Of Graphs and Tables: Zero-Shot Node Classification with Tabular Foundation Models

Adrian Hayler
University of Oxford

Xingyue Huang
University of Oxford

İsmail İlkan Ceylan
TU Wien / AITHYRA

Michael Bronstein
University of Oxford / AITHYRA

Ben Finkelshtein
University of Oxford

Abstract

Graph foundation models (GFMs) have recently emerged as a promising paradigm for achieving broad generalization across various graph data. However, existing GFMs are often trained on datasets that were shown to poorly represent real-world graphs, limiting their generalization performance. In contrast, tabular foundation models (TFMs) not only excel at classical tabular prediction tasks but have also shown strong applicability in other domains such as time series forecasting, natural language processing, and computer vision. Motivated by this, we take an alternative view to the standard perspective of GFMs and reformulate node classification as a tabular problem. Each node can be represented as a row with feature, structure, and label information as columns, enabling TFMs to directly perform *zero-shot* node classification via in-context learning. In this work, we introduce TabGFM, a graph foundation model framework that first converts a graph into a table via feature and structural encoders, applies multiple TFMs to diversely subsampled tables, and then aggregates their outputs through ensemble selection. Through experiments on 28 real-world datasets, TabGFM achieves consistent improvements over task-specific GNNs and state-of-the-art GFMs, highlighting the potential of tabular reformulation for scalable and generalizable graph learning.

1 Introduction

Graph foundation models (GFMs) have recently emerged as a central research direction in graph machine learning [1–3]. However, their effectiveness on node classification benchmarks has so far been limited, with performance improvements over standard task-specific graph neural networks (GNNs) [4–6] often being marginal [7, 8]. A growing body of work argues that this shortfall is primarily due to the characteristics of the training data: many available pretraining graphs are small in scale, contain outdated node features, and rely on heuristic or artificial topological structures, making them poor representatives of real-world graphs [9–11].

In contrast, tabular foundation models (TFMs) have demonstrated broad applicability by representing heterogeneous domains in a unified tabular format. This perspective has enabled strong generalization in settings as diverse as computer vision [12], natural language processing [13–15]. This motivates our central research question: *Can the generalization strengths of tabular foundation models be effectively leveraged to build a graph foundation model for node classification?*

To address this question, we first observe that once the graph's topological structure has been exploited, either through neighbor aggregation or through least-squares solutions in GraphAny [7] and TS-GNNs [8], the node classification problem naturally reduces to a classification task over the set of feature vectors. At this stage, the objective is to map these vectors to labels. In traditional

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: New Perspectives in Advancing Graph Machine Learning.

GNNs and TS-GNNs, this step is typically handled by a lightweight classifier, whereas GraphAny relies on a more elaborate attention mechanism to obtain the final predictions. Thus, this perspective naturally aligns node classification with tabular learning and opens the door to leveraging TFMs.

Motivated by this perspective, we introduce *TabGFM*, a graph foundation model framework that reinterprets node classification as an ensemble learning problem over tabular models. TabGFM first converts the input graph to a table by computing node-level features using pre-defined feature and structural encoders, as shown in Figure 1. However, the resulting tables are often too large and contain diverse features and label spaces that current TFMs [16] are not designed to handle. We employ an ensemble aggregation strategy: subsampling multiple smaller, size-constrained tables, applying TFM to each table to obtain individual predictions, and aggregating them via ensemble selection [17]. The resulting design unifies graph-specific insights with advances in tabular learning, yielding a single, generalizable graph foundation model that **does not require pretraining on graph data, yet still outperforms the state-of-the-art GFMs and task-specific GNNs by 7%.**

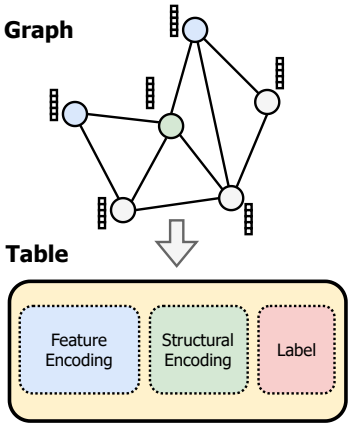

Figure 1: An illustration of how TabGFM transforms a graph into a tabular representation using feature and structural encoders.

Contributions. Our work makes the following contributions toward building generalizable GFMs:

1. We are the first to formulate *node classification* as a *tabular classification* problem, which enables the use of *tabular foundation models* trained exclusively on tabular data for *zero-shot inference* on arbitrarily large unseen graphs.
2. We introduce TabGFM, a graph foundation model framework that represents nodes as rows in a table and adapts TFMs to node classification via subsampling and ensemble aggregation.
3. We evaluate our framework on 28 real-world node classification datasets, demonstrating significant improvements over existing GFMs and task-specific GNNs, **improving the averaged accuracy from** 65.78% **to** 73.10%**.**

2 Related work

Graph foundation models for node classification. Graph Neural Networks (GNNs) are the dominant approach for graph machine learning tasks. However, these models are typically trained separately on each dataset and lack the ability to generalize across different feature and label spaces. Graph Foundation Models (GFMs) have emerged [1–3] to address this gap by learning transferable representations from diverse graph data. GFMs have shown strong performance on the *label inpainting* problem [8], a subtask of inductive node-based learning where the test-time graph contains partially observed features, analogous to image inpainting. One of the first attempts in this direction is GraphAny [7], which characterizes the natural symmetries required of a GFM: node permutation-equivariance, label permutation-equivariance, and feature permutation-invariance. It then constructs an ensemble by combining the closed-form solutions of multiple least-squares models that respect these symmetries. An attention mechanism is used to weight the ensemble components, and the resulting model has been shown to generalize to unseen graphs with arbitrary feature and label spaces. The subsequent TS-GNN model [8] formalizes the aforementioned symmetries into a theoretically grounded framework for GFMs. It derives linear layers that respect the same symmetry constraints, proves the universality of the resulting architecture over multi-sets, and shows empirically that performance improves as the number of training graphs increases. Our proposed method TabGFM, incorporates tabular foundation models to allow for a richer set of ensemble models, leading to significantly improved generalization capabilities and empirical performance.

Tabular foundation models and their application. A tabular learning problem involves predicting labels from data organized in a table, where each row corresponds to an instance and columns correspond to features. Unlike structured domains such as images or text, tabular data are heterogeneous and lack strong inductive biases, which historically limited transferability across datasets.

