PITFALLS OF EVALUATING GRAPH FOUNDATION MODELS FOR NODE PROPERTY PREDICTION

Anonymous authors
Paper under double-blind review

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 get 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 Machine Learning 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 node property prediction have been recently proposed, the field has not converged to a unified evaluation setting, and different works evaluate their models in widely different ways. We argue that the currently used evaluation settings often have significant limitations, preventing reliable assessment of GFM quality and comparison of GFMs with each other and to other types of models. In this work, we aim to highlight common pitfalls in the evaluation of GFMs for node property prediction, discuss potential solutions, and draw attention to open questions. We discuss such aspects as dataset and evaluation metric selection, aggregation of results across different datasets, proper tuning of baselines, and computational cost considerations, supporting each issue with illustrative examples from recent works in the field. We conclude with a fair reevaluation of 9 GFMs from the literature, showing that only the most recent GFMs based on the Prior-data Fitted Networks paradigm can compete with (and often outperform) properly tuned classic Graph Neural Networks in predictive performance, although at a much 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 at which GNNs have demonstrated particular success, 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; Chen et al., 2021; Derrow-Pinion et al., 2021; Borisyuk et al., 2024; De Nadai et al., 2024; Zheng 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 has 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, there have been a number of GFMs designed specifically for node property prediction across various datasets

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

(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. We argue that many commonly used evaluation practices have significant limitations that can prevent reliable assessment of GFM quality and comparison of GFMs with each other and to other types of models.

In this work, we aim to highlight the main recurring problems in evaluating GFMs for node property prediction, point out solutions where available, and draw attention to open problems. We discuss pitfalls of dataset selection, evaluation metric selection, results aggregation across datasets, baseline evaluation, and computational efficiency. For each issue, we provide supporting examples from recent works in the field. While such questions as what evaluation metrics to use and how to aggregate them across datasets may seem straightforward, we show that not taking proper care of them may result in misleading conclusions. We emphasize that our aim is not to criticize prior research, but to highlight common pitfalls and encourage more rigorous evaluation practices in the field.

To obtain a reliable assessment of the current state of the field, we conduct a large empirical study reevaluating 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 cannot outperform properly tuned GNNs. Only several most recent GFMs (Eremeev et al., 2025; 2026; Hayler et al., 2025) 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 being able to compete with and often outperform GNNs, although these models also come with significantly higher inference costs.

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 working with texts, images, video, and audio data. 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 as a general way to represent data from different domains. This has led to a natural result that, while many models being currently proposed in the literature get called GFMs, they are all designed for 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 models when calling them GFMs.

In this work, we focus on GFMs for node property prediction. 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 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 C. 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 D). 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 COMMON PITFALLS IN EVALUATING NODE PROPERTY PREDICTION GFMS

3.1 PITFALL 1: DATASET SELECTION

Problem 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-label relationships (homophily), 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 structure, 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 Machine Learning benchmarks have been strongly 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 believe these issues call for a change in standard datasets used for benchmarking Graph Machine Learning methods, and a 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. We provide more discussion of dataset problems, including some newly identified, in Appendix B.1.

Examples As of now, dataset selection for benchmarking GFMs for node property prediction does not always ensure the diversity and quality needed for reliable conclusions. Let us briefly review several setups commonly appearing in the literature. Some works evaluate GFMs exclusively on text-attributed graphs (Liu et al., 2024; Li et al., 2024; He et al., 2025), which does not allow investigating how well a model can work with features of other types. Xu et al. (2024); Yang et al. (2025) propose benchmarks for GFMs that consist predominantly of academic citation networks, which does not provide the necessary diversity. To improve diversity, Finkelshtein et al. (2025); Hayler et al. (2025); Choi et al. (2026) collect most of the available node property prediction datasets from the PyTorch Geometric library (Fey & Lenssen, 2019). However, the resulting collection still contains a disproportionally large share of citation networks and other text-attributed graphs, and also includes a number of datasets with known issues, such as datasets without node features (`airports-usa`, `airports-brazil`, `airports-europe`), datasets with some incorrect labels (`cora`, `citeseer`, `pubmed` — see (Li et al., 2023)), and datasets with duplicated nodes between train and test sets (`squirrel`, `chameleon` — see (Platonov et al., 2023)).

Potential solutions Due to increasing concerns about the quality of standard Graph Machine Learning 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 Machine Learning 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 better testbed for evaluating GFMs for node property prediction than benchmarks commonly used in the literature, and also encourage researchers to release more datasets representing real-world applications of Graph Machine Learning.

3.2 PITFALL 2: EVALUATION METRIC SELECTION

Problem For each dataset, an evaluation metric appropriate for the considered task should be chosen. The suitability of a metric should be first and foremost determined by its usefulness for real-world instantiations of this task. While standard metrics such as Accuracy for multiclass classification, ROC-AUC and Average Precision (also known as PR-AUC) for binary classification, and MSE, MAE, R^2 for regression can be appropriate for many tasks, some tasks can require more specialized metrics. A notable property of many node classification datasets in Graph Machine Learning (both classic and recently proposed ones) is class imbalance. Kumawat et al. (2025) recently argued against the use of Accuracy and ROC-AUC for standard Graph Machine Learning datasets with class imbalance and in favor of using Balanced Accuracy and Average Precision. We note that if an incorrect prediction for any object leads to equal negative outcomes (e.g., monetary losses), then Accuracy is a perfectly valid metric regardless of class imbalance. However, if errors for objects from some classes lead to larger negative outcomes (e.g., from the point of view of business objectives or societal fairness), then using a different metric is required. For most current multiclass node classification datasets, standard Accuracy seems an appropriate metric (e.g., product category classification in co-purchasing networks likely has roughly equal negative outcomes for misclassification of objects of different classes), although there is a wide range of alternative metrics for imbalanced multiclass classification with different properties (Gösgens et al., 2021). However, most current binary node classification datasets have strong class imbalance and obviously different outcomes for misclassification of objects of different classes (e.g., fraud detection, churn prediction, advertisement targeting, explicit content detection tasks). In such situations, Accuracy is not an appropriate metric, as it assigns equal costs to False Positive and False Negative predictions. ROC-AUC and Average Precision are metrics meant for imbalanced binary classification. They have different properties and, while there is some current discussion on their benefits compared to each other (Mihelich et al., 2024; McDermott et al., 2024), it is generally considered that Average Precision is more appropriate at least in cases of higher False Positive costs, which are prevalent in Graph Machine Learning datasets, since this metric assigns higher costs to incorrect high-confidence positive predictions. We emphasize that a metric appropriate for the considered task should be carefully chosen for each dataset. We provide a more detailed discussion of evaluation metrics and their selection for common node property prediction tasks in Appendix B.2.

Examples In the current GFM literature, Accuracy is frequently used to evaluate predictive performance on all node classification datasets including those with imbalanced binary classification tasks (Finkelshtein et al., 2025; Hayler et al., 2025; Choi et al., 2026). However, as discussed above, Accuracy is not a suitable measure for tasks with unequal costs of False Positive and False Negative predictions, e.g., such datasets as `tolokers`, `questions`, and `minesweeper`. The problem is best illustrated by the `questions` dataset which is a fraud detection dataset with extreme class imbalance: only slightly less than 3% of the nodes belong to the positive class. Due to the extreme class imbalance in this dataset, Accuracy does not distinguish different models: in Finkelshtein et al. (2025); Hayler et al. (2025); Choi et al. (2026), all the reported models achieve roughly the same Accuracy of slightly above 97% on this dataset (which is roughly the majority class proportion).

