.64	29.92 ± 2.78	64.54 ± 1.67	13.50 ± 0.98	N/A	N/A	N/A	N/A	N/A	N/A	18.25	N/A	19.73	N/A
TS-GNN (ICL)	20.09 ± 1.29	38.54 ± 0.94	36.85 ± 15.65	20.44 ± 1.05	N/A	N/A	N/A	N/A	N/A	N/A	16.25	N/A	18.52	N/A
GCOPE (FT)	19.57 ± 0.12	28.35 ± 1.24	66.11 ± 0.74	15.41 ± 3.07	N/A	N/A	N/A	N/A	N/A	N/A	17.50	N/A	22.52	N/A
SAMGPT (FT)	14.64 ± 0.35	36.87 ± 1.04	46.36 ± 2.49	16.16 ± 0.87	N/A	N/A	N/A	N/A	N/A	N/A	17.00	N/A	17.46	N/A
MDGFM (FT)	9.08 ± 1.00	31.24 ± 1.71	31.24 ± 8.50	15.52 ± 1.46	N/A	N/A	N/A	N/A	N/A	N/A	18.75	N/A	3.36	N/A
G2T-LimiX (ICL)	58.05 ± 0.28	61.60 ± 0.18	78.98 ± 0.44	48.42 ± 0.78	76.14 ± 0.08	32.70 ± 0.14	65.16 ± 0.07	56.62 ± 0.13	71.31 ± 0.06	61.58 ± 0.08	6.25	5.20	87.59	51.71
TAG-TabPFNv2 (ICL)	57.17 ± 0.46	59.33 ± 1.00	77.38 ± 0.29	47.87 ± 0.39	N/A	N/A	N/A	N/A	N/A	N/A	11.00	N/A	84.47	N/A
TAG-LimiX (ICL)	58.69 ± 0.39	59.46 ± 1.05	78.87 ± 0.13	50.19 ± 0.46	N/A	N/A	N/A	N/A	N/A	N/A	6.25	N/A	87.31	N/A
GraphPFN (ICL)	55.25 ± 0.75	61.29 ± 0.12	80.25 ± 0.05	51.79 ± 0.11	77.88 ± 0.09	31.63 ± 0.06	64.85 ± 0.13	58.52 ± 0.12	73.20 ± 0.08	62.79 ± 0.08	5.50	4.70	89.10	60.39
G2T-LimiX (FT)	65.59 ± 0.21	59.75 ± 1.14	80.65 ± 0.05	50.39 ± 0.19	77.37 ± 0.17	34.09 ± 0.37	66.29 ± 0.12	62.21 ± 0.19	74.91 ± 0.06	63.24 ± 0.07	4.25	3.40	91.02	73.69
GraphPFN (FT)	78.67 ± 0.20	62.80 ± 0.39	80.90 ± 0.03	53.49 ± 0.81	81.06 ± 0.24	35.07 ± 0.34	67.30 ± 0.21	64.12 ± 0.27	79.00 ± 0.14	65.35 ± 0.06	1.00	1.00	100.00	100.00

Table 2. Time required for: hyperparameter tuning (Tun), a single training run with the best hyperparameters (Tr), a single inference run (including inference-time ensembling for PFN-based models) with the best hyperparameters (Inf) using NVIDIA Tesla A100 80GB GPU. Note that for GNNs, the models with the best hyperparameters can be widely different in size across GNN types and datasets.

	tolokers-2			city-reviews			artnet-views			city-roads-M			city-roads-L			hm-prices		
	Tun	Tr	Inf	Tun	Tr	Inf	Tun	Tr	Inf	Tun	Tr	Inf	Tun	Tr	Inf	Tun	Tr	Inf
Crit.-GraphSAGE	2.16h	1.63m	0.05s	14.02h	11.70m	0.23s	4.08h	2.93m	0.03s	2.22h	1.44m	0.03s	6.70h	3.66m	0.06s	1.00d	15.44m	0.09s
Class.-GraphSAGE	17.49h	7.23m	0.06s	1.81d	15.38m	0.25s	20.68h	11.94m	0.06s	1.68h	35.74s	0.02s	6.82h	3.29m	0.06s	7.57d	1.69h	2.11s
Crit.-GCN	2.70h	1.47m	0.04s	7.11h	3.44m	0.13s	1.93h	33.73s	0.02s	1.77h	35.47s	0.02s	4.07h	4.13m	0.03s	19.59h	7.21m	0.05s
Class.-GCN	1.07d	7.84m	0.07s	1.88d	23.24m	0.22s	11.20h	4.62m	0.02s	2.59h	1.37m	0.02s	11.98h	8.24m	0.10s	1.59d	53.72m	0.44s
Crit.-GAT	3.22h	2.23m	0.05s	19.76h	14.30m	0.24s	2.77h	1.69m	0.02s	2.70h	1.77m	0.02s	8.18h	5.56m	0.04s	2.42d	36.54m	0.18s
Class.-GAT	1.45d	15.62m	0.06s	2.50d	35.30m	0.33s	19.74h	12.68m	0.05s	1.70h	54.22s	0.02s	17.75h	11.31m	0.11s	2.47h	21.08m	1.87s
Crit.-LGT	3.24h	2.03m	0.04s	20.78h	16.96m	0.30s	5.46h	4.64m	0.04s	4.63h	2.87m	0.04s	5.46h	2.32m	0.02s	3.99d	1.05h	0.37s
Class.-LGT	1.79d	9.23m	0.06s	1.89d	1.57h	1.19s	1.16d	14.16m	0.10s	5.72h	1.46m	0.02s	18.12h	20.08m	0.13s	1.20h	23.18m	0.84s
G2T-LimiX	24.74m	2.56m	8.93s	13.02h	1.22h	6.63m	2.40h	22.65m	1.26m	1.35h	8.82m	38.27s	19.10h	1.42h	7.92m	6.87h	35.28m	6.45m
TAG-TabPFNv2	N/A	N/A	1.90m	N/A	N/A	32.77m	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
TAG-LimiX	N/A	N/A	1.84m	N/A	N/A	36.60m	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
GraphPFN	40.25m	3.78m	6.01s	1.08d	2.50h	1.52m	8.49h	1.09h	29.49s	3.16h	17.05m	15.72s	21.01h	2.16h	58.52s	1.32d	3.49h	2.47m

Further, as we have discussed, obtaining strong results from GNNs requires extensive hyperparameter tuning, i.e., performing many training runs (in our work, we perform 100 training runs). The development and usage process of PFN-based GFMs is notably different. First, these models require a relatively expensive pretraining stage to obtain the base model. However, this pretraining is performed once by the model creators and then the same base model can be applied to different datasets, thus the pretraining costs are not relevant to the costs of model usage for a particular downstream application. After pretraining, PFNs can operate in the ICL regime without any dataset-specific training. However, as we have shown, dataset-specific fine-tuning of PFNs typically improves their performance. Thus, dataset-specific training is still often desirable. If dataset-specific training is performed, it requires selecting training hyperparameters, i.e., performing dataset-specific hyperparameter tuning. However, since the PFN architecture is completely fixed after pretraining, far fewer hyperparameters remain to be tuned than for GNNs trained from scratch (in our work, we perform 10 training runs to select the learning rate).