The emergence of Prior-Data Fitted Networks (PFNs) [18] gave rise to a series of tabular foundation models (TFMs), which are trained entirely on synthetic datasets sampled from structural causal models and can solve tabular learning problems via in-context learning in a single forward pass. Building on this paradigm, TabPFN [16, 19] pioneered the field by training a transformer on a massive collection of synthetic datasets; showing strong one-shot performance. TabICL [20] further advanced this paradigm by employing efficient factorized attention, allowing it to scale to larger datasets.

Beyond classical tabular benchmarks, TFMs have shown surprising generalization capabilities in other domains, including computer vision [12], natural language processing [13–15], and time-series forecasting [21]. These results suggest that TFMs are not limited to standard tabular tasks, but can serve as a unifying framework for heterogeneous data. Our proposed model, TabGFM, builds on this insight by framing node classification as a tabular learning problem, enabling TFMs to adapt to new graphs without retraining and bridging their advances with the challenges of graph machine learning.

3 Background: Graphs, Label Inpainting and TabPFN

Graphs. We consider a simple, undirected[1], unweighted graph $G = (V, E, \boldsymbol{X}, \boldsymbol{Y})$ with N nodes. The graph structure (V, E) is represented by an adjacency matrix $\boldsymbol{A} \in \{0,1\}^{N\times N}$. Each node is equipped with a feature vector and a class label: the feature vectors are collected in the matrix $\boldsymbol{X} \in \mathbb{R}^{N\times F}$, and the one-hot encoded labels across C classes are given by $\boldsymbol{Y} \in \{0,1\}^{N\times C}$. The (random-walk) *normalized adjacency matrix* is defined as $\hat{\boldsymbol{A}} = \boldsymbol{D}^{-1}\boldsymbol{A}$, where $\boldsymbol{D}$ is a diagonal degree matrix $\boldsymbol{D} = \mathrm{diag}(d_1, \ldots, d_n)$ with $d_i = \sum_{j=1}^{n} \boldsymbol{A}_{ij}$ denoting the degree of node i. Given a matrix $\boldsymbol{M} \in \mathbb{R}^{N\times D}$ and a subset of nodes $S \subseteq V$, we write $\boldsymbol{M}_S \in \mathbb{R}^{|S|\times D}$ for the submatrix consisting of the rows of $\boldsymbol{M}$ indexed by S. For a single node $v \in V$, the corresponding row vector is denoted by $\boldsymbol{M}_v \in \mathbb{R}^D$. Lastly, we denote $[N] = \{1, 2, \ldots, N\}$.

Label Inpainting. We study the label inpainting [7, 8] setting, a subtask of inductive node classification. Let $L \subset V$ denote the set of *labeled nodes* with labels $\boldsymbol{Y}_L \in \mathbb{R}^{|L|\times C}$, and let $Q \subseteq V \setminus L$ denote the set of *query nodes*. The goal is to predict the missing labels $\boldsymbol{Y}_Q \in \mathbb{R}^{|Q|\times C}$ for the query nodes, given the existing labels. Unlike the classical semi-supervised regime, which assumes a fixed set of training labels, label inpainting treats the labeled nodes themselves as part of the input, and the test graphs are also partially labeled. This formulation enables generalization across varying label sets and unseen graphs: given a partially labeled graph, the task is to "fill in" the missing labels, analogous to inpainting in computer vision, by leveraging both node features and structural information.

TabPFN. TabPFN, and PFNs more generally, cast tabular learning as an in-context learning problem. The training dataset $(\boldsymbol{X}_L, \boldsymbol{Y}_L) \in \mathbb{R}^{|L|\times(F+C)}$ is provided to TabPFN as a *context*[2], analogous to few-shot examples in large language models. Additionally, a set of *query rows* $\boldsymbol{X}_Q \in \mathbb{R}^{|Q|\times F}$ is provided for which TabPFN returns a per-node probability distribution $\mathrm{TabPFN}((\boldsymbol{X}_L, \boldsymbol{Y}_L), \boldsymbol{X}_Q) = \hat{\boldsymbol{Y}}_Q \in \mathbb{R}^{|Q|\times C}$. Each query row's $q \in Q$ prediction $\hat{\boldsymbol{Y}}_q \in \mathbb{R}^C$ is an approximation of the posterior predictive distribution (PPD) $p(\boldsymbol{Y}_q | \boldsymbol{X}_q, (\boldsymbol{X}_L, \boldsymbol{Y}_L))$, which implicitly encodes strong tabular priors from synthetic data-generating process used during training [18].

4 The TabGFM framework

In this section we present *TabGFM*, a framework that reinterprets node classification as an ensemble learning problem over tabular models. Given a graph $G = (V, E, \boldsymbol{X}, \boldsymbol{Y})$, our key idea is to reduce graph-structured data into a tabular form that can be directly processed by powerful tabular foundation models. TabGFM decouples the problem into three stages:

1. **Node-level encoding:** Each node is represented as a row in a table by concatenating its label (if available) with multiple feature and structure encoders. This step transforms graph information into a tabular format that can be processed by tabular models.

[1] All results naturally extend to directed graphs; we focus on undirected graphs for ease of presentation.

[2] While TabPFN is designed to operate directly on the raw labels, for notational convenience we assume throughout that the model instead receives one-hot encoded label representations.

2. **Tabular learning:** Since TFMs are limited to relatively small tables, we construct multiple subsampled tables by selecting subsets of columns and labeled rows, and obtain separate predictions from the TFM given each subsample.
3. **Ensemble aggregation:** Predictions from the subsampled tables are combined through ensemble selection, which uses predictions on held-out data to determine ensemble weights that both account for model quality and model interactions. The final prediction for each query node is obtained as a weighted combination of these outputs.

This modular design allows TabGFM to exploit the strengths of tabular foundation models while remaining a lightweight and training-free graph foundation model.