Potential solutions We advise to always carefully choose appropriate metrics for each dataset. Ideally, this should be done by dataset creators, so dataset users can then follow. We note that the recent GraphLand and GraphBench benchmarks come with predefined metrics for each dataset.

3.3 PITFALL 3: RESULTS AGGREGATION ACROSS DATASETS

Problem Since it is important to evaluate GFMs for node property prediction on a wide and diverse range of datasets, and it generally cannot be expected that a single model achieves the best results on all of these datasets, it is often desirable to aggregate evaluation results across datasets to compare these aggregated results. Several approaches for such aggregation are commonly used in the literature, none of which we believe are perfect.

The simplest aggregation is the average of the obtained metrics across all datasets, which is used, for instance, in Finkelshtein et al. (2025); Hayler et al. (2025). This approach can in principle be used even with different metrics as long as they have the same range, e.g., if Accuracy, Average Precision, and R^2 are used for different datasets. However, raw metric values across different datasets are typically not commensurable (even if the same metric is used for all datasets), and thus averaging them

does not produce an informative value. Also, this aggregation gives more weight to improvements on those datasets where the ranges of obtained metrics are larger. In particular, a large improvement on one dataset may outweigh the lack of improvements or performance loss on the remaining datasets. Another common approach is to compute average model ranks across all datasets, as done, for instance, in Eremeev et al. (2025; 2026). The problem with ranks is that they do not account for the magnitudes of performance differences and thus may become noisy when multiple models achieve very close results. Yet another approach is to average normalized metrics, i.e., metrics linearly transformed so that the worst model achieves a value of 0 and the best model achieves a value of 1 (these transformed metrics are sometimes called normalized scores). This aggregation was used, for instance, in Hollmann et al. (2025). A potential problem with this approach is that if all models achieve almost the same results on some dataset, insignificant differences between their results will still be given a large weight. Also, the obtained ranking significantly depends on the set of compared models since even a single model can substantially affect the range of raw metric values and, consequently, the normalized scores for all the models. Another approach is to use the Elo rating system (Elo, 1967) or similar systems popularized in machine learning evaluation by Chatbot Arena (Chiang et al., 2024) and recently adapted for evaluating Tabular Foundation Models (which face the same challenge as GFMs when it comes to aggregation of results across multiple datasets) on TabArena (Erickson et al., 2025). However, similarly to average ranks, such rating systems do not account for the magnitudes of performance differences an rely exclusively on pairwise comparisons, which is suitable for user preference comparisons of LLMs on Chatbot Arena, but ignores a lot of potentially useful information when metric values are available such as in node property prediction. Finally, the problem of comparison of predictive models across multiple datasets has been considered from the statistical testing point of view (Demšar, 2006), recommending the Friedman test (Friedman, 1937; 1940; Iman & Davenport, 1980) and its corresponding Nemenyi post-hoc test (Nemenyi, 1963), which is the most principled of the considered approaches. However, a potential problem with it is that these tests are generally conservative and can have little statistical power. Overall, it is not clear what are the best approaches for aggregating model results across datasets.

Examples To demonstrate how using different results aggregation can lead to drastically different conclusions, we provide an illustrative synthetic example in Table 1. Here, three algorithms (A, B, and C) are evaluated on three datasets. Depending on the aggregation used (average metric, average rank, or average normalized score) each of the three algorithms can be considered the best, which clearly demonstrates that choosing an aggregation approach is critical. While this example is artificially created to illustrate the problem, similar issues frequently occur in practice when researchers compare a number of GNN and GFM models and aggregate their results across several datasets. We present one such example based on results reported in a research paper in Appendix B.3.

Table 1: Aggregation example

Algorithm	A	B	C
Dataset 1	90	50	48
Dataset 2	20	50	48
Dataset 3	40	41	42
Avg metric	**50**	47	46
Avg rank	2.33	**1.67**	2.00
Avg norm score	0.33	0.52	**0.64**

Potential solutions We believe finding the best approach(es) to aggregating results across multiple datasets is currently an open problem. It is also unlikely that any one aggregated value would be able to capture all the nuances of comparison across multiple datasets. Thus, for more holistic results assessment, our recommendation is to present the results of multiple aggregation approaches side by side and take into account the limitations of each aggregation when analyzing the results. Further, manual examination of results on individual datasets before aggregation is beneficial, since it can unveil how the specifics of model performance distributions on particular datasets (e.g., presence of outliers or, on the contrary, all results being extremely close to each other) can affect the aggregated metrics.

3.4 Pitfall 4: Baseline evaluation

Problem Obtaining representative results for baselines is paramount for evaluating a new method. However, it is a well-known issue in machine learning research that often much less effort is dedicated to tuning baselines than to optimizing a newly proposed method (Lipton & Steinhardt, 2018). While many other fields of machine learning have recently adopted the use of standardized datasets on which a new (typically very large) model is evaluated once and then its obtained results are reported on a public leaderboard and can be referenced by further works, this is not the case for Graph Machine Learning, where relatively small models and datasets prevail, and many new works reeval-

uate these models, with the used experimental settings differing dramatically across works. As a result, reported metrics for the same model on the same dataset can differ drastically in different works.

It is critically important to evaluate baselines fairly. In modern Graph Machine Learning, baselines are typically message-passing GNNs. Two issues frequently prevent a fair evaluation of GNNs. First, GNNs (like most relatively small models trained from scratch) are very sensitive to hyperparameter selection, and proper hyperparameter tuning is often not performed. Second, most GNN implementations do not use standard architectural elements like skip-connections (He et al., 2016) and normalizations (Ioffe & Szegedy, 2015; Ba et al., 2016), which can significantly improve the performance of GNNs (at the same time, GFMs, which are most often based on the Transformer architecture (Vaswani et al., 2017), typically use these elements to their advantage). A number of works have performed more fair reevaluations of classic GNNs (Shchur et al., 2018; Errica et al., 2020; Platonov et al., 2023; Tönshoff et al., 2023; Luo et al., 2024; 2025; Bazhenov et al., 2025) in different settings, often focusing on hyperparameter search and/or simple architectural improvements, and shown that these models can achieve much stronger results than what is commonly reported in the literature. However, their results and insights have not yet been widely adopted when evaluating baselines for comparison with GFMs, which may lead to these comparisons not fully reflecting the capabilities of baseline models.

Examples Widely different results are often reported for the same model and the same dataset in different works. For illustration purposes, we will consider the `ogbn-arxiv` dataset from the OGB benchmark (Hu et al., 2020), which is particularly convenient since it has a single official data split that is typically used. The official OGB leaderboard[1] reports the Accuracy values of 55.50 for graph-agnostic MLP and 73.60 for a simple GNN — GCN (Kipf & Welling, 2017). At the same time, the results on this dataset reported in the literature for another standard GNN — GAT (Veličković et al., 2018) — vary from as high as 73.56 Huang et al. (2021) to as low as 57.93 (Finkelshtein et al., 2025; Hayler et al., 2025). This is one example illustrating a frequently occurring issue of widely different baseline evaluation practices across research papers.

Potential solutions Significant efforts should always be given to obtaining representative results with baselines. When evaluating GNNs, extensive hyperparameter tuning should be performed, and standard deep learning building blocks like skip-connections and normalizations should be incorporated into the architecture. (Luo et al., 2024) is a recent work providing a relatively simple but strong evaluation protocol for node classification with GNNs including both hyperparameter search and architectural improvements. This work shows that classic GNNs are still often the best models.