For a representative subset of datasets, we provide the time required for the entire hyperparameter tuning, a single training run, and a single inference run of GNNs and PFN-based GFMs in Table 2 (we provide a similar table for VRAM usage in Appendix F). Note that if a PFN-based GFM is used in the ICL regime, then only the inference resources are relevant, since no dataset-specific training is performed.

First, we emphasize that in industrial settings, which are arguably the most impactful real-world applications of node property prediction, inference time is what generally matters the most, since it is typically constrained by the specifics of the application such as the need for (almost) real-time prediction. We observe that in inference efficiency, GNNs strongly outcompete PFN-based GFMs: the inference time of GNNs is typically below 1 second, while PFN-based GFMs require several seconds to several minutes, depending on the dataset (with inference time of TAG models sometimes reaching half an hour). The memory requirements of GNN inference are also typically significantly below those of PFN-based GFMs. However, when model development time rather than inference efficiency is critical, PFNs can be preferable to GNNs, as they can make predictions in the ICL regime without any dataset-specific fine-tuning, and even their fine-tuning can sometimes take less time than developing GNNs since PFNs require less hyperparameter search.

4. Conclusion

We performed a fair evaluation of a diverse set of GFMs on real-world node property prediction datasets and found that only PFN-based GFMs outcompete well-tuned classic GNNs, although at the cost of more expensive inference.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2623–2631, 2019.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

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

Beaini, D., Huang, S., Cunha, J. A., Li, Z., Moisescu-Pareja, G., Dymov, O., Maddrell-Mander, S., McLean, C., Wenkel, F., Müller, L., et al. Towards foundational models for molecular learning on large-scale multi-task datasets. In *International Conference on Learning Representations*, 2024.

Bechler-Speicher, M., Finkelshtein, B., Frasca, F., Müller, L., Tönshoff, J., Siraudin, A., Zaverkin, V., Bronstein, M. M., Niepert, M., Perozzi, B., Galkin, M., and Morris, C. Position: Graph learning will lose relevance due to poor benchmarks. In *International Conference on Machine Learning*, 2025.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems*, 24, 2011.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Borisyuk, F., He, S., Ouyang, Y., Ramezani, M., Du, P., Hou, X., Jiang, C., Pasumarthy, N., Bannur, P., Tiwana, B., Liu, P., Dangi, S., Sun, D., Pei, Z., Shi, X., Zhu, S., Shen, Q., Lee, K.-H., Stein, D., Li, B., Wei, H., Ghoting, A., and Ghosh, S. LiGNN: Graph neural networks at LinkedIn. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4793–4803, 2024.

Chen, H., Wang, L., Lin, Y., Yeh, C.-C. M., Wang, F., and Yang, H. Structured graph convolutional networks with stochastic masks for recommender systems. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pp. 614–623, 2021.

Choi, J., Kang, W., Kim, M., Kim, J., and Park, N. Can TabPFN compete with gnns for node classification via graph tabularization? *arXiv preprint arXiv:2512.08798*, 2025.

Choi, J., Kim, J., Kang, W., and Park, N. Learning posterior predictive distributions for node classification from synthetic graph priors. In *International Conference on Learning Representations*, 2026.

De Nadai, M., Fabbri, F., Gigioli, P., Wang, A., Li, A., Silvestri, F., Kim, L., Lin, S., Radosavljevic, V., Ghael, S., Nyhan, D., Bouchard, H., Lalmas, M., and Damianou, A. Personalized audiobook recommendations at Spotify through graph neural networks. In *Companion Proceedings of the ACM Web Conference 2024*, pp. 403–412, 2024.

Derrow-Pinion, A., She, J., Wong, D., Lange, O., Hester, T., Perez, L., Nunkesser, M., Lee, S., Guo, X., Wiltshire, B., et al. ETA prediction with graph neural networks in Google maps. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 3767–3776, 2021.

Dong, K., Mao, H., Guo, Z., and Chawla, N. V. Universal link predictor by in-context learning on graphs. *arXiv preprint arXiv:2402.07738*, 2024.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. *Advances in Neural Information Processing Systems*, 28, 2015.

Eremeev, D., Bazhenov, G., Platonov, O., Babenko, A., and Prokhorenkova, L. Turning tabular foundation models into graph foundation models. *arXiv preprint arXiv:2508.20906*, 2025.

Eremeev, D., Platonov, O., Bazhenov, G., Babenko, A., and Prokhorenkova, L. GraphPFN: A prior-data fitted graph foundation model. In *International Conference on Machine Learning*, 2026.

Errica, F., Podda, M., Bacciu, D., and Micheli, A. A fair comparison of graph neural networks for graph classification. In *International Conference on Learning Representations*, 2020.

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *Representation Learning on Graphs and Manifolds Workshop at The Seventh International Conference on Learning Representations*, 2019.

Fey, M., Hu, W., Huang, K., Lenssen, J. E., Ranjan, R., Robinson, J., Ying, R., You, J., and Leskovec, J. Relational deep learning: Graph representation learning on relational databases. *arXiv preprint arXiv:2312.04615*, 2023.

Finkelshtein, B., Ceylan, İ. İ., Bronstein, M., and Levie, R. Equivariance everywhere all at once: A recipe for graph foundation models. *Advances in Neural Information Processing Systems*, 2025.