4.1 Node-level encoding

Given $G = (V, E, \boldsymbol{X}, \boldsymbol{Y})$, we employ I feature encoders $\{\phi^{(i)}\}_{i=1}^{I} : \mathbb{R}^{N\times(F+N)} \to \mathbb{R}^{N\times D_1}$ and J structure encoders $\{\psi^{(j)}\}_{j=1}^{J} : \mathbb{R}^{N\times N} \to \mathbb{R}^{N\times D_2}$. These produce a tabular representation $\boldsymbol{T} \in \mathbb{R}^{N\times(ID_1+JD_2+C)}$, where each node $v \in V$ corresponds to a row defined as

$$\boldsymbol{T}_v = \Big(\phi_v^{(1)}(\boldsymbol{X}, \boldsymbol{A}), \ldots, \phi_v^{(I)}(\boldsymbol{X}, \boldsymbol{A}), \psi_v^{(1)}(\boldsymbol{A}), \ldots, \psi_v^{(J)}(\boldsymbol{A}), \boldsymbol{Y}_v\Big),$$

where $\phi_v^{(i)}(\boldsymbol{X}, \boldsymbol{A}) \in \mathbb{R}^{D_1}$ and $\psi_v^{(j)}(\boldsymbol{A}) \in \mathbb{R}^{D_2}$ denote the outputs of the i-th feature encoder and the j-th structure encoder for node v, respectively. The term $\boldsymbol{Y}_v$ is the one-hot label encoding if $v \in L$, and the zero-vector otherwise.

Feature encoders. We include the raw node features $\phi_v^{(0)}(\boldsymbol{X}, \boldsymbol{A}) = \boldsymbol{X}$. To incorporate local structural information, we add neighborhood-smoothed features, which have been shown to improve performance. Specifically, we use k-order neighborhood averages for $k \in \{1, \cdots, 4\}$, i.e., $\phi_v^{(k)}(\boldsymbol{X}, \boldsymbol{A}) = \hat{\boldsymbol{A}}^k \boldsymbol{X}$, where $\hat{\boldsymbol{A}}$ is the normalized adjacency matrix. Beyond such smoothed features, we also employ LinearGNNs [7] as feature encoders.

Structure encoders. Unlike GNNs, TFMs operate on sets of rows and are thus unaware of the underlying topology. To reintroduce this information, we employ well-established structural encoders designed to capture both local and global graph structure. Specifically, we include (1) RandomWalkPE [22], denoted as $\psi_v^{(1)}$, which encodes local structures, (2) LaplacianEigenvectorPE [23], denoted as $\psi_v^{(2)}$, the top-20 eigenvectors of the normalized Laplacian providing smooth global encodings, and (3) GPSE [24], denoted as $\psi_v^{(3)}$, pretrained embeddings from frozen message-passing networks that have been shown to be strong general-purpose structure encoders.

Remark 4.1. *The construction of both feature and structure encodings is* lightweight and training-free*, relying only on closed-form computations.*

4.2 Tabular learning and ensemble aggregation

Tabular learning. The tabular representation of a graph is denoted by $\boldsymbol{T} = (\boldsymbol{Z}, \boldsymbol{Y})$, where $\boldsymbol{Z} \in \mathbb{R}^{N\times(ID_1+JD_2)}$ stores the *row features* obtained by concatenating all feature and structure encodings, and $\boldsymbol{Y} \in \{0, 1\}^{N\times C}$ contains the corresponding row one-hot label vectors. We distinguish between *labeled rows* $(\boldsymbol{Z}_L, \boldsymbol{Y}_L) \in \mathbb{R}^{|L|\times(ID_1+JD_2+C)}$, which include both row features and labels, and *query rows* $\boldsymbol{Z}_Q \in \mathbb{R}^{|Q|\times(ID_1+JD_2)}$, which include only row features. The task is to inpaint the missing labels $\hat{\boldsymbol{Y}}_Q$ for the query rows, which could be achieved by directly applying TabPFN as

$$\hat{\boldsymbol{Y}}_Q \;=\; \mathrm{TabPFN}\big((\boldsymbol{Z}_L, \boldsymbol{Y}_L),\, \boldsymbol{Z}_Q\big). \tag{1}$$

However, TabPFN is limited to relatively small tables, supporting at most $10{,}000$ rows and 500 columns. This restriction prevents it from serving as a general-purpose GFM, since the encoded tables in TabGFM reach up to $90{,}000$ labeled rows and $40{,}000$ columns.

We address this limitation by constructing B smaller subsampled tables $\{\boldsymbol{T}^{(b)}\}_{b=1}^{B}$, each fitting TabPFN's size constraints. Each $\boldsymbol{T}^{(b)}$ is obtained by (1) retaining all unlabeled rows, (2) uniformly subsampling columns, and (3) class-balanced subsampling of labeled rows to preserve its distribution. TabPFN is then applied to each $\boldsymbol{T}^{(b)}$, producing predictions for the query rows $\hat{\boldsymbol{Y}}_Q^{(b)}$.

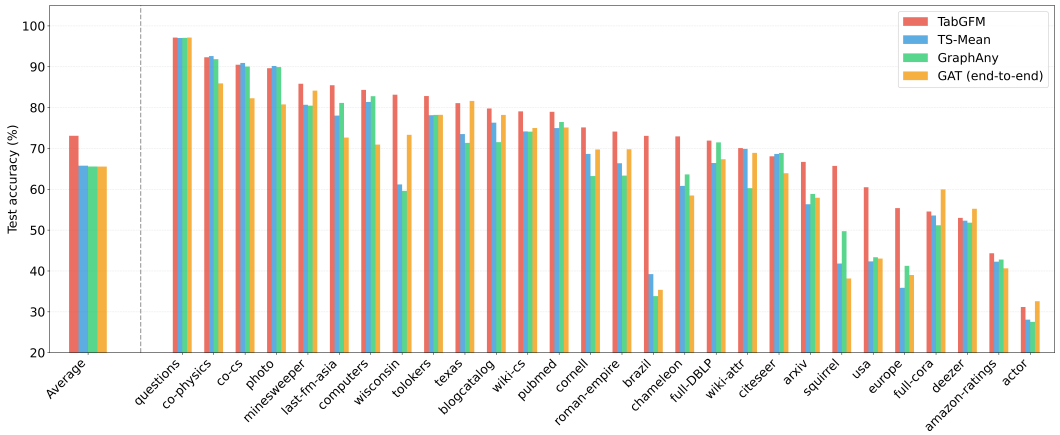

Figure 2: Zero-shot accuracy for the baselines GraphAny and TS-Mean, and our TabGFM, compared with end-to-end test accuracy of GAT.