3.5 Pitfall 5: Computational cost considerations

Problem Besides comparing the performance of a newly proposed model to baselines, it is also highly desirable to compare their computational costs (both time and memory required). When developing GFMs for node property prediction, this issue is particularly important, as GFMs are often significantly more computationally expensive than baseline GNNs. Nevertheless, computational cost comparisons are typically not provided in the current GFM literature.

However, comparing computational costs of GFMs and baseline GNNs is not straightforward, as these models operate in very different regimes. GNNs follow the classic paradigm of requiring relatively long training (hundreds to thousands model forward and backward passes) to optimize model parameters and then performing relatively fast inference (a single model forward pass) to make predictions. The development and usage process of GFMs is notably different. First, these models require a (typically very 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. However, GFMs differ in how they can be applied to these downstream tasks. Some GFMs support in-context learning (ICL) regime, which means that they can make predictions in a single forward pass for any new datasets. Other GFMs need to be fine-tuned for each new dataset before they can make predictions, making their application to a new dataset similar to that of GNNs. And even models that support ICL can often obtain better

[1] `https://ogb.stanford.edu/docs/leader_nodeprop/#ogbn-arxiv`

results if fine-tuned for each specific dataset. Further, the time of a single GNN training or GFM fine-tuning run is not particularly illustrative, as these models require hyperparameter tuning. As discussed in the previous section, hyperparameter tuning is critical for GNNs and obtaining the best results with them can sometimes require performing more than a hundred training runs. GFMs often have less hyperparameters, as the model architecture is completely fixed after pretraining, however, each individual training run for them is often more expensive. Providing both the time of a single training or fine-tuning run and the time spent for the entire hyperparameter tuning process is most informative.

We also note that the computational costs of GNN training or GFM fine-tuning and those of their inference have very different weight in real-world applications. Arguably most applications of node property prediction are in large-scale industrial settings such as social networks, financial transaction networks, co-purchasing networks, transportation networks, recommendation systems, etc. In such settings, relatively large resources can typically be allocated to the model development (which includes the multiple training runs required for hyperparameter tuning), however, inference then needs to be performed frequently on large graphs and provide results in (nearly) real time, which leads to strict limits for inference computational costs. Thus, it is the inference efficiency rather than training or fine-tuning efficiency that is typically the bottleneck in large-scale industrial applications which are some of the most notable usages of node property prediction.

Finally, let us note that when working with large-scale graphs, GNNs can be easily adapted to perform training and inference using only relatively small randomly sampled subgraphs (Hamilton et al., 2017; Chen et al., 2018; Chiang et al., 2019; Balin & Çatalyürek, 2023). This is, however, not the case for some of the GFMs proposed in the literature which utilize global attention that requires access to the entire graph.

Examples To the best of our knowledge, no prior work proposing GFMs for node property prediction has provided a comprehensive evaluation of computational costs of their models compared to baselines. Overall, GFMs for node property prediction being propsoed in the literature are based on widely different principles and use different architectures and model sizes, and thus strongly differ in computational costs: from relatively small and fast models that are comparable to standard GNNs (Finkelshtein et al., 2025), to more computationally expensive PFN-based models (Eremeev et al., 2025; 2026; Choi et al., 2025; 2026; Hayler et al., 2025), to extremely expensive models that utilize LLMs (Liu et al., 2024; He et al., 2025). However, recent benchmarking in (Bazhenov et al., 2025) demonstrates that even smaller GFMs have troubles scaling to million-node-scale graphs which GNNs can handle even without subgraph sampling.

Potential solutions We encourage creators of GFMs to transparently report inference and fine-tuning time and required memory for graphs of different sizes. While different real-world applications of node property prediction work with graphs of vastly different sizes and have vastly different computational cost requirements, providing computational cost information allows making conclusions about which types of applications a particular GFM is suitable for.

4 Reevaluating GFMs for node property prediction

Due to the issues discussed in the previous sections, we believe the currently available evaluations of GFMs for node property prediction do not allow making reliable conclusions about model performance. Thus, we set out to reevaluate a large amount of GFMs in a more reliable setting. As our main benchmark, we adopt 10 datasets from the GraphLand suite (Bazhenov et al., 2025) (we use the 10 sub-million-nodes datasets, since all the considered GFMs cannot scale to the remaining 4 larger datasets), using their official `RL` (random low) data splits. We use the officially recommended metrics for these datasets, which are Accuracy for multiclass classification, Average Precision for binary classification, and R^2 for regression. To aggregate results across all datasets, we use both average ranks and average normalized scores. As our baselines, we use architecturally improved versions of classic GNNs (GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), and GAT (Veličković et al., 2018)) from Luo et al. (2024). For them, we perform 100 trials of hyperparameter search using the Tree-structured Parzen Estimator (TPE) algorithm (Bergstra et al., 2011; Watanabe, 2023) implemented in the Optuna library (Akiba et al., 2019). More details on our experimental setup are provided in Appendix E.

Table 2: Model performance on 10 GraphLand datasets (RL data split). Average rank (lower is better) and average normalized score (higher is better) are computed either only across classification datasets (*rank (cls)* and *score (cls)* columns) or across all datasets (*rank (all)* and *score (all)* columns). The three best results are highlighted with color: 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) ↑
Improved-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
Improved-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
Improved-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
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

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. 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 fine-tuning (FT) regime. OpenGraph, AnyGraph, TS-GNN, and all PFN-based GFMs support in-context learning (ICL) regime, and the official implementations of G2T-FM and GraphPFN also support fine-tuning regime. We evaluate each model using all the supported regimes. PFN-based models support inference-time ensembling, which improves results but further increases inference time. The official implementations of G2T-FM, TAG, and GraphPFN use an ensemble of multiple inference runs by default, and we follow this setting (we analyze the influence of inference-time ensembling on the predictive performance and inference speed of these models in Appendix G). Note that some of the considered models do not support node regression — we evaluate these models only on node classification datasets.

We provide the experimental results in Table 2. Based on them, we make the following observations:

- The results of GFMs clearly separate them 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, sometimes 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 of the 10 datasets (and even on these 2 datasets, the first place belongs to GraphPFN).
- Comparing PFN-based GFMs and their different regimes 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. 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 is also based on LimiX but 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-LimiX significantly increases.
- GraphPFN in the fine-tuning regime is the best model overall, achieving the best results on all the considered datasets (often outperforming all other models by a substantial margin).
- While there are some discrepancies in model rankings with average rank and average normalized score, in our experiments they agree in most cases, and the main observations described in the previous bullet points hold regardless of which of these result aggregation approaches is used.

Table 3: 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). 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
Improved-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
Improved-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
Improved-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
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

In Appendix A, we additionally provide results on 11 classic node classification datasets. While we recommend relying on these experiments less due to issues with many classic datasets discussed in Section 3.1, the obtained results on these datasets agree with our main conclusions: non-PFN-based GFMs underperform classic GNNs, while PFN-based GFMs on average outperform them; results of PFN-based GFMs are improved by fine-tuning; fine-tuned GraphPFN is the best model on average; and the two considered result aggregation metrics provide quite similar model rankings.