Frasca, F., Jogl, F., Eliasof, M., Ostrovsky, M., Schönlieb, C.-B., Gärtner, T., and Maron, H. Towards foundation models on graphs: An analysis on cross-dataset transfer of pretrained GNNs. *arXiv preprint arXiv:2412.17609*, 2024.

Galkin, M., Yuan, X., Mostafa, H., Tang, J., and Zhu, Z. Towards foundation models for knowledge graph reasoning. In *International Conference on Learning Representations*, 2024.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pp. 1263–1272, 2017.

Grinsztajn, L., Flöge, K., Key, O., Birkel, F., Jund, P., Roof, B., Manium, M., Hoo, S. B., Bühler, M., Garg, A., Safaric, D., Robertson, J., Jäger, B., Alessi, S., Hayler, A., Moroshan, V., Purucker, L., Singer, P., Arazi, A., Siems, J., Metzen, J. H., Grab, G., Erickson, N., Guo, S., Kalfon, E., Bing, S., Salinas, D., Cornu, C., Wehrhahn, L. C., Kriuchkova, D., Kaya, K., Sidhoum, L., Salmon, M., Chen, J., Hulsebos, M., LeCun, Y., Müller, S., Schölkopf, B., Gambhir, S., Hollmann, N., and Hutter, F. TabPFN-3: Technical report. *arXiv preprint arXiv:2605.13986*, 2026.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30, 2017.

Hayler, A., Huang, X., Ceylan, I. I., Bronstein, M., and Finkelshtein, B. Bringing graphs to the table: Zero-shot node classification via tabular foundation models. *arXiv preprint arXiv:2509.07143*, 2025.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

He, Y., Sui, Y., He, X., and Hooi, B. 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.

Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. TabPFN: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations*, 2023.

Hollmann, N., Müller, S., Purucker, L., Krishnakumar, A., Körfer, M., Hoo, S. B., Schirrmeister, R. T., and Hutter, F. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

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

Lachi, D., Azabou, M., Arora, V., and Dyer, E. L. GraphFM: A generalist graph transformer that learns transferable representations across diverse domains. *Transactions on Machine Learning Research (TMLR)*, 2025.

Li, Y., Xiong, M., and Hooi, B. GraphCleaner: Detecting mislabelled samples in popular graph learning benchmarks. In *International Conference on Machine Learning*, pp. 20195–20209, 2023.

Li, Y., Wang, P., Li, Z., Yu, J. X., and Li, J. ZeroG: Investigating cross-dataset zero-shot transferability in graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1725–1735, 2024.

Lipton, Z. C. and Steinhardt, J. Troubling trends in machine learning scholarship. *arXiv preprint arXiv:1807.03341*, 2018.

Liu, H., Feng, J., Kong, L., Liang, N., Tao, D., Chen, Y., and Zhang, M. One for all: Towards training one graph model for all classification tasks. In *International Conference on Learning Representations*, 2024.

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

Luo, Y., Shi, L., and Wu, X.-M. Classic GNNs are strong baselines: Reassessing GNNs for node classification. *Advances in Neural Information Processing Systems*, 37: 97650–97669, 2024.

Luo, Y., Shi, L., and Wu, X.-M. Can classic GNNs be strong baselines for graph-level tasks? Simple architectures meet excellence. In *International Conference on Machine Learning*, 2025.

Mironov, M. and Prokhorenkova, L. Revisiting graph homophily measures. In *Learning on Graphs Conference*, 2024.

Müller, S., Hollmann, N., Arango, S. P., Grabocka, J., and Hutter, F. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022.

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

Nguyen, T. and Grover, A. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling. In *International Conference on Machine Learning*, 2022.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

Platonov, O. and Prokhorenkova, L. Cluster attention for graph machine learning. *arXiv preprint arXiv:2604.07492*, 2026.

Platonov, O., Kuznedelev, D., Babenko, A., and Prokhorenkova, L. Characterizing graph datasets for node classification: Homophily-heterophily dichotomy and beyond. *Advances in Neural Information Processing Systems*, 36:523–548, 2023a.

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

Qu, J., Holzmüller, D., Varoquaux, G., and Morvan, M. L. TabICL: A tabular foundation model for in-context learning on large data. In *International Conference on Machine Learning*, 2025.

Qu, J., Holzmüller, D., Varoquaux, G., and Morvan, M. L. TabICLv2: A better, faster, scalable, and open tabular foundation model. In *International Conference on Machine Learning*, 2026.

Rubachev, I., Kotelnikov, A., Kartashev, N., and Babenko, A. On finetuning tabular foundation models. *arXiv preprint arXiv:2506.08982*, 2025.

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

Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., and Sun, Y. Masked label prediction: Unified message passing model for semi-supervised classification. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, 2021.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

Stoll, T., Qian, C., Finkelshtein, B., Parviz, A., Weber, D., Frasca, F., Shavit, H., Siraudin, A., Mielke, A., Anastacio, M., Müller, E., Bechler-Speicher, M., Bronstein, M., Galkin, M., Hoos, H., Niepert, M., Perozzi, B., Tönshoff, J., and Morris, C. GraphBench: Next-generation graph learning benchmarking. *arXiv preprint arXiv:2512.04475*, 2025.

Tönshoff, J., Ritzert, M., Rosenbluth, E., and Grohe, M. Where did the gap go? Reassessing the long-range graph benchmark. In *Learning on Graphs Conference*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Wang, D., Lin, J., Cui, P., Jia, Q., Wang, Z., Fang, Y., Yu, Q., Zhou, J., Yang, S., and Qi, Y. A semi-supervised graph attentive network for financial fraud detection. In *2019 IEEE international conference on data mining (ICDM)*, pp. 598–607. IEEE, 2019a.

Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., Xiao, T., He, T., Karypis, G., Li, J., and Zhang, Z. Deep graph library: A graph-centric, highly-performant package for graph neural networks. In *Representation Learning on Graphs and Manifolds Workshop at The Seventh International Conference on Learning Representations*, 2019b.

Wang, S., Wang, B., Shen, Z., Deng, B., and Kang, Z. Multi-domain graph foundation models: Robust knowledge transfer via topology alignment. In *International Conference on Machine Learning*, 2025.

Wang, X., Wang, D., Chen, L., Wang, F.-Y., and Lin, Y. Building transportation foundation model via generative graph transformer. In *2023 IEEE 26th International conference on intelligent transportation systems (ITSC)*, pp. 6042–6047. IEEE, 2023.

