GRAFT: IN-CONTEXT CLASSIFICATION AND REGRESSION AT NODE, EDGE AND GRAPH LEVEL

Anonymous authors
Paper under double-blind review

ABSTRACT

A single pretrained model can now match graph neural networks trained on the same labels, at the node, edge and graph levels and for classes or values. We show this with Graft, a 27M-parameter in-context model and the first to answer all six graph task types. Given labeled examples from a graph, it predicts every query with no gradient step or dataset-specific head. Graft accepts arbitrary feature columns through an exchangeable tokenizer, turns every target (a node, a pair of nodes, a whole graph) into a row over the nodes it spans, and treats classification as a special case of regression. Pretraining combines sixteen families of random graphs with random labeling functions and 59 real datasets. Across 49 dataset–task pairs, frozen Graft matches or beats the better of GCN and GIN trained on the labels in its prompt on 45 of them at a standard label budget per task, including every link-prediction and every regression benchmark, and it has the best average rank among all same-label methods on five of the six task types. The gap is widest in regression, where networks trained on few labels often fall below the mean predictor. Graft stays stable on prompts sixteen times longer than any it saw in pretraining. It also generalizes beyond its pretraining data: on dataset families withheld entirely, classification is nearly unchanged from the model that saw them, and fine-tuning on the dataset's labels recovers much of the regression loss.

1 INTRODUCTION

A citation graph whose nodes carry 500 word counts and a molecule whose atoms carry 9 descriptors do not fit the input layer of one conventional network, and asking of a graph which class a node belongs to, whether two nodes are linked and what a whole graph's property is takes three different output heads. Graph learning therefore still trains one network per dataset, per prediction level and per task type. The input side is now being solved from several directions: analytical predictors that ignore feature identity (Zhao et al., 2025), statistics over heterogeneous attributes (Shen et al., 2025), feature-permutation-invariant architectures (Finkelshtein et al., 2025) and shared input operators (Eliasof et al., 2026); and in-context models now read graphs with new features from a prompt of labeled examples (Eremeev et al., 2025b; Zhuo & Luo, 2026). The output side is still mostly open. Most in-context graph models answer only node classification; OFA, UniGraph, GIT and GILT reach link and graph classification (Liu et al., 2024a; He et al., 2025; Wang et al., 2025c; Ma et al., 2025); and none of them predicts a continuous target