Note that our results are different from those reported in some prior works, where for each of the GFMs considered in our experiments it was reported that it outperforms classic GNNs, while we find that this is only the case for PFN-based GFMs. We believe this discrepancy stems from differences in baseline tuning and evaluation methodology, which further illustrates the pitfalls discussed in the previous sections.

Now, let us discuss computational efficiency. For a representative subset of datasets, we provide time required for the entire hyperparameter tuning, a single training run, and a single inference run of GNNs and PFN-based GFMs in Table 3 (we provide a similar table for memory usage in Appendix F). All experiments were run on NVIDIA Tesla A100 80GB GPUs.

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, while significant resources can be allocated to model development such as hyperparameter tuning. And in inference time, GNNs significantly outcompete PFN-based GFMs: GNNs generally make predictions in far less than 1 second on the considered datasets, while even the most efficient PFN-based GFMs — G2T-LimiX and GraphPFN — require from a few seconds to a few minutes (TAG models are even slower, likely due to using a more complex ensembling strategy). Note that the time complexity of PFN-based GFMs is quadratic in the number of input node features (since they use each input feature as a separate token and apply Transformer self-attention to these tokens), and G2T-FM and TAG models generate a large number of additional graph-based node features, which explains while they are significantly slower than GraphPFN, even though the latter has a larger and more complex architecture due to using additional graph message passing layers. The memory requirements of GNN inference are also typically significantly below those of PFN-based GFMs (see Appendix F).

However, when model development time is critical, PFNs can be preferable to GNNs, as they can make predictions in the ICL regime without any dataset-specific fine-tuning (although their predictive quality is typically not as good in the ICL regime as in the FT regime, as discussed above), and even their fine-tuning can take less time than developing GNNs since PFNs require less hyperparameter search (due to the architecture being entirely fixed) and each individual training run of hyperparameter search often converges faster for them than for GNNs (likely due to an effective pretrained initialization).

5 Conclusion

We discussed common limitations in evaluating GFMs for node property prediction and provided a rigorous reevaluation of a wide range of models, showing that only recent PFN-based GFMs can compete with well-tuned classic GNNs and often outperform them, although at a much higher inference cost. We hope this work encourages the adoption of more standardized, rigorous, and reliable evaluation practices as the field continues to develop.

AI USE STATEMENT

In this work, we used generative AI tools for some parts of implementing methods. All AI-generated code was verified and tested for correctness by the authors. The authors take full responsibility for the final versions of the implemented methods. We did not use AI tools for the following tasks with required disclosure: proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, cleaning and reformatting datasets, supporting qualitative and thematic data analysis, interpreting results. The following tasks with required disclosure are not applicable to our work: generating synthetic datasets, helping developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, assisting with translation.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 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.

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.

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

Dominique Beaini, Shenyang Huang, Joao Alex Cunha, Zhiyi Li, Gabriela Moisescu-Pareja, Oleksandr Dymov, Samuel Maddrell-Mander, Callum McLean, Frederik Wenkel, Luis Müller, et al. Towards foundational models for molecular learning on large-scale multi-task datasets. In *International Conference on Learning Representations*, 2024.

Maya Bechler-Speicher, Ben Finkelshtein, Fabrizio Frasca, Luis Müller, Jan Tönshoff, Antoine Siraudin, Viktor Zaverkin, Michael M. Bronstein, Mathias Niepert, Bryan Perozzi, Mikhail Galkin, and Christopher Morris. Position: Graph learning will lose relevance due to poor benchmarks. In *International Conference on Machine Learning*, 2025.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems*, 24, 2011.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Fedor Borisyuk, Shihai He, Yunbo Ouyang, Morteza Ramezani, Peng Du, Xiaochen Hou, Chengming Jiang, Nitin Pasumarthy, Priya Bannur, Birjodh Tiwana, Ping Liu, Siddharth Dangi, Daqi Sun, Zhoutao Pei, Xiao Shi, Sirou Zhu, Qianqi Shen, Kuang-Hsuan Lee, David Stein, Baolei Li, Haichao Wei, Amol Ghoting, and Souvik Ghosh. LiGNN: Graph neural networks at LinkedIn. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4793–4803, 2024.

Huiyuan Chen, Lan Wang, Yusan Lin, Chin-Chia Michael Yeh, Fei Wang, and Hao Yang. 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.

Jianfei Chen, Jun Zhu, and Le Song. Stochastic training of graph convolutional networks with variance reduction. In *International Conference on Machine Learning*, 2018.

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.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E Gonzalez, et al. Chatbot arena: An open platform for evaluating LLMs by human preference. *arXiv preprint arXiv:2403.04132*, 2024.

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

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

Marco De Nadai, Francesco Fabbri, Paul Gigioli, Alice Wang, Ang Li, Fabrizio Silvestri, Laura Kim, Shawn Lin, Vladan Radosavljevic, Sandeep Ghael, David Nyhan, Hugues Bouchard, Mounia Lalmas, and Andreas Damianou. Personalized audiobook recommendations at Spotify through graph neural networks. In *Companion Proceedings of the ACM Web Conference 2024*, pp. 403–412, 2024.

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

Austin Derrow-Pinion, Jennifer She, David Wong, Oliver Lange, Todd Hester, Luis Perez, Marc Nunkesser, Seongjae Lee, Xueying Guo, Brett Wiltshire, Peter W. Battaglia, Vishal Gupta, Ang Li, Zhongwen Xu, Alvaro Sanchez-Gonzalez, Yujia Li, and Petar Veličković. 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.

Kaiwen Dong, Haitao Mao, Zhichun Guo, and Nitesh V Chawla. Universal link predictor by in-context learning on graphs. *arXiv preprint arXiv:2402.07738*, 2024.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in Neural Information Processing Systems*, 28, 2015.

Arpad E. Elo. The proposed USCF rating system, its development, theory, and applications. *Chess life*, 22(8):242–247, 1967.

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.

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

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.

Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *International Conference on Learning Representations*, 2020.

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

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

Fabrizio Frasca, Fabian Jogl, Moshe Eliasof, Matan Ostrovsky, Carola-Bibiane Schönlieb, Thomas Gärtner, and Haggai Maron. Towards foundation models on graphs: An analysis on cross-dataset transfer of pretrained GNNs. *arXiv preprint arXiv:2412.17609*, 2024.

Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. Towards foundation models for knowledge graph reasoning. In *International Conference on Learning Representations*, 2024.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pp. 1263–1272, 2017.

Martijn Gösgens, Anton Zhiyanov, Aleksey Tikhonov, and Liudmila Prokhorenkova. Good classification measures and how to find them. *Advances in Neural Information Processing Systems*, 34: 17136–17147, 2021.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, Dominik Safaric, Jake Robertson, Benjamin Jäger, Simone Alessi, Adrian Hayler, Vladyslav Moroshan, Lennart Purucker, Philipp Singer, Alan Arazi, Julien Siems, Jan Hendrik Metzen, Georg Grab, Nick Erickson, Siyuan Guo, Eliott Kalfon, Simon Bing, David Salinas, Clara Cornu, Lilly Charlotte Wehrhahn, Diana Kriuchkova, Kursat Kaya, Lydia Sidhoum, Marie Salmon, Jerry Chen, Madelon Hulsebos, Yann LeCun, Samuel Müller, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-3: Technical report. *arXiv preprint arXiv:2605.13986*, 2026.

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

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.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in Neural Information Processing Systems*, 33:22118–22133, 2020.

Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R Benson. Combining label propagation and simple models out-performs graph neural networks. In *International Conference on Learning Representations*, 2021.