Ensemble aggregation. We aggregate the predictions through an affine combination, where the contribution of each predictor is determined via *ensemble selection* [17]. A *hold-out set* $H \subset L$, is randomly sampled from the labeled nodes, with the remaining nodes $A = L \setminus H$ called *anchor nodes*. Given TabPFN's limitations, we cannot directly provide $(\boldsymbol{X}_A, \boldsymbol{Y}_A)$ as context to the models. Therefore, as described above, each TabPFN model independently generates its own subsampled table $\boldsymbol{T}_{\text{ES}}^{(b)}$ as described in the previous section, where the subset of columns sampled per model stays fixed between ensemble selection and inference. We then create held-out predictions per model:

$$\hat{\boldsymbol{Y}}_H^{(b)} \;=\; \text{TabPFN}\big(\boldsymbol{T}_{\text{ES}}^{(b)}, \boldsymbol{Z}_H^{(b)}\big),$$

where $\boldsymbol{Z}_H^{(b)}$ denotes the column-subsampled features of the held-out rows. Ensemble selection [17] approximates the weights $\hat{w}^{(b)}$ through greedy forward selection of the following intractable problem

$$(\hat{w}^{(1)}, \ldots, \hat{w}^{(B)}) = \underset{\substack{\forall b \in [B],\, w^{(b)} \in \mathbb{R} \\ \sum_{b=1}^{B} w^{(b)} = 1}}{\arg\max} \; \text{Acc}\Big(\sum_{b=1}^{B} w^{(b)} \hat{\boldsymbol{Y}}_H^{(b)}, \boldsymbol{Y}_H\Big),$$

where Acc is the accuracy function and $\boldsymbol{Y}_H$ denotes the labels of the held-out rows. The optimized weights are then used to combine the query predictions, yielding the final ensemble prediction as

$$\hat{\boldsymbol{Y}}_Q = \sum_{b=1}^{B} \hat{w}^{(b)} \hat{\boldsymbol{Y}}_Q^{(b)}.$$

Remark 4.2. *While we focus on TabPFN for its zero-shot in-context prediction, our framework is general and can accommodate alternative tabular learners (e.g., gradient-boosted trees).*

5 Experiments

In this section, we would like to answer the following questions:

Q1 How do TabGFM's zero-shot generalization capabilities compare against existing GFMs?

Q2 Is TabGFM able to improve performance as the number of subsampled tables increases?

Datasets. Following Finkelshtein et al. [8], we evaluate on 28 diverse node-classification datasets using their official splits. These include *brazil*, *usa*, and *europe* [25]; *chameleon*, *squirrel* [26]; *roman-empire*, *amazon-ratings*, *minesweeper*, *questions*, and *tolokers* [10]. We further employ *wiki-attr* and *blogcatalog* [27]; *cornell*, *wisconsin*, *texas*, and *actor* [28]; and the classical benchmarks *cora*, *citeseer*, and *pubmed* [29]. In addition, we consider *co-cs*, *co-physics*, *computers*, and *photo* [30]; *full-DBLP* and *full-cora* [31]; *wiki-cs* [32]; and *last-fm-asia* and *deezer* [33]. Finally, we include the large-scale *arxiv* dataset [34]. Dataset statistics can be found in Appendix C.

Setup. We employ TabGFM using an ensemble of 10 subsampled tables combined with 8 additional LinearGNNs, comparing it against GraphAny [7] and TS-Mean [8], both trained on Cora, and GAT [5] trained end-to-end. All results reflect the mean accuracy and standard deviation over 5 random seeds. Further implementation details can be found in Appendix B.

5.1 How do TabGFM's zero-shot generalization capabilities compare against existing GFMs?

Results. TabGFM displays strong performance on all datasets, either performing comparable with or outperforming both GraphAny and TS-Mean, leading to an **average test accuracy of 73.10%** in comparison to 66.55% for GAT, 65.56% for GraphAny and 65.78% for TS-Mean, a difference of **over 7% mean accuracy.** As a consequence, TabGFM is the **first GFM to demonstrate substantial improvements over end-to-end baselines (Q1)**. TabGFM widens the gap to end-to-end baselines, increasing from less than 0.3% to over 7.5%.

We believe that the significant difference in performance can be attributed to the much larger scale of training data available for training the tabular foundation models at the core of TabGFM. For instance, TabPFN was trained on 130 million synthetic datasets, which allows it to experience a larger data distribution. In contrast, the GFM pretrained on the largest number of datasets to date is a variant of TS-Mean, which was trained on nine datasets. While training on additional datasets does increase its performance [8], TS-Mean's average accuracy of 68.57% remains substantially below TabGFM's mean accuracy of 74.39% on the remaining 20 datasets. Given the limited number of publicly available graph datasets, we hypothesize that it is unlikely that GFMs trained solely on (real-world) graph data will be able to match the performance of TabGFM.

5.2 Is TabGFM able to improve performance as the number of subsampled tables increases?

In addition to the previously described setup, we now vary the number of TabPFN models and associated subsampled tables B from 1 to 10 to understand their impact on TabGFM's performance.

Results. Figure 3 shows that as expected, TabGFM's performance steadily increases with the number of subsampled tables, though the rate of improvement gradually flattens for larger B. This suggests that additional subsampled tables provide useful complementary information (**Q2**). The diminishing gains are likely due to increased redundancy among the tables, particularly on smaller datasets.

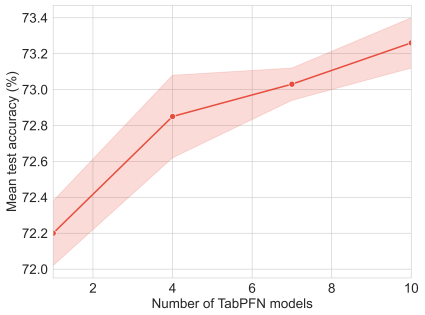

Figure 3: Zero-shot accuracy of TabGFM across 28 datasets as a function of the number of subsampled tables.

6 Conclusions

We introduced TabGFM, a graph foundation model (GFM) framework that reformulates node classification as a tabular label inpainting problem. TabGFM represents nodes as rows in a table through a combination of feature and structural encodings, and leverages pretrained tabular foundation models (TFMs), which are applied to subsampled tables and aggregated through ensemble selection. This approach yields significant improvements over existing GFMs, requires no pretraining on graph data, and still outperforms state-of-the-art GFMs and task-specific GNNs by 7%.

Looking ahead, an important direction for TabGFM is to move beyond its limitation of relying on predefined feature and structural encoders, toward fine-tuning existing TFM over data that captures richer graph information. Our finding suggests that true generalization may come not from a more complex message passing mechanism, but from reframing graph problems into those modalities where foundation models already excel. In this light, another promising avenue is to extend TabGFM to tasks such as link prediction and graph classification, as it is currently limited to node classification, thereby moving toward a more general-purpose GFM.

Acknowledgments

MB is supported by EPSRC Turing AI World-Leading Research Fellowship No. EP/X040062/1 and EPSRC AI Hub on Mathematical Foundations of Intelligence: An "Erlangen Programme" for AI No. EP/Y028872/1. BF is funded by the Clarendon scholarship.