Wang, Y., Zhang, X., Yu, H., Hao, M., Ren, G., Yuan, H., Mao, L., Zhang, Y., Yuan, C., and Cui, P. LimiX-2M: Mitigating low-rank collapse and attention bottlenecks in tabular foundation models. In *International Conference on Machine Learning*, 2026.

Watanabe, S. Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance. *arXiv preprint arXiv:2304.11127*, 2023.

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

Xia, L., Kao, B., and Huang, C. OpenGraph: Towards open graph foundation models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 974–983, 2018.

Yu, X., Gong, Z., Zhou, C., Fang, Y., and Zhang, H. SAMGPT: Text-free graph foundation model for multi-domain pre-training and cross-domain adaptation. In *Proceedings of the ACM Web Conference 2025*, pp. 1142–1153, 2025.

Zhang, X., Ren, G., Yu, H., Yuan, H., Wang, H., Li, J., Wu, J., Mo, L., Mao, L., Hao, M., Dai, N., Xu, R., Li, S., Zhang, T., He, Y., Wang, Y., Zhang, Y., Xu, Z., Li, D., Gao, F., Zou, H., Liu, J., Liu, J., Xu, J., Cheng, K., Li, K., Zhou, L., Li, Q., Fan, S., Lin, X., Han, X., Li, X., Lu, Y., Xue, Y., Jiang, Y., Wang, Z., Wang, Z., and Cui, P. LimiX: Unleashing structured-data modeling capability for generalist intelligence. *arXiv preprint arXiv:2509.03505*, 2025.

Zhao, H., Chen, A., Sun, X., Cheng, H., and Li, J. 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.

Zhao, T., Liu, Y., Kolodner, M., Montemayor, K., Ghazizadeh, E., Batra, A., Fan, Z., Gao, X., Guo, X., Ren, J., Park, S., Yu, P., Yu, J., Vij, S., and Shah, N. GiGL: Large-scale graph neural networks at Snapchat. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 5225–5236, 2025.

Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., and Koutra, D. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020.

A. Non-PFN-Based GFMs for Node Property Prediction

The key challenge for GFMs dealing with node property prediction is that graphs can come from vastly different domains (e.g., social networks, co-purchasing networks, road networks). This implies that the graphs can have different node feature spaces and node target spaces, i.e., have features and targets with different dimensions, distributions, meanings, and relations to each other. Thus, a successful GFM for node property prediction must be able to effectively adapt to graphs with different node feature and target spaces. The GFMs being proposed in the literature differ significantly in their mechanisms for this adaptation. In this section, we briefly describe a few common examples of these mechanisms. For tackling different node feature spaces, some GFMs construct additional synthetic edges based on node feature vector similarity and then discard node features (Xia et al., 2024), while others only work with text-attributed graphs, i.e. graphs with textual descriptions as node features, which allows using Large Language Models (LLMs) to project these features to a common latent space (Liu et al., 2024; Li et al., 2024; He et al., 2025). However, such approaches cannot work effectively with all graphs, as many real-world graphs come with diverse numerical and categorical features that cannot be effectively processed by current LLMs and are too valuable for the task to discard. Other approaches use dimensionality reduction techniques like SVD or PCA to project all features into a space of the same dimensionality (Xia & Huang, 2024; Zhao et al., 2024; Wang et al., 2025; Yu et al., 2025). However, the obtained spaces for different datasets are only matched in dimensionality, not in semantics. Another approach is to use a lightweight input module trained for each dataset from scratch (Lachi et al., 2025), which can work well in the fine-tuning regime, but does not allow making predictions in the in-context learning regime. Mechanisms for adapting to different target spaces also vary. Some methods cast node classification as link prediction relative to virtual class nodes (Xia et al., 2024; Xia & Huang, 2024), however, this does not support node regression. A different approach uses a lightweight output module trained for each dataset from scratch (Lachi et al., 2025), however, this does not support making predictions in the in-context learning regime.

B. Prior-Data Fitted Networks (PFNs)

The Prior-data Fitted Networks (PFNs) paradigm was introduced by Müller et al. (2022) (similar ideas concurrently appeared in (Nguyen & Grover, 2022) under the name of Transformer Neural Processes). The idea of PFNs is to train models to make predictions on previously unseen datasets in a single forward pass. PFNs perform in-context learning (ICL): rather than updating the model parameters for each new dataset, they use the context provided as input to make predictions without per-dataset training. In the PFNs paradigm, the input to the model is an entire dataset that is split into two parts: a set of training samples paired with their labels (the *context*), and a set of test samples without labels for which the predictions should be made (the *query*). During the forward pass, the model uses the context samples to make predictions for the query samples, thus performing ICL. At the end of the forward pass, the model directly outputs predictions for the query samples. PFNs use (variants of) the Transformer architecture (Vaswani et al., 2017). Each sample in the dataset is represented as one or more tokens. Samples interact through inter-sample attention that uses a specific attention pattern: context (training) samples are allowed to attend to all other context samples but not query samples, while query (test) samples are allowed to attend only to context samples and not to each other. This attention pattern means that predictions for each query sample are based only on this query sample and the context samples. A PFN is trained on a large number of diverse datasets to make its ICL mechanism work. At each training step, a PFN receives several new datasets, makes predictions for their query samples in a single forward pass, these predictions are used for loss computation, then the backward pass is performed to compute model parameter gradients, the parameters are updated based on these gradients concluding the training step, and the next training step uses a different set of datasets. To ensure a sufficient number and diversity of training datasets, these datasets are typically synthetically generated. A *prior distribution* over datasets is defined, and synthetic datasets are sampled from this distribution. The design of the prior distribution can strongly affect the predictive performance of the PFN. Müller et al. (2022) showed that the PFN training procedure trains the model to approximate the posterior predictive distribution under the chosen prior distribution, which provides a theoretical motivation for the PFNs paradigm. PFNs can also be viewed as a meta-learning (i.e., "learning how to learn") approach that learns how to perform in-context learning.