Ronald L Iman and James M Davenport. Approximations of the critical region of the friedman statistic. *Communications in Statistics-Theory and Methods*, 9(6):571–595, 1980.

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

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

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

Rupam Kumawat, Manasvi Aggarwal, Sahil Manchanda, Suhas Powar, and Yatin Katyal. Unpacking evaluation pitfalls on standard GNN benchmarks, 2025. URL `https://openreview.net/forum?id=H0BZJxOmE4`.

Divyansha Lachi, Mehdi Azabou, Vinam Arora, and Eva L Dyer. GraphFM: A generalist graph transformer that learns transferable representations across diverse domains. *Transactions on Machine Learning Research (TMLR)*, 2025.

Yuhan Li, Peisong Wang, Zhixun Li, Jeffrey Xu Yu, and Jia Li. 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.

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

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

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. 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*, 2025.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000.

Matthew B McDermott, Haoran Zhang, Lasse H Hansen, Giovanni Angelotti, and Jack Gallifant. A closer look at AUROC and AUPRC under class imbalance. *Advances in Neural Information Processing Systems*, 37:44102–44163, 2024.

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

Martin Mihelich, François Castagnos, and Charles Dognin. Interplay of ROC and precision-recall AUCs: Theoretical limits and practical implications in binary classification. In *International Conference on Machine Learning*, 2024.

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

Galileo Namata, Ben London, Lise Getoor, and Bert Huang. Query-driven active surveying for collective classification. In *Proceedings of the Workshop on Mining and Learning with Graphs (MLG-2012)*, 2012.

Peter Bjorn Nemenyi. *Distribution-free multiple comparisons.* Princeton University, 1963.

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

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*, 2023.

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. In *International Conference on Machine Learning*, 2026.

Benedek Rozemberczki and Rik Sarkar. Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 1325–1334, 2020.

Benedek Rozemberczki, Ryan Davies, Rik Sarkar, and Charles Sutton. Gemsec: Graph embedding with self clustering. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2019*, pp. 65–72. ACM, 2019.

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

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–93, 2008.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. In *Relational Representation Learning Workshop at NeurIPS*, 2018.

Timo Stoll, Chendi Qian, Ben Finkelshtein, Ali Parviz, Darius Weber, Fabrizio Frasca, Hadar Shavit, Antoine Siraudin, Arman Mielke, Marie Anastacio, Erik Müller, Maya Bechler-Speicher, Michael Bronstein, Mikhail Galkin, Holger Hoos, Mathias Niepert, Bryan Perozzi, Jan Tönshoff, and Christopher Morris. GraphBench: Next-generation graph learning benchmarking. *arXiv preprint arXiv:2512.04475*, 2025.

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

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 Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 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*, 2019.

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.

Xuhong Wang, Ding Wang, Liang Chen, Fei-Yue Wang, and Yilun Lin. Building transportation foundation model via generative graph transformer. In *2023 IEEE 26th International conference on intelligent transportation systems (ITSC)*, pp. 6042–6047. IEEE, 2023.

Yuanrui Wang, Xingxuan Zhang, Han Yu, Mingchao Hao, Gang Ren, Hao Yuan, Li Mao, Yunjia Zhang, Chun Yuan, and Peng Cui. LimiX-2M: Mitigating low-rank collapse and attention bottlenecks in tabular foundation models. In *International Conference on Machine Learning*, 2026.

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

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. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

Yuhao Xu, Xinqi Liu, Keyu Duan, Yi Fang, Yu-Neng Chuang, Daochen Zha, and Qiaoyu Tan. GraphFM: A comprehensive benchmark for graph foundation model. *arXiv preprint arXiv:2406.08310*, 2024.

Jinyu Yang, Liangwei Yang, Zeyuan Guo, Jiayi Gao, Jing Wu, Tianhao Chai, Hai Huang, Cheng Yang, and Chuan Shi. Benchmarking graph foundation models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 5866–5875, 2025.

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

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 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.

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 Web Conference 2025*, pp. 1142–1153, 2025.

Shihao Zhang, Linlin Yang, Michael Bi Mi, Xiaoxu Zheng, and Angela Yao. Improving deep regression with ordinal entropy. In *International Conference on Learning Representations*, 2023.

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*, 2025.

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.

Tong Zhao, Yozen Liu, Matthew Kolodner, Kyle Montemayor, Elham Ghazizadeh, Ankit Batra, Zihao Fan, Xiaobin Gao, Xuan Guo, Jiwen Ren, Serim Park, Peicheng Yu, Jun Yu, Shubham Vij, and Neil Shah. 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.

Da Zheng, Xiang Song, Qi Zhu, Jian Zhang, Theodore Vasiloudis, Runjie Ma, Houyu Zhang, Zichen Wang, Soji Adeshina, Israt Nisa, et al. GraphStorm: all-in-one graph machine learning framework for industry applications. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 6356–6367, 2024.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020.

A ADDITIONAL RESULTS ON CLASSIC DATASETS

In addition to the GraphLand datasets, we also conducted experiments on 11 popular classic node classification datasets. Note that we generally trust results obtained on these datasets less than those obtained on the GraphLand datasets, due to many of the classic datasets being less realistic and often having other issues as discussed in Section 3.1 and Appendix B.1. Nevertheless, we provide these results to make our empirical study more comprehensive and demonstrate that our conclusions also generalize to classic datasets. Still, we encourage the Graph Machine Learning community to continue creating, releasing, and adopting more realistic and reliable open datasets.

The considered classic node classification datasets are: citation networks `cora` and `pubmed` from McCallum et al. (2000); Sen et al. (2008); Namata et al. (2012); Yang et al. (2016), a Wikipedia articles hyperlink network `wiki-cs` from Mernyei & Cangea (2020), co-authorship networks `coauthor-cs` and `coauthor-physics` from Shchur et al. (2018), co-purchasing networks `amazon-computers`, `amazon-photo` from Shchur et al. (2018), social networks `lastfm-asia` from Rozemberczki & Sarkar (2020) and `facebook` from Rozemberczki et al. (2019), a co-purchasing network `amazon-ratings` from Platonov et al. (2023), a question-answering website network `questions` from Platonov et al. (2023). For all of these datasets, we use random class-stratified data splits with 10%/10%/80% train/val/test proportions.

The results on the 11 considered classic node classification datasets are provided in Table 4. The main observations that can be made from them agree with those made from the results on the GraphLand datasets in Section 4:

- The GFMs are clearly separated into two groups: non-PFN-based GFMs almost always underperform well-tuned classic GNNs on the considered classic datasets (the only exception is TS-GNN on the `amazon-ratings` datasets, where it is better than classic GNNs), sometimes by a substantial margin. In contrast, PFN-based GFMs typically outperform GNNs, with one of PFN-based GFMs achieving the best result on 10 of the 11 datasets.
- When comparing PFN-based GFMs and their different regimes to each other, we see that fine-tuning almost always improves the performance of these models (with the only exception being GraphPFN on the `pubmed` dataset, where results of the model in the ICL and FT regimes match). But even in the ICL regime, they always outperform all other GFMs and are competitive with well-tuned GNNs. 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. However, they are in most cases outperformed by GraphPFN (which is also based on LimiX, but directly incorporates graph message passing in the PFN architecture and thus natively supports graph-structured data) in the ICL regime: on 7 out of the 11 dataset for G2T-LimiX and on 6 out of the 11 datasets for TAG-LimiX. Further, in the fine-tuning regime, the lead of GraphPFN over G2T-LimiX significantly increases.
- GraphPFN in the fine-tuning regime is the best model overall, achieving the best results on 7 of the 11 considered datasets and getting in the top-3 models on 3 more datasets. It obtains an impressive average normalized score of 99.76, showing that even when it is not the best model, it is only slightly behind the best competitor.
- Average rank and average normalized score provide quite similar model rankings in our experiments, and the main observations hold regardless of which of these result aggregation approaches is used.