References

[1] Haitao Mao, Zhikai Chen, Wenzhuo Tang, Jianan Zhao, Yao Ma, Tong Zhao, Neil Shah, Mikhail Galkin, and Jiliang Tang. Position: Graph foundation models are already here. In *ICML*, 2024.

[2] Xingyue Huang, Pablo Barcelo, Michael M. Bronstein, Ismail Ilkan Ceylan, Mikhail Galkin, Juan L Reutter, and Miguel Romero Orth. How expressive are knowledge graph foundation models? In *ICML*, 2025.

[3] Xingyue Huang, Mikhail Galkin, Michael M Bronstein, and İsmail İlkan Ceylan. Hyper: A foundation model for inductive link prediction with knowledge hypergraphs. *arXiv*, 2025.

[4] Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[5] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[6] Ben Finkelshtein, Xingyue Huang, Michael M Bronstein, and Ismail Ilkan Ceylan. Cooperative graph neural networks. In *ICML*, 2024.

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

[8] Ben Finkelshtein, İsmail İlkan Ceylan, Michael Bronstein, and Ron Levie. Equivariance everywhere all at once: A recipe for graph foundation models. In *NeurIPS*, 2025.

[9] Maya Bechler-Speicher, Ben Finkelshtein, Fabrizio Frasca, Luis Müller, Jan Tönshoff, Antoine Siraudin, Viktor Zaverkin, Michael M Bronstein, Mathias Niepert, Bryan Perozzi, et al. Position: Graph learning will lose relevance due to poor benchmarks. In *ICML*, 2025.

[10] 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 *ICLR*, 2023.

[11] Jan Tönshoff, Martin Ritzert, Eran Rosenbluth, and Martin Grohe. Where did the gap go? reassessing the long-range graph benchmark. In *LoG*, 2023.

[12] Calvin McCarter. What exactly has tabpfn learned to do? *arXiv*, 2025.

[13] Boris Van Breugel and Mihaela Van Der Schaar. Position: Why tabular foundation models should be a research priority. In *ICML*, 2024.

[14] Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. Tabllm: Few-shot classification of tabular data with large language models. In *International Conference on Artificial Intelligence and Statistics*, pages 5549–5581. PMLR, 2023.

[15] Martin Mráz, Breenda Das, Anshul Gupta, Lennart Purucker, and Frank Hutter. Towards benchmarking foundation models for tabular data with text. *arXiv*, 2025.

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

[17] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *ICML*, 2004.

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

[19] 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*, 2025.

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

[21] Shi Bin Hoo, Samuel Müller, David Salinas, and Frank Hutter. From tables to time: How tabpfn-v2 outperforms specialized time series forecasting models. *arXiv*, 2025.

[22] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *ICLR*, 2022.

[23] Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *JMLR*, 2023.

[24] Semih Cantürk, Renming Liu, Olivier Lapointe-Gagné, Vincent Létourneau, Guy Wolf, Dominique Beaini, and Ladislav Rampášek. Graph positional and structural encoder. In *ICML*, 2024.

[25] Leonardo F.R. Ribeiro, Pedro H.P. Saverese, and Daniel R. Figueiredo. struc2vec: Learning node representations from structural identity. In *KDD*, 2017.

[26] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 2021.

[27] Renchi Yang, Jieming Shi, Xiaokui Xiao, Yin Yang, Sourav S. Bhowmick, and Juncheng Liu. Pane: scalable and effective attributed network embedding. *The VLDB Journal*, 2023.

[28] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *ICLR*, 2020.

[29] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, 2016.

[30] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. In *NeurIPS Datasets and Benchmarks Track*, 2023.

[31] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *ICLR*, 2018.

[32] Péter Mernyei and Cătălina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv*, 2022.

[33] Benedek Rozemberczki and Rik Sarkar. Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models. In *CIKM*, 2020.

[34] 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. In *NeurIPS*, 2021.

[35] Anna Gaulton, Louisa J. Bellis, A. Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, and John P. Overington. ChEMBL: a large-scale bioactivity database for drug discovery. *Nucleic Acids Research*, 2012.

[36] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 1994.

A Additional results

A.1 The Importance of ensemble selection and LinearGNNs

Table 1: Zero-shot accuracy Mean of TabGFM with and without ensemble selection and LinearGNNs.

Method	**Mean Test Accuracy (%)**
TabGFM	73.26 ± 0.14
unweighted ensembling	71.61 ± 0.10
w/o LinearGNNs	71.03 ± 0.31
w/o TabPFN models	69.74 ± 0.24

Setup. We initialize TabGFM with an ensemble comprising ten TabPFN models and eight LinearGNNs, and systematically ablate individual components to assess their contributions. Specifically, we evaluate three alternative configurations: (i) replacing ensemble selection with uniform averaging, (ii) removing all LinearGNNs, and (iii) removing all TabPFN models. For all experiments, we omit PCA preprocessing.

Results. Table 1 reports mean test set accuracies across the 28 datasets described in Section 5. In all cases, the ablated variants substantially underperform the full TabGFM model. While LinearGNNs exhibit lower standalone accuracy, their architectural diversity yields error patterns that are less correlated with those of TabPFNs, enabling ensemble selection to construct stronger ensembles. However, the gains from such diversity are fully realized only when ensemble weights are optimized in a principled manner; naive averaging markedly reduces performance. Finally, we observe that even large and diverse ensembles of LinearGNNs, despite the benefits of ensemble selection, fail to recover the strong tabular inductive biases encoded in transformer-based models such as TabPFN.

A.2 Per-dataset results for Figure 2

Table 2: Zero-shot per-dataset accuracy of TabGFM, TS-Mean and Graphany, with GraphAny and TS-Mean trained on cora.