The PFNs paradigm was used to create successful Foundation Models for tabular data such as TabPFN (Hollmann et al., 2023; 2025; Grinsztajn et al., 2026), TabICL (Qu et al., 2025; 2026), and LimiX (Zhang et al., 2025; Wang et al., 2026). Inspired by this, several works have adopted this paradigm to create GFMs for node property prediction (Eremeev et al., 2025; 2026; Choi et al., 2025; 2026; Hayler et al., 2025). The earlier of these models — G2T-FM (Eremeev et al., 2025), TabPFN-GN (Choi et al., 2025), and TAG (Hayler et al., 2025) — cast graph node classification/regression tasks as tabular classification/regression tasks by augmenting node features with aggregated graph neighborhood features and graph structure information, then discarding the graph structure, and directly applying unchanged tabular Foundation Models to the obtained

feature tables. These three works mostly differ in the way they construct the additional graph-based features. Later PFN-based GFMs — GraphPFN (Eremeev et al., 2026) and NodePFN (Choi et al., 2026) — are significantly more involved: they create custom graph-native model architectures by augmenting standard PFN Transformers with graph neighborhood aggregation modules, design prior distributions over attributed graphs, and train these models on synthetic node property prediction datasets sampled from these prior distributions. Thus, these models handle graph-structured data natively instead of having to simplify it by converting to a table. GraphPFN and NodePFN differ in their model architecture and prior distribution designs, as well as in the fact that GraphPFN initializes non-graph-specific parts of the model from the Tabular Foundation Model LimiX-16M (Zhang et al., 2025), while NodePFN trains all parameters in the model from scratch. Note that in our work we do not evaluate TabPFN-GN, since there is no publicly available implementation of it, and NodePFN, since it has been shown by Eremeev et al. (2026) that NodePFN is strongly outperformed by GraphPFN (we hypothesize that this is due to some or all of the following: NodePFN uses an older architecture based on TabPFN while GraphPFN uses a newer architecture based on LimiX-16M which is in turn based on TabPFNv2, NodePFN is pretrained from scratch while GraphPFN initializes most of its parameters from LimiX-16M, and GraphPFN uses a significantly more complex prior distribution over graph datasets than NodePFN).

PFNs are designed specifically to be able to make predictions in the in-context learning regime. However, it was shown both for PFN-based Tabular Foundation Models (Rubachev et al., 2025) and Graph Foundation Models (Eremeev et al., 2025; 2026) that their prediction quality can often be improved by additionally fine-tuning them on the downstream dataset.

Our experimental results in Section 3 demonstrate that PFN-based GFMs are currently the best GFMs for node property prediction both in the in-context learning and fine-tuning regimes, and are the only current GFMs capable of outperforming well-tuned classic GNNs.

C. Improved GNNs

We believe the predictive performance of classic GNNs is often underestimated due to using weak GNN backbones. Many GNN implementations do not include such standard deep learning building blocks as skip-connections (He et al., 2016) and normalizations (Ioffe & Szegedy, 2015; Ba et al., 2016), which often significantly improve the performance of neural models and are frequently used by the models being compared to GNNs. Further, from our extensive experience of training GNNs across a wide range of datasets, tasks, and settings, both for research and for industrial applications, we observe that obtaining optimal performance with GNNs requires extensive hyperparameter tuning, which is often not performed in prior works. Thus, in this work, to fairly represent the capabilities of GNNs, we use GNN implementations with improved backbones from Platonov et al. (2023b) and Luo et al. (2024) and perform extensive per-dataset hyperparameter search for each of the considered GNNs. The GNN implementations from Platonov et al. (2023b) include skip-connections (He et al., 2016), 2-layer MLPs between graph message passing modules, and an option to use layer normalization (Ba et al., 2016), or batch normalization (Ioffe & Szegedy, 2015), or no normalization (this option is treated as a hyperparameter). The GNN implementations from Luo et al. (2024) include an option to use layer normalization (Ba et al., 2016), or batch normalization (Ioffe & Szegedy, 2015), or no normalization, and an option to use ego- and neighbor-embedding separation (which they refer to as a linear residual connection), which allows the model to concatenate the embedding of the ego-node to the result of aggregating embeddings of its neighbors during graph message passing, thus preserving more information about the ego-node in its combined representation (Hamilton et al., 2017; Zhu et al., 2020; Platonov et al., 2023b). All the considered GNNs also include dropout (Srivastava et al., 2014). For all the considered GNNs, we perform 100 trials of hyperparameter search with the popular TPE algorithm (Bergstra et al., 2011; Watanabe, 2023), see Appendix E and Appendix I for more details.

More specifically, we consider 4 GNN variants and have 2 versions of each of these 4 variants: one following the implementation from Platonov et al. (2023b) and one following the implementation from Luo et al. (2024). These 4 variants are 2 classic GNN models — GCN (Kipf & Welling, 2017) and GraphSAGE (Hamilton et al., 2017) — which both use fixed graph neighborhood aggregation (message passing) operations, and 2 GNN models with adaptive attention-based graph neighborhood aggregation operations — GAT (Veličković et al., 2018) with simple additive attention and Local Graph Transformer (LGT) (Shi et al., 2021; Platonov et al., 2023b) with scaled dot product attention (Vaswani et al., 2017) (note that this Graph Transformer variant only allows each node to attend to its neighbors — we specifically refer to it as a Local Graph Transformer following Platonov & Prokhorenkova (2026) to distinguish it from more common Global Graph Transformers that allow each node to attend to any other node unconstrained by the graph structure). The official implementation of GNNs from Platonov et al. (2023b) includes all 4 of these GNN variants, while the official implementation of GNNs from Luo

et al. (2024) includes 3 of them — it does not have an implementation of LGT, which we add by introducing an option to use the TransformerConv module from PyTorch Geometric (Fey & Lenssen, 2019) in the MPNNs class from the official codebase of Luo et al. (2024).

In our experiments in Section 3, we observe that the GNN implementations from Luo et al. (2024) typically perform better for the simpler GCN and GraphSAGE models, while the GNN implementations from Platonov et al. (2023b) typically perform better for the more complex attention-based GAT and LGT models. The GAT implementation from Platonov et al. (2023b) (referred to as Crit.-GAT in Tables 1, 2, 3) is the strongest of the GNNs in our experiments, achieving the fifth place among all the considered models according to both average rank and average normalized score aggregated metrics (and the third and the second places on the subset of only node classification datasets according to average rank and average normalized score aggregated metrics, respectively). Further, we also observe that the GNN implementations from Platonov et al. (2023b) tend to be more computationally efficient (often substantially) in both required time and memory usage than the GNN implementations from Luo et al. (2024). This is due to graph message passing in GNNs from Platonov et al. (2023b) being implemented using DGL (Wang et al., 2019b), while graph message passing in GNNs from Luo et al. (2024) is implemented using PyTorch Geometric (Fey & Lenssen, 2019): we find that DGL typically allows more efficient implementation of graph message passing than PyTorch Geometric (unfortunately, to the best of our knowledge, DGL is no longer being actively developed).