B ADDITIONAL DISCUSSION OF COMMON PITFALLS

B.1 PITFALL 1: DATASET SELECTION

Historically, datasets for node property prediction in Graph Machine Learning 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

Table 4: Model performance on 11 classic node classification datasets. Accuracy is computed for multiclass classification datasets and average precision is computed for binary classification datasets. Average rank (lower is better) and average normalized score (higher is better) are provided in *rank (all)* and *score (all)* columns. The three best results are highlighted with color: first, second, third.

	mult. class.										bin. class.		
	cora	pubmed	wiki-cs	coauth-cs	coauth-ph	amzn-comp	amzn-photo	lastfm-a	facebook	amzn-ratings	questions	rank (all) ↓	score (all) ↑
Improved-GraphSAGE	82.05 ± 0.50	85.72 ± 0.14	82.29 ± 0.08	93.81 ± 0.03	96.05 ± 0.02	90.03 ± 0.14	93.91 ± 0.18	83.84 ± 0.26	92.50 ± 0.11	42.08 ± 0.43	16.81 ± 0.43	6.45	91.04
Improved-GCN	82.16 ± 0.55	86.63 ± 0.19	81.16 ± 0.48	94.64 ± 0.12	96.33 ± 0.04	91.03 ± 0.28	93.59 ± 0.17	83.91 ± 0.53	93.38 ± 0.31	42.02 ± 0.45	17.55 ± 0.53	4.91	92.12
Improved-GAT	79.92 ± 0.82	85.70 ± 0.14	82.60 ± 0.20	93.57 ± 0.13	95.85 ± 0.06	90.37 ± 0.19	93.42 ± 0.39	84.80 ± 0.50	93.20 ± 0.11	42.53 ± 0.33	15.01 ± 0.89	6.73	90.14
OpenGraph	69.11 ± 1.39	72.90 ± 1.20	75.47 ± 1.20	82.72 ± 0.34	88.74 ± 0.56	80.01 ± 0.54	88.25 ± 0.22	76.22 ± 2.52	78.40 ± 2.48	29.85 ± 1.75	3.64 ± 0.89	11.73	56.44
AnyGraph	75.10 ± 6.00	67.57 ± 5.19	65.71 ± 2.31	75.46 ± 3.63	60.62 ± 13.09	76.92 ± 1.66	63.51 ± 7.71	70.64 ± 3.23	64.90 ± 7.16	33.28 ± 3.13	4.43 ± 0.41	13.09	33.33
TS-GNN	48.06 ± 4.49	64.41 ± 5.11	46.25 ± 9.77	51.08 ± 5.99	78.25 ± 4.85	53.72 ± 7.92	66.16 ± 9.50	49.00 ± 4.95	77.88 ± 2.73	42.99 ± 0.13	5.00 ± 0.48	13.73	17.93
GCOPE	79.96 ± 0.73	81.62 ± 0.36	62.29 ± 0.69	85.47 ± 0.31	92.17 ± 0.16	75.08 ± 0.31	80.06 ± 0.53	81.04 ± 0.19	84.97 ± 0.19	40.49 ± 0.46	7.67 ± 0.27	11.09	65.94
SAMGPT	68.90 ± 1.31	71.93 ± 1.45	63.82 ± 2.35	85.90 ± 0.89	90.77 ± 0.50	77.51 ± 0.83	87.26 ± 1.05	69.42 ± 0.49	80.61 ± 0.93	21.37 ± 1.83	8.24 ± 0.48	11.82	51.66
MDGFM	63.92 ± 1.59	66.73 ± 2.03	48.40 ± 5.08	82.75 ± 1.76	88.27 ± 1.37	73.83 ± 2.23	83.02 ± 2.85	53.66 ± 1.62	66.12 ± 4.29	22.59 ± 2.33	8.91 ± 2.66	13.18	33.96
G2T-LimiX (ICL)	81.85 ± 0.33	89.19 ± 0.18	81.09 ± 0.09	93.88 ± 0.07	95.72 ± 0.03	90.80 ± 0.09	94.27 ± 0.07	84.61 ± 0.11	91.66 ± 0.05	44.65 ± 0.18	15.46 ± 0.93	5.82	92.33
TAG-TabPFNv2 (ICL)	81.61 ± 0.29	87.80 ± 0.20	82.59 ± 0.12	91.81 ± 0.23	94.41 ± 0.06	90.18 ± 0.10	92.53 ± 0.96	86.22 ± 0.19	93.11 ± 0.17	43.97 ± 0.52	15.07 ± 1.32	6.27	91.13
TAG-LimiX (ICL)	80.03 ± 0.26	88.09 ± 0.18	82.64 ± 0.17	91.60 ± 0.20	94.19 ± 0.08	89.85 ± 0.37	91.32 ± 1.06	85.66 ± 0.23	92.85 ± 0.14	44.89 ± 0.15	15.53 ± 1.99	6.36	90.65
GraphPFN (ICL)	80.18 ± 0.61	90.36 ± 0.13	81.69 ± 0.15	94.29 ± 0.05	96.18 ± 0.07	90.80 ± 0.17	94.95 ± 0.07	84.34 ± 0.20	91.70 ± 0.21	44.52 ± 0.27	20.79 ± 0.49	4.82	95.28
G2T-LimiX (FT)	82.83 ± 0.39	90.88 ± 0.14	83.11 ± 0.39	94.38 ± 0.12	95.93 ± 0.16	91.41 ± 0.16	94.76 ± 0.11	87.10 ± 0.34	92.53 ± 0.13	46.05 ± 0.23	20.83 ± 0.44	2.55	98.02
GraphPFN (FT)	83.04 ± 0.66	90.36 ± 0.13	83.53 ± 0.25	94.89 ± 0.09	96.32 ± 0.08	91.39 ± 0.16	94.98 ± 0.14	87.91 ± 0.30	93.94 ± 0.12	45.89 ± 0.53	22.76 ± 0.55	1.45	99.76

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 some 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`, `aiports-usa`, `aiports-brazil`, `aiports-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 (`aiports-usa`, `airports-brazil`, `aiports-europe`, `deezer-hu`, `deezer-hr`, `deezer-ro`, `ogbn-proteins`), some are synthetic or semi-synthetic (`minsweeper`, `roman-empire`). Popular heterophilous node property prediction datasets have been criticized by Platonov et al. (2023) who pointed 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 likely indicates a bug in the data collection process and leads to train-test data leakage as duplicated nodes appear in both train and test data parts (`squirrel`, `chameleon`). While Platonov et al. (2023) 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 for there appearing duplicated nodes is not known and thus it is also not known whether the datasets with these nodes removed are meaningful, yet some works use these filtered dataset versions for evaluation.

Further, classic node property prediction datasets focus on node classification tasks with node regression being almost not represented at all, despite it being a wide-spread task in real-world applications. It is often considered that regression tasks might require specialized machine learning methods for achieving the best performance on them (Zhang et al., 2023), thus representing such tasks in benchmarks is important.