Dataset	GAT	GraphAny	TS-Mean	TabGFM (PCA)	TabGFM (w/o PCA)
actor	32.59 ± 0.83	27.54 ± 0.20	28.09 ± 0.93	31.18 ± 0.18	31.18 ± 0.18
amazon-ratings	40.63 ± 0.66	42.80 ± 0.09	42.27 ± 1.40	44.34 ± 0.32	44.34 ± 0.32
arxiv	57.93 ± 3.44	58.85 ± 0.03	56.33 ± 2.58	66.70 ± 0.21	66.70 ± 0.21
blogcatalog	78.20 ± 7.23	71.54 ± 3.04	76.30 ± 2.92	79.77 ± 1.06	79.77 ± 1.06
brazil	35.38 ± 4.21	33.84 ± 15.65	39.23 ± 5.70	73.08 ± 4.03	73.08 ± 4.03
chameleon	58.46 ± 6.23	63.64 ± 1.48	60.83 ± 5.41	72.94 ± 1.13	72.94 ± 1.13
citeseer	63.92 ± 0.84	68.88 ± 0.10	68.66 ± 0.19	68.08 ± 0.61	68.08 ± 0.61
co-cs	82.28 ± 0.86	90.06 ± 0.80	90.92 ± 0.47	90.49 ± 0.50	90.97 ± 0.27
co-physics	85.92 ± 1.10	91.85 ± 0.34	92.61 ± 0.61	92.31 ± 0.27	92.33 ± 0.22
computers	70.94 ± 3.40	82.79 ± 1.13	81.37 ± 1.25	84.33 ± 0.33	84.33 ± 0.33
cornell	69.73 ± 2.26	63.24 ± 1.32	68.65 ± 2.42	75.14 ± 2.16	75.14 ± 2.16
deezer	55.22 ± 2.33	51.82 ± 2.49	52.31 ± 2.52	52.99 ± 1.65	52.99 ± 1.65
europe	39.00 ± 4.30	41.25 ± 7.25	35.88 ± 6.91	55.38 ± 2.30	55.38 ± 2.30
full-DBLP	67.34 ± 2.75	71.48 ± 1.44	66.42 ± 3.65	71.92 ± 1.46	71.92 ± 1.46
full-cora	59.95 ± 0.88	51.18 ± 0.78	53.58 ± 0.73	54.56 ± 0.19	58.45 ± 0.24
last-fm-asia	72.65 ± 0.48	81.14 ± 0.42	78.03 ± 1.02	85.47 ± 0.29	85.47 ± 0.29
minesweeper	84.15 ± 0.24	80.46 ± 0.11	80.68 ± 0.38	85.83 ± 0.13	85.83 ± 0.13
photo	80.78 ± 3.59	89.91 ± 0.88	90.18 ± 1.30	89.63 ± 0.48	89.63 ± 0.48
pubmed	75.12 ± 0.89	76.46 ± 0.08	74.98 ± 0.56	78.96 ± 0.43	78.96 ± 0.43
questions	97.13 ± 0.05	97.07 ± 0.03	97.02 ± 0.01	97.14 ± 0.01	97.14 ± 0.01
roman-empire	69.80 ± 4.18	63.34 ± 0.58	66.36 ± 1.02	74.12 ± 0.28	74.12 ± 0.28
squirrel	38.16 ± 1.04	49.74 ± 0.47	41.81 ± 0.80	65.71 ± 0.13	65.71 ± 0.13
texas	81.62 ± 6.45	71.35 ± 2.16	73.51 ± 4.01	81.08 ± 2.09	81.08 ± 2.09
tolokers	78.22 ± 0.37	78.20 ± 0.03	78.12 ± 0.09	82.82 ± 0.15	82.82 ± 0.15
usa	43.03 ± 2.08	43.35 ± 1.62	42.34 ± 2.12	60.50 ± 0.80	60.50 ± 0.80
wiki-attr	68.91 ± 9.50	60.27 ± 3.06	69.89 ± 1.31	70.09 ± 0.92	70.09 ± 0.92
wiki-cs	74.99 ± 0.59	74.11 ± 0.60	74.16 ± 2.07	79.07 ± 0.54	79.07 ± 0.54
wisconsin	73.33 ± 8.27	59.61 ± 5.77	61.18 ± 11.38	83.14 ± 1.33	83.14 ± 1.33
Average (28 graphs)	65.55 ± 2.82	65.56 ± 1.86	65.78 ± 2.28	73.10 ± 0.15	73.26 ± 0.14

Setup. We adopt the experimental setup described in Section 5 and Appendix B for both TabGFM and all baselines. We apply PCA preprocessing for all baselines on the three affected datasets (see

Table 3: Zero-shot per-dataset accuracy of TS-Mean and TabGFM across 20 datasets, with T-Mean trained on the remaining 9 datasets.

Dataset	TS-Mean	TabGFM (PCA)	TabGFM (w/o PCA)
amazon-ratings	42.20 ± 0.72	44.34 ± 0.32	44.34 ± 0.32
arxiv	56.13 ± 1.67	66.70 ± 0.21	66.70 ± 0.21
blogcatalog	77.90 ± 3.80	79.77 ± 1.06	79.77 ± 1.06
brazil	40.77 ± 14.80	73.08 ± 4.03	73.08 ± 4.03
chameleon	56.97 ± 3.68	72.94 ± 1.13	72.94 ± 1.13
citeseer	68.14 ± 0.30	68.08 ± 0.61	68.08 ± 0.61
co-cs	91.36 ± 0.36	90.49 ± 0.50	90.97 ± 0.27
co-physics	92.80 ± 0.54	92.31 ± 0.27	92.33 ± 0.22
cornell	74.59 ± 4.91	75.14 ± 2.16	75.14 ± 2.16
deezer	51.88 ± 2.85	52.99 ± 1.65	52.99 ± 1.65
full-DBLP	66.64 ± 3.55	71.92 ± 1.46	71.92 ± 1.46
full-cora	53.20 ± 1.70	54.56 ± 0.19	58.45 ± 0.24
last-fm-asia	78.07 ± 0.76	85.47 ± 0.29	85.47 ± 0.29
minesweeper	80.05 ± 0.05	85.83 ± 0.13	85.83 ± 0.13
pubmed	77.82 ± 0.56	78.96 ± 0.43	78.96 ± 0.43
questions	97.03 ± 0.02	97.14 ± 0.01	97.14 ± 0.01
squirrel	37.35 ± 1.62	65.71 ± 0.13	65.71 ± 0.13
wiki-attr	73.71 ± 1.61	70.09 ± 0.92	70.09 ± 0.92
wiki-cs	74.31 ± 0.90	79.07 ± 0.54	79.07 ± 0.54
wisconsin	80.47 ± 6.04	83.14 ± 1.33	83.14 ± 1.33
Average (20 graphs)	68.57 ± 2.52	74.39 ± 0.28	74.61 ± 0.26

Appendix B). All baselines are trained on cora, unless stated otherwise. We report results for TabGFM both with and without PCA preprocessing.