D. Dataset Selection

Reliable evaluation of GFMs for node property prediction requires careful dataset selection. Since the key characteristic of Foundation Models is that they can work well across diverse settings, the selected datasets should be diverse. In node property prediction, there are several potential dimensions of diversity, and it is desirable to cover all of them: diversity in data domains (e.g., social networks, web graphs, co-purchasing networks, road networks), diversity in tasks (e.g., fraud detection, view count prediction, topic prediction, CTR prediction), diversity in graph structure (size, sparsity, clustering, average shortest path length), diversity in graph structure and node labels relationships (homophily/assortativity — see Newman (2003); Platonov et al. (2023a); Mironov & Prokhorenkova (2024) for how to reliably measure this property), diversity in node features (features with different dimensions, distributions, and meanings).

Historically, standard benchmarks in graph node property prediction have been mostly limited to topic prediction in academic citation networks (perhaps due to the ease of obtaining such datasets), which obviously does not offer the necessary diversity — these datasets represent a single domain and task, have very similar graph structures, similarly high homophily, and typically come with outdated text-based features such as bag-of-words, TF-IDF, or bag-of-word-embeddings. Recently, standard Graph ML benchmarks have been extensively criticized (Bechler-Speicher et al., 2025; Bazhenov et al., 2025). Besides extensive reliance on citation networks and general low diversity, other notable points of critique include questionable practical relevance of many datasets, lack of complex and challenging tasks, limited representation of non-textual features common in real-world applications, unclear benefits of the graph structure for the considered tasks, potential bugs introduced in some datasets during data collection (we describe a number of previously reported and new problems with commonly used datasets in Appendix H). We believe these issues call for a change in standard datasets used for benchmarking Graph ML methods, and the rise of interest in developing GFMs provides a convenient opportunity to realize such a change in benchmarking together with a change in the types of models being benchmarked.

Due to increasing concerns about the quality of standard Graph ML datasets, two new benchmarks aiming for high quality, diversity, and practical relevance have been recently proposed: GraphLand (Bazhenov et al., 2025) and GraphBench (Stoll et al., 2025). While GraphBench mostly aims at widening the scope of Graph ML tasks and does not present many classic node property prediction tasks, GraphLand aims exactly at node property prediction and provides a diverse collection of datasets representing real-world industrial applications of this task. We believe GraphLand can be a much better testbed for evaluating GFMs for node property prediction than those commonly used in the current literature, and also encourage researchers and practitioners to release more datasets representing real-world applications of Graph ML. In our work, we use the 10 sub-million-node datasets from GraphLand for benchmarking GFMs for node property prediction (we do not consider the 4 million-node-scale datasets as none of the considered GFMs can work with datasets of such size, which highlights the importance of developing more scalable GFMs as a future direction of research).

E. Experimental Setup

For all datasets from the GraphLand benchmark, we use the official RL (random low) data splits, which are $10\%/10\%/80\%$ random stratified train/val/test splits. For each dataset and model, we report the mean and standard deviation of the obtained results computed over 10 runs. Graphs in some of the considered datasets are undirected and in some are directed; we preprocess the directed graphs by converting them to undirected, as done for the original GraphLand benchmark experiments (Bazhenov et al., 2025).

Hyperparameter selection is extremely important for achieving strong performance with GNNs, and very different hyperparameters can be optimal for different datasets. Thus, for each of the considered GNNs, we run 100 trials of hyperparameter optimization for each dataset using the popular Tree-structured Parzen Estimator (TPE) algorithm (Bergstra et al., 2011; Watanabe, 2023) provided in the Optuna library (Akiba et al., 2019). For each of the considered GNNs, we search over all hyperparameters available in the corresponding official implementation. The complete hyperparameter search spaces are provided in Appendix I. We train all GNNs with the AdamW optimizer (Kingma & Ba, 2015; Loshchilov & Hutter, 2019) for a maximum of 3000 steps using early stopping based on the validation set performance with a patience of 1000 steps.

GFMs can support the in-context learning (ICL) regime, the fine-tuning (FT) regime, or both. One of the benefits of the ICL regime is that it does not require hyperparameter optimization, as the pretrained model is used as is without weight optimization. In the FT regime, hyperparameter search for GFMs can be useful. However, on the one hand, many hyperparameter search trials often cannot be afforded, as GFMs are typically much more computationally expensive than classic GNNs, and, on the other hand, the hyperparameter search space is naturally much smaller, as the architecture of the model is fixed, and the only significant tunable hyperparameter is typically the learning rate. Thus, for GFMs in the FT regime, we only tune the learning rate, for which we greedily search over a set of 10 values. We follow the official implementations of the considered GFMs for other details of the fine-tuning strategy.

Some GFMs incorporate randomness in their inference, which allows for inference-time ensembling, i.e., running inference multiple times and aggregating predictions. Among the GFMs evaluated by us, this technique is supported and used by default by PFN-based GFMs: G2T-FM, TAG, and GraphPFN. We provide more details on inference-time ensembling and its influence on prediction quality of PFN-based GFMs in Appendix G.

We use the official implementations provided by the authors for all GNNs and GFMs. All models used in our experiments are implemented with PyTorch (Paszke et al., 2019), DGL (Wang et al., 2019b), and PyTorch Geometric (Fey & Lenssen, 2019). All experiments were run on NVIDIA Tesla A100 80GB GPUs. Note that, due to the specifics of our infrastructure, our time measurements reported in Table 2 may have noise up to 15%, but this does not affect our conclusions, as differences between time measurements for different model classes (GNNs and PFN-based GFMs) are significantly larger than that.

F. Memory Requirements of GNNs and PFN-Based GFMs

In Table 3, we provide the VRAM required for training and inference of GNNs and PFN-based GFMs (note that, when numbers above 80GB are reported, training on multiple GPUs with model parallelism was used). It can be seen that PFN-based GFMs are typically significantly more memory-intensive than GNNs, which is due to PFNs treating each feature as a separate token and thus creating a separate vector representation for each feature (while GNNs create a single vector representation for all features of a single node), and also due to PFNs often using larger models.