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.

Some works (Xia et al., 2024; Xia & Huang, 2024) compare the performance of GFMs and GNNs in node classification in few-shot settings, i.e., with only a few labeled training nodes available (in case of Xia et al. (2024), with 1 or 5 labeled training nodes per class). Such few-shot data splits strongly favor models performing in-context learning (i.e., GFMs), since models performing classic learning by optimizing their weights (i.e., GNNs) tend to strongly overfit when only a few training samples are available. Evaluating and improving few-shot performance of in-context learning methods is a valuable research goal in its own right, however, we note that conclusions drawn from few-shot settings may not transfer to the data-rich scenarios typical of large-scale industrial applications, and this distinction should be made explicit. In particular, specific GFMs outperforming GNNs in node property prediction in few-shot setting does not mean that these GFMs become better models for the most wide-spread and impactful industrial applications of node property prediction. Further, we note

that some works (Xia et al., 2024; Xia & Huang, 2024; Hayler et al., 2025) refer to their GFMs as performing 'zero-shot' predictions; however, this terminology may cause confusion, as these models do require labeled examples at inference time, and thus their prediction regime is more accurately described as in-context learning, while 'zero-shot' typically refers to models making predictions with no labeled training examples available, which to the best of our knowledge no current non-LLM-based (and thus not limited to text-attributed graphs) GFM for node property prediction is capable of.

Recently, there has been a lot of critic of standard datasets in Graph Machine Learning and attempts to create better benchmarks (Bechler-Speicher et al., 2025; Bazhenov et al., 2025; Stoll et al., 2025). GraphLand (Bazhenov et al., 2025) and GraphBench (Stoll et al., 2025) provide benchmarks with more practically-relevant tasks, significantly larger diversity across various aspects, fixed realistic data splits, and predefined evaluation metrics. GraphLand in particular focuses on node property prediction and, in our opinion, is highly suitable for evaluating GFMs for this task.

B.2 Pitfall 2: Evaluation metric selection

Most currently available node property prediction datasets are node classification datasets. The simplest metric for classification is Accuracy, which is the proportion of correct class predictions. Accuracy assumes that all incorrect predictions have the same cost, i.e., decrease the metric by the same value. Whether this assumption is justified, and thus whether Accuracy is an appropriate metric, depends on the specific task. We believe that for most currently available multiclass node classification datasets, Accuracy can be considered an appropriate metric, as these datasets represent tasks like product category prediction in co-purchasing networks, website topic prediction in web networks, location prediction in social networks, and paper topic prediction in citation networks. For these tasks, it seems reasonably fair to assume equal costs for incorrect predictions for different classes. However, the situation is vastly different for the currently available binary classification datasets. Most of them are highly imbalanced, with positive samples constituting a much smaller portion of the dataset than negative samples. For binary classification, we can separate all predictions into True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) predictions. Then, Accuracy can be defined as $\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$. Thus, equal costs are assigned to FP and FN predictions. However, this is almost never the case for real-world imbalanced binary classification tasks. For example, in node property prediction, common examples of binary classification tasks include fraud detection, churn prediction, advertisement targeting, explicit content detection. In these tasks, the costs of FP and FN prediction are different. For such cases, metrics like ROC-AUC and Average Precision are more appropriate. Both of these metrics are computed by measuring the area under a certain curve on a 2-dimensional plot representing the performance of a binary classifier at various classification threshold values. ROC-AUC (also known as AUROC) is the area under the Receiver Operating Characteristic curve, which is the plot of the True Positive Rate (TPR) against False Positive Rate (FPR), where $\text{TPR} = \frac{\text{TP}}{\text{TP+FN}}$ and $\text{FPR} = \frac{\text{FP}}{\text{TN+FP}}$. Average Precision (also known as PR-AUC or AUPR) is the area under the Precision-Recall curve, which is the plot of Precision against Recall (which is another name for TPR), where $\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}$ and $\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$. There is current research into the properties of ROC-AUC and Average Precision metrics (Mihelich et al., 2024; McDermott et al., 2024), but it is generally agreed upon that Average Precision is preferable when there is a need to reduce False Positives, as this metric assigns higher costs to incorrect high-confidence positive predictions. Some examples of tasks with such a preference are tasks where all positive predictions must be verified and the verification cost is relatively high, or retrieval-like tasks where a small number of most relevant samples need to be found (i.e., "searching for needles in a haystack" tasks), or simply tasks where there is a relatively small proportion of positive samples and the financial, reputational, or social costs of False Positive predictions are high. These settings are typical for many industrial binary node classification tasks such as those represented in graph node classification datasets in the GraphLand benchmark, and thus Average Precision seems a more appropriate metric for most of them.

In real-world scenarios, examining single-value metrics is often not enough and much more care about selecting algorithms for each particular task is typically taken, such as examining confusion matrices and/or Precision-Recall and Receiver Operating Characteristic curves. However, we understand that for evaluation of general machine learning methods not tied to a particular application a single-value metric is highly desirable, especially when evaluation is performed across a large

Table 5: Reproduction of Table 1 from (Finkelshtein et al., 2025) with 3 additional rows marked with *. The results of MeanGNN, GAT, GraphAny, TS-Mean, and TS-GAT on 28 node classification datasets (accuracy is reported for all datasets), with 4 ways to aggregate these results across datasets (1 original, 3 added by us). In each of the rows with aggregated results, the 3 best results are highlighted with color: first, second, third.