Results. Table 2 presents per-dataset accuracies alongside overall mean performance. The overall mean accuracies are also displayed in Figure 2, which reports results for TabGFM (PCA). TabGFM (PCA) achieves the best performance on 20 of the 28 datasets, demonstrating consistent strength across diverse real-world tasks. Omitting PCA preprocessing slightly improves results on all affected datasets, with the most significant gain observed on full-cora.

We also report per-dataset results of TS-Mean trained on nine datasets (cora, texas, tolokers, photo, roman-empire, usa, actor, computers, europe) and TabGFM (with and without PCA preprocessing) for the remaining 20 datasets in Table 3. As TabGFM does not train on graph data, per-dataset results do not change between Table 2 and Table 3 and are provided for convenience. Although the performance of TS-Mean improves with additional training datasets, it remains significantly below TabGFM, outperforming it only on four out of twenty remaining datasets.

A.3 Per-dataset results for Figure 3

Setup. We follow the setup described in Section 5.2, which extends the general experiment setup from Section 5 and Appendix B. In this experiment, we vary the number of subsampled tables and corresponding TabPFN models $B \in \{1, \ldots, 10\}$. All results are reported without PCA preprocessing.

Results. Per-dataset and overall mean accuracies for selected values of $B \in \{0, 1, 4, 7, 10\}$ are reported in Table 4, while Figure 3 visualizes the overall trend together with standard errors. Across datasets, accuracy generally increases with larger B. The most substantial improvements occur when moving from $B = 0$ (only LinearGNN components) to $B = 1$, where TabPFN models are first introduced. For datasets with a large number of classes, this threshold can shift to higher B (see Appendix B). Beyond this point, performance gains remain consistent, though less pronounced, as B increases further.

Table 4: Zero-shot per-dataset accuracy of TabGFM, with varying TabPFN models B.

Dataset	B=0	B=1	B=4	B=7	B=10
actor	31.24 ± 0.25	30.84 ± 0.15	31.16 ± 0.23	31.22 ± 0.12	31.18 ± 0.18
amazon-ratings	43.61 ± 0.28	43.90 ± 0.15	44.16 ± 0.25	44.29 ± 0.33	44.34 ± 0.32
arxiv	58.43 ± 0.05	58.43 ± 0.05	58.43 ± 0.05	65.72 ± 0.27	66.70 ± 0.21
blogcatalog	79.13 ± 1.01	80.25 ± 0.99	79.31 ± 0.94	80.09 ± 1.16	79.77 ± 1.06
brazil	53.08 ± 3.73	71.54 ± 4.65	76.92 ± 4.55	73.85 ± 1.88	73.08 ± 4.03
chameleon	70.66 ± 1.23	72.72 ± 0.85	72.11 ± 1.05	73.20 ± 1.16	72.94 ± 1.13
citeseer	64.68 ± 1.03	65.38 ± 1.23	67.10 ± 0.32	67.12 ± 0.75	68.08 ± 0.61
co-cs	90.90 ± 0.14	90.90 ± 0.14	90.97 ± 0.19	91.04 ± 0.20	90.97 ± 0.27
co-physics	92.27 ± 0.27	92.14 ± 0.26	92.14 ± 0.28	92.15 ± 0.20	92.33 ± 0.22
computers	82.18 ± 0.17	84.03 ± 0.68	84.34 ± 0.31	83.94 ± 0.48	84.33 ± 0.33
cornell	75.14 ± 1.58	74.05 ± 1.83	75.68 ± 2.09	76.76 ± 1.38	75.14 ± 2.16
deezer	52.08 ± 1.28	51.72 ± 1.39	51.68 ± 1.40	51.67 ± 1.61	52.99 ± 1.65
europe	43.25 ± 2.72	55.88 ± 1.49	54.50 ± 2.53	53.50 ± 3.12	55.38 ± 2.30
full-DBLP	70.31 ± 1.08	72.43 ± 0.79	73.40 ± 1.09	72.17 ± 1.03	71.92 ± 1.46
full-cora	57.18 ± 0.25	57.18 ± 0.25	57.18 ± 0.25	58.44 ± 0.17	58.45 ± 0.24
last-fm-asia	84.68 ± 0.28	84.68 ± 0.28	85.12 ± 0.29	85.61 ± 0.31	85.47 ± 0.29
minesweeper	82.11 ± 0.17	85.17 ± 0.15	85.74 ± 0.10	85.75 ± 0.09	85.83 ± 0.13
photo	90.78 ± 0.38	90.51 ± 0.40	90.80 ± 0.62	90.41 ± 0.60	89.63 ± 0.48
pubmed	75.66 ± 0.71	77.68 ± 0.67	76.64 ± 1.73	76.52 ± 1.77	78.96 ± 0.43
questions	97.04 ± 0.03	97.03 ± 0.03	97.05 ± 0.03	97.10 ± 0.02	97.14 ± 0.01
roman-empire	67.59 ± 0.17	67.59 ± 0.17	73.65 ± 0.17	73.86 ± 0.23	74.12 ± 0.28
squirrel	56.64 ± 0.47	64.94 ± 0.35	65.46 ± 0.41	65.57 ± 0.14	65.71 ± 0.13
texas	82.70 ± 2.78	82.16 ± 2.36	80.00 ± 1.08	81.08 ± 2.09	81.08 ± 2.09
tolokers	79.03 ± 0.10	82.21 ± 0.22	82.82 ± 0.16	82.84 ± 0.15	82.82 ± 0.15
usa	50.45 ± 2.49	61.44 ± 0.76	59.78 ± 0.92	60.54 ± 0.86	60.50 ± 0.80
wiki-attr	66.18 ± 1.88	66.18 ± 1.88	69.66 ± 0.75	70.26 ± 0.98	70.09 ± 0.92
wiki-cs	77.62 ± 0.26	78.96 ± 0.60	79.01 ± 0.64	79.08 ± 0.53	79.07 ± 0.54
wisconsin	78.04 ± 2.66	81.57 ± 2.29	85.10 ± 0.48	81.18 ± 1.00	83.14 ± 1.33
Average	69.74 ± 0.24	72.20 ± 0.18	72.85 ± 0.23	73.03 ± 0.09	73.26 ± 0.14

B Implementation details

All experiments can be run using a single NVIDIA L40S GPU. Reported results are averaged across five runs with random seeds {0, 1, 2, 3, 4}. The source code to reproduce our experiments is available at `https://github.com/ahayler/tag`.