G. Inference-Time Ensembling of PFN-Based GFMs

Some models incorporate randomness in their inference, which allows for inference-time ensembling, i.e., running inference multiple times and aggregating predictions, which can lead to better prediction quality, although at a higher computational cost. This technique is commonly used in models following the PFNs paradigm, and is used by default by all the considered PFN-based GFMs: G2T-FM, TAG, and GraphPFN. We follow the official implementations of these models. For G2T-FM and GraphPFN, the official implementations use 10 inference runs for ensembling. In this section, we additionally investigate how this inference-time ensembling influences the predictive performance of these models. We only consider G2T-FM and GraphPFN models here, as TAG uses a more complicated ensembling strategy and, besides using multiple runs of the Tabular Foundation Model backbone, also includes in its ensemble a number of 'linear GNN' models (which are linear models fitted to predict node targets from node features augmented with various graph-neighborhood-aggregated features).

We report the results obtained on a subset of datasets by a single inference run and 10 inference runs of G2T-LimiX and GraphPFN in Table 4. We can see that inference-time ensembling almost always improves the predictive performance of the

Table 3. The memory (VRAM, in GB) required for: a single training run with the best hyperparameters (Tr), a single inference run with the best hyperparameters (Inf). Note that for GNNs, the models with the best hyperparameters can be widely different in size across GNN types and datasets. The *Crit.* and *Class.* prefixes refer to improved GNNs from Platonov et al. (2023b) and Luo et al. (2024), respectively (by the first words of the paper titles).

	tolokers-2		city-reviews		artnet-views		city-roads-M		city-roads-L		hm-prices	
	Tr	Inf	Tr	Inf	Tr	Inf	Tr	Inf	Tr	Inf	Tr	Inf
Crit.-GraphSAGE	1.74	0.44	26.23	3.10	3.16	0.41	4.16	0.65	10.91	1.93	5.55	1.14
Class.-GraphSAGE	0.34	0.32	9.34	5.33	1.32	0.75	0.43	0.21	3.11	1.44	28.98	28.29
Crit.-GCN	0.34	0.08	1.50	0.45	0.43	0.10	0.64	0.12	3.69	0.65	3.17	1.19
Class.-GCN	3.24	1.92	16.25	9.29	0.90	0.47	1.81	1.06	8.57	4.06	25.10	23.16
Crit.-GAT	0.80	0.10	18.00	1.77	1.31	0.34	3.16	0.58	6.98	0.74	34.31	2.89
Class.-GAT	4.94	1.28	41.97	12.34	7.68	2.19	2.38	1.02	17.45	3.45	79.32	39.52
Crit.-LGT	0.94	0.16	31.68	2.93	3.87	0.33	7.45	0.62	2.09	0.44	10.36	2.89
Class.-LGT	4.51	1.75	77.82	17.54	12.66	2.53	1.81	0.62	17.11	2.22	51.50	34.04
G2T-LimiX	16.57	4.54	53.05	30.97	25.55	25.32	21.04	20.81	73.17	68.19	64.20	63.97
TAG-TabPFNv2	N/A	10.15	N/A	29.17	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
TAG-LimiX	N/A	17.75	N/A	28.54	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
GraphPFN	8.12	1.63	146.66	31.95	73.60	14.97	41.78	8.30	145.49	36.34	70.81	20.52

Table 4. The difference in predictive performance of PFN-based GFMs when making a single inference run and when using inference-time ensembling of 10 inference runs.

	tolokers-2	city-reviews	artnet-exp	hm-prices	avazu-ctr	city-roads-M	twitch-views	artnet-views
Best GNN	58.94 ± 0.91	78.53 ± 0.08	49.43 ± 0.19	75.43 ± 0.22	32.43 ± 0.36	60.22 ± 0.47	78.02 ± 0.13	58.87 ± 0.32
G2T-LimiX (ICL, 1 run)	61.13 ± 0.21	77.29 ± 0.54	48.44 ± 0.23	75.41 ± 0.04	32.41 ± 0.12	64.53 ± 0.09	71.08 ± 0.07	60.95 ± 0.09
G2T-LimiX (ICL, 10 runs)	61.60 ± 0.18	78.98 ± 0.44	48.42 ± 0.78	76.14 ± 0.08	32.70 ± 0.14	65.16 ± 0.07	71.31 ± 0.06	61.58 ± 0.08
GraphPFN (ICL, 1 run)	60.97 ± 0.13	80.01 ± 0.03	51.46 ± 0.11	77.33 ± 0.06	31.43 ± 0.06	64.03 ± 0.11	72.75 ± 0.04	62.07 ± 0.05
GraphPFN (ICL, 10 runs)	61.29 ± 0.12	80.25 ± 0.05	51.79 ± 0.11	77.88 ± 0.09	31.63 ± 0.06	64.85 ± 0.13	73.20 ± 0.08	62.79 ± 0.08
G2T-LimiX (FT, 1 run)	60.82 ± 0.57	80.13 ± 0.13	49.79 ± 0.21	76.72 ± 0.20	34.03 ± 0.35	65.87 ± 0.11	74.31 ± 0.13	62.08 ± 0.12
G2T-LimiX (FT, 10 runs)	59.75 ± 1.14	80.65 ± 0.05	50.39 ± 0.19	77.37 ± 0.17	34.09 ± 0.37	66.29 ± 0.12	74.91 ± 0.06	63.24 ± 0.07
GraphPFN (FT, 1 run)	62.72 ± 0.24	80.34 ± 0.17	53.33 ± 0.23	80.16 ± 0.17	34.89 ± 0.16	66.02 ± 0.10	78.46 ± 0.09	64.37 ± 0.04
GraphPFN (FT, 10 runs)	62.80 ± 0.39	80.90 ± 0.03	53.49 ± 0.81	81.06 ± 0.24	35.07 ± 0.34	67.30 ± 0.21	79.00 ± 0.14	65.35 ± 0.06

model (there are exceptions in 2 cases for G2T-LimiX), and these improvements range from very minor to quite substantial. However, even without inference-time ensembling, the performance of these models remains very strong. In the ICL regime, G2T-LimiX and GraphPFN outperform the best considered GNN on 3 and 6 out of the 8 datasets, respectively, and in the FT regime, G2T-LimiX and GraphPFN outperform the best considered GNN on 7 and 8 out of the 8 datasets, respectively. We note that, when run sequentially (like in our case) rather than in parallel, such inference-time ensembling increases inference time tenfold (although it does not affect memory requirements). Overall, inference-time ensembling provides a prediction quality/time trade-off, and whether using it is worth it depends on the particular application.

H. Problems of Commonly Used Node Property Prediction Datasets

Historically, datasets for node property prediction in Graph ML have been dominated by academic citation networks, perhaps due to the ease of obtaining such datasets from open sources. Many works evaluate their methods predominantly or even exclusively on such datasets. However, this practice significantly limits the diversity of the evaluation setting: most citation network datasets are very similar to each other, and they also represent only a single data domain (which is also not particularly relevant to any real-world impactful applications). Further, the task most typically performed on these datasets is paper topic prediction; however, the topic of a paper can often be not uniquely defined, and it has been shown that a lot of nodes in these datasets appear to be mislabeled (Li et al., 2023).

There are also problems with other commonly used datasets for node property prediction. Some of them do not represent realistic applications (`coauthor-cs`, `coauthor-physics`, `airports-usa`, `airports-brazil`, `airports-europe`), in some, the provided graph does not appear to be beneficial for the considered task (`actor`), some come with no (or almost no) node features (`airports-usa`, `airports-brazil`, `airports-europe`, `deezer-hu`, `deezer-hr`, `deezer-ro`, `ogbn-proteins`), some are synthetic or semi-synthetic (`minesweeper`, `roman-empire`). Popular heterophilous node property prediction datasets have been criticized by Platonov et al. (2023b)

who point out that some of these datasets are extremely small and provide insufficient class representation (texas, cornell, wisconsin, with the texas dataset having a class that consists of a single node), while others appear to have a large number of duplicated nodes, which indicates a bug in the data collection process and leads to train-test data leakage (squirrel, chameleon). While Platonov et al. (2023b) created versions of these datasets with duplicated nodes removed, the purpose of these versions was to empirically demonstrate that the presence of duplicated nodes leads to biased evaluation results, rather than to make these filtered versions new standard benchmarks, since the exact reason duplicated nodes appear is not known and thus it is also not known whether the datasets with these nodes removed are meaningful and useful; yet some works use these filtered dataset versions for evaluation.

The problem with benchmarking in node property prediction is exacerbated by the fact that most datasets do not come with standardized data splits. As a consequence, different data splits are frequently used for the same datasets (often without being explicitly specified), thus complicating model comparison between different works, and some works borrow results reported elsewhere while using a different data split for their own experiments.

I. Hyperparameter Search Spaces for GNNs

In this section, we provide the complete hyperparameter search spaces for all the considered GNNs and GFMs. We use the following notation for distributions:

- Int[low, high, step] — integer hyperparameter sampled from a uniform distribution from low to high with step step.
- Float[low, high, step] — real hyperparameter sampled from a uniform distribution from low to high with step step.
- LogUniform[low, high] — real hyperparameter sampled from a log-uniform distribution from low to high.
- Cat[a, b, c, ...] — categorical hyperparameter sampled from a uniform distribution on unordered values a, b, c, ...

The complete hyperparameter search spaces are provided in Table 5 for GNN implementations from Platonov et al. (2023b) and in Table 6 for GNN implementations from Luo et al. (2024).

We note that identity_aggregation refers to concatenating the pre-aggregation node embedding to the output of the neighborhood aggregation operation, since it essentially uses the identity function to propagate the node representation further down the model (this technique popularized by Hamilton et al. (2017) is also sometimes referred to as ego- and neighbor-embedding separation (Zhu et al., 2020; Platonov et al., 2023b)). The other hyperparameter names are self-explanatory.

Table 5. The hyperparameter search distributions for Crit.-GNNs, i.e., GNN implementations from Platonov et al. (2023b).

Parameter	Distribution	Comment
learning_rate	LogUniform[0.00003, 0.01]	
weight_decay	LogUniform[0.0001, 1]	
dropout	Float[0, 0.5, 0.05]	
num_backbone_blocks	Int[1, 10, 1]	
hidden_dim	Int[96, 768, 32]	
normalization_type	Cat[None, LayerNorm, BatchNorm]	
$\log_2$(num_heads)	Int[1, 3, 1]	only for GAT and LGT models

Table 6. The hyperparameter search distributions for Class.-GNNs, i.e., GNN implementations from Luo et al. (2024).

Parameter	Distribution	Comment
learning_rate	LogUniform[0.00003, 0.01]	
weight_decay	LogUniform[0.0001, 1]	
dropout	Float[0, 0.5, 0.05]	
num_backbone_blocks	Int[1, 10, 1]	
hidden_dim	Int[96, 768, 32]	
identity_aggregation	Cat[False, True]	
normalization_type	Cat[None, LayerNorm, BatchNorm]	
$\log_2$(num_heads)	Int[1, 3, 1]	only for GAT and LGT models
num_input_layers	Int[0, 1, 1]	

J. Limitations

Our study reevaluates 9 GFMs for node property prediction from prior literature, covering several distinct model families. However, GFM development is an active research area, and new models continue to appear regularly. Thus, our evaluation is not exhaustive and, in particular, may omit recently proposed or concurrently developed methods.

Our experiments are conducted on datasets from the recently introduced GraphLand benchmark (Bazhenov et al., 2025). In Appendix D, we motivate this choice and discuss limitations of other existing node property prediction datasets. We believe GraphLand is currently the most suitable benchmark for node property prediction evaluation because it covers diverse and practical node property prediction tasks. Nevertheless, our conclusions may not generalize to domains outside GraphLand, such as proprietary datasets, highly specialized application areas, or datasets with different feature types or distributions.

Despite these limitations, to the best of our knowledge, our work provides the most comprehensive evaluations of GFMs for node property prediction to date.

K. Reproducibility

The code for reproducing our experiments with improved GNN implementations from Platonov et al. (2023b) and Luo et al. (2024) is available in this GitHub repository. For each of the considered GFMs, the official codebase for the respective model was used (with support for datasets from the GraphLand benchmark (Bazhenov et al., 2025) added by us where necessary).