	MeanGNN	GAT	GraphAny	TS-Mean	TS-GAT
actor	32.03 ± 0.29	32.59 ± 0.83	27.54 ± 0.20	28.09 ± 0.93	28.80 ± 2.12
amazon-ratings	40.74 ± 0.13	40.63 ± 0.66	42.80 ± 0.09	42.27 ± 1.40	41.92 ± 0.47
arxiv	50.94 ± 0.38	57.93 ± 3.44	58.85 ± 0.03	56.33 ± 2.58	53.13 ± 2.04
blogcatalog	84.48 ± 0.74	78.20 ± 7.23	71.54 ± 3.04	76.30 ± 2.92	78.44 ± 5.18
brazil	32.31 ± 7.50	35.38 ± 4.21	33.84 ± 15.65	39.23 ± 5.70	36.92 ± 10.03
chameleon	61.75 ± 0.94	58.46 ± 6.23	63.64 ± 1.48	60.83 ± 5.41	54.65 ± 4.58
citeseer	65.06 ± 1.30	63.92 ± 0.84	68.88 ± 0.10	68.66 ± 0.19	68.70 ± 0.48
co-cs	80.87 ± 0.69	82.28 ± 0.86	90.06 ± 0.80	90.92 ± 0.47	91.13 ± 0.43
co-physics	79.05 ± 1.13	85.92 ± 1.10	91.85 ± 0.34	92.61 ± 0.61	92.23 ± 0.61
computers	73.88 ± 0.88	70.94 ± 3.40	82.79 ± 1.13	81.37 ± 1.25	80.23 ± 2.44
cornell	63.78 ± 1.48	69.73 ± 2.26	63.24 ± 1.32	68.65 ± 2.42	71.35 ± 4.52
deezer	54.91 ± 1.81	55.22 ± 2.33	51.82 ± 2.49	52.31 ± 2.52	52.88 ± 2.86
europe	39.12 ± 6.44	39.00 ± 4.30	41.25 ± 7.25	35.88 ± 6.91	38.38 ± 6.32
full-DBLP	65.01 ± 2.40	67.34 ± 2.75	71.48 ± 1.44	66.42 ± 3.65	64.30 ± 3.75
full-cora	55.85 ± 1.04	59.95 ± 0.88	51.18 ± 0.78	53.58 ± 0.73	52.82 ± 0.90
last-fm-asia	77.23 ± 1.01	72.65 ± 0.48	81.14 ± 0.42	78.03 ± 1.02	76.60 ± 1.44
minesweeper	84.06 ± 0.18	84.15 ± 0.24	80.46 ± 0.11	80.68 ± 0.38	80.72 ± 0.78
photo	88.95 ± 1.08	80.78 ± 3.59	89.91 ± 0.88	90.18 ± 1.30	89.66 ± 1.23
pubmed	74.56 ± 0.13	75.12 ± 0.89	76.46 ± 0.08	74.98 ± 0.56	75.46 ± 0.86
questions	97.16 ± 0.06	97.13 ± 0.05	97.07 ± 0.03	97.02 ± 0.01	97.02 ± 0.01
roman-empire	69.37 ± 0.66	69.80 ± 4.18	63.34 ± 0.58	66.36 ± 1.02	67.76 ± 0.60
squirrel	43.32 ± 0.66	38.16 ± 1.04	49.74 ± 0.47	41.81 ± 0.80	38.29 ± 4.29
texas	76.76 ± 1.48	81.62 ± 6.45	71.35 ± 2.16	73.51 ± 4.01	74.05 ± 4.10
tolokers	78.59 ± 0.66	78.22 ± 0.37	78.20 ± 0.03	78.12 ± 0.09	78.01 ± 0.23
usa	43.93 ± 1.16	43.03 ± 2.08	43.35 ± 1.62	42.34 ± 2.12	41.69 ± 2.59
wiki-attr	74.23 ± 0.89	68.91 ± 9.50	60.27 ± 3.06	69.89 ± 1.31	72.53 ± 1.59
wiki-cs	71.97 ± 1.70	74.99 ± 0.59	74.11 ± 0.60	74.16 ± 2.07	73.24 ± 2.49
wisconsin	74.12 ± 12.20	73.33 ± 8.27	59.61 ± 5.77	61.18 ± 11.38	65.88 ± 9.86
average accuracy	65.50 ± 1.75	65.55 ± 2.82	65.56 ± 1.86	65.78 ± 2.28	65.60 ± 2.74
average rank*	2.89	2.86	2.96	3.07	3.21
average normalized score*	52.53	54.71	54.17	48.95	45.17
Elo rating*	1603	1569	1443	1452	1433

number of datasets. Yet we believe the choice of this single-value metric for each datasets should be made with care.

B.3 Pitfall 3: Results aggregation across datasets

In this section, we illustrate that using different approaches to aggregating results across datasets can lead to drastically different conclusions. This issue frequently occurs when several models are compared on a set of diverse datasets since it is rarely the case that one model achieves the best results on all the datasets. As one example, we consider the results reported by Finkelshtein et al. (2025). In Table 5, we reproduce Table 1 from Finkelshtein et al. (2025), but with 3 additional rows (the 3 lowest rows marked with *) in which we provide 3 more approaches to aggregating results across multiple datasets. The table reports experimental results of 5 models across 28 datasets. Originally, the results across datasets are aggregated by computing averages of Accuracy values across the datasets, which, as we have discussed in Section 3.3, is not necessarily a reliable way to compare models. We additionally consider three other aggregation approaches: average rank, average normalized score, and Elo Rating. As shown in Table 5, all four aggregations provide different rankings: according to average Accuracy, the best two models are TS-Mean and TS-GAT; according to average rank — GAT and MeanGNN; according to average normalized score — GAT and GraphAny; and according to the Elo ratings[2] — MeanGNN and GAT. Finally, the Friedman test

[2]The Elo ratings were computed using the Elo library: `https://github.com/ddm7018/Elo`.

(the implementation from the SciPy library (Virtanen et al., 2020) was used[3]) cannot reject the null hypothesis that the results of all 5 models are on the same level (p-value equals 0.925), although we note that the Friedman test is rather conservative, and further its results in this particular case can be not very reliable due to a relatively small number of models being compared. These results demonstrate that vastly different model rankings and thus different conclusions can be obtained when using different approaches to aggregating experimental results across multiple datasets. We believe identifying the most reliable approach(es) to aggregating results across datasets is currently an open research question.

C 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.

D Prior-data Fitted Networks

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

[3] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.friedmanchisquare.html`

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 4 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.

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/validation/test splits. For the classic node classification datasets, we also use $10\%/10\%/80\%$ random stratified train/validation/test splits, which in this case are created by us. For each dataset and model, we report the mean and standard deviation of the obtained results computed over 10 runs.

Hyperparameter selection is extremely important for achieving strong performance with GNNs, and very different hyperparameters can be optimal for different datasets. Thus, for the baseline GNNs (GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), and GAT (Veličković et al.,

Table 6: 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.

	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
Improved-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
Improved-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
Improved-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
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

2018)) we run 100 trials of hyperparameter optimization per 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). The exact hyperparameters we tune follow (Luo et al., 2024) and include: learning rate, dropout probability, weight decay, number of GNN layers, hidden dimension, number of input layers, whether ego- and neighbor-embedding separation (Hamilton et al., 2017; Zhu et al., 2020; Platonov et al., 2023) is used, whether BatchNorm (Ioffe & Szegedy, 2015), LayerNorm (Ba et al., 2016), or no normalization is used, number of attention heads (only for GAT). 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 in-context learning (ICL) regime, or fine-tuning (FT) regime, or both. One of the benefits of the ICL regime is that it does note 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, 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: $\{10^{-6}, 3 \times 10^{-6}, 5 \times 10^{-6}, 10^{-5}, 3 \times 10^{-5}, 5 \times 10^{-5}, 10^{-4}, 3 \times 10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$. 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. These models use an ensemble of multiple inference runs by default, and we also use this setting. 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), PyTorch Geometric (Fey & Lenssen, 2019), and DGL (Wang et al., 2019). All experiments were run on NVIDIA Tesla A100 80GB GPUs.

F Memory requirements of GNNs and PFN-based GFMs

In Table 6, 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.

Table 7: 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

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 7. We can see that inference-time ensembling almost always improves the predictive performance of the 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 Limitations

In our work, we have reevaluated 9 GFMs for node property prediction from prior literature which represent a number of very different types of models. However, development of GFMs for node property prediction is an active research field with new models regularly appearing. Thus, it is likely that we have not considered all the currently available models. Further, some works proposing GFMs for node property prediction do not release their implementations, which makes reevaluation of these models impossible.

Designing a fair and reliable experimental setup is extremely important for model evaluation in machine learning, but, as our work demonstrates, it is also far from being simple. While we have highlighted 5 common pitfalls and attempted to fix them in our own experimental setup, there are possibly other important improvements worth applying to experimental setups in the field including our own. Yet we believe our work provides as of now the most rigorous and reliable evaluation of GFMs for node property prediction.

I REPRODUCIBILITY

The codebase used to run experiments with all the considered GNNs is provided in the following anonymous repository: `https://anonymous.4open.science/r/pitfalls-submit-4D7X`. For running experiments with all the considered GFMs, their corresponding official open source implementations were used.