TabGFM node-level embeddings. For the GPSE embeddings [24] (see section 4.1), we use the checkpoint trained on ChEML [35]. For RandomWalkPE [22] and LaplacianEigenvectorPE [23], we pick $k = 20$. Due to computational constraints, we only compute RandomWalkPE for graphs with fewer than or equal to 5000 nodes. TabGFM filters out the LinearGNN based on RandomWalkPE for datasets with more than 5000 nodes.

TabPFN. For all our experiments, we use the newest version of TabPFN at the time of writing. For each TabPFN model, we subsample 2500 random labelled rows and 400 columns. To account for the imbalance in the number of feature-encoding columns vs. the number of structure-encoding columns, 300 of 400 columns are sampled from feature encodings and 100 are sampled from structure encodings.

Since TabPFN natively supports at most ten classes, we adopt the error-correcting output code (ECOC) strategy [36] suggested by Hollmann et al. [19], which splits tasks with more than ten classes into B subtasks of at most ten classes each. Our subsampling strategy is applied independently to each subtask and aggregated outputs form one predictor in the ensemble selection. Given C classes, the ECOC-strategy generally needs at least $\lceil C/9 \rceil$ subproblems to ensure coverage; we do not apply TabPFN models on the dataset with $B < \lceil C/9 \rceil$.

Table 5: Statistics of the 28 node classification datasets.

Dataset	#Nodes	#Edges	#Feats	#Classes	Train/Val/Test (%)
actor	7,600	30,019	932	5	48.0/32.0/20.0
amazon-ratings	24,492	186,100	300	5	50.0/25.0/25.0
arxiv	169,343	1,166,243	128	40	53.7/17.6/28.7
blogcatalog	5,196	343,486	8,189	6	2.3/48.8/48.8
brazil	131	1,074	131	4	61.1/19.1/19.8
chameleon	2,277	36,101	2,325	5	48.0/32.0/20.0
citeseer	3,327	9,104	3,703	6	3.6/15.0/30.1
co-cs	18,333	163,788	6,805	15	1.6/49.2/49.2
co-physics	34,493	495,924	8,415	5	0.3/49.9/49.9
computers	13,752	491,722	767	10	1.5/49.3/49.3
cora	2,708	10,556	1,433	7	5.2/18.5/36.9
cornell	183	554	1,703	5	47.5/32.2/20.2
deezer	28,281	185,504	128	2	0.1/49.9/49.9
europe	399	5,995	399	4	20.1/39.8/40.1
full-DBLP	17,716	105,734	1,639	4	0.5/49.8/49.8
full-cora	19,793	126,842	8,710	70	7.1/46.5/46.5
last-fm-asia	7,624	55,612	128	18	4.7/47.6/47.6
minesweeper	10,000	78,804	7	2	50.0/25.0/25.0
photo	7,650	238,162	745	8	2.1/49.0/49.0
pubmed	19,717	88,648	500	3	0.3/ 2.5/ 5.1
questions	48,921	307,080	301	2	50.0/25.0/25.0
roman-empire	22,662	65,854	300	18	50.0/25.0/25.0
squirrel	5,201	217,073	2,089	5	48.0/32.0/20.0
texas	183	558	1,703	5	47.5/31.7/20.2
tolokers	11,758	1,038,000	10	2	50.0/25.0/25.0
usa	1,190	13,599	1,190	4	6.7/46.6/46.6
wiki	2,405	17,981	4,973	17	14.1/42.9/43.0
wiki-cs	11,701	431,206	300	10	5.0/15.1/49.9
wisconsin	251	900	1,703	5	47.8/31.9/20.3

LinearGNNs. The eight additional LinearGNNs in are each based on one of the eight node-level encodings presented in Section 4.1. We normalize the outputs $\mathbf{l} = (l_c)_{c \in C}$ of the LinearGNNs before ensembling using the following proportional scaling, which maps the smallest logit to 0

$$\boldsymbol{l}'_c = \boldsymbol{l}_c - \min_i \boldsymbol{l}_i + \epsilon \quad \forall c \in [C],$$
$$\boldsymbol{p} = \frac{\boldsymbol{l}'}{\sum_i \boldsymbol{l}'_i}.$$

This ensures that ensemble models with low weights cannot overrule other ensemble members by predicting unbounded logits with high amplitude.

Ensemble selection. Our implementation of ensemble selection [17] samples models with replacement, breaks ties (during the greedy selection) randomly and implements early stopping, i.e. it picks the first model configuration in the history that achieved the best possible (accuracy) score on the held-out predictions. We used k-fold cross-validation to generate held-out predictions for all labeled nodes L. For TabPFN-based models, we use two folds and five folds for LinearGNNs.

C Dataset statistics

Dataset statistics for all 28 datasets used in Section 5 are presented in Table 5.

D Impact statement

This paper presents work aimed at advancing the field of graph machine learning through the development of a graph foundation model for node classification. Potential applications of such models include social network analysis, recommendation systems, fraud detection, and knowledge graph completion. These applications can have significant societal impacts, ranging from improved information retrieval and decision support to risks such as bias amplification, privacy concerns, or the reinforcement of misinformation. However, we do not identify any specific, immediate concerns requiring special attention in the context of this work.

Submission of papers to NeurIPS 2025

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We propose a novel graph foundation model based on tabular foundation models and validate our claim empirically. In addition to the results in the main paper, we present detailed results in the appendix.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We acknowledge several limitations in this work, including the restriction of TabGFM to the node classification setting and its dependence on predefined feature and structural encoders to capture relevant graph information.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper does not include any theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide all information necessary to reproduce the results presented in the paper, including details on the model architecture, datasets, hardware setup, and other relevant materials.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide all code required to reproduce our experiments in a GitHub repository, together with detailed documentation of the experimental setup in both the main paper and the appendix.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide an overview of the experimental setup sufficient to appreciate the results in the main paper and further experimental details in the appendix. The appropriate sections of the appendix are clearly referred to in the main paper for easy reference.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Any Figure or Table that includes standard errors contains appropriate labeling. The factor of variability (random seeds) is mentioned in the experiments section and further explained in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report sufficient hardware statistics to replicate all our experiments in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work adheres to the NeurIPS Code of Ethics, including anonymity and data-use considerations.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: Yes. The paper discusses both positive and negative societal impacts in the appendix. However, we do not identify any specific, immediate concerns requiring special attention in the context of this work.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We do not believe that our work requires additional safeguards. The used model checkpoints and datasets are already publicly accessible.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All third-party assets are cited with versions and terms, and licenses are respected.

Guidelines:

- The answer NA means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does introduce any new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: This paper does not use LLMs in its method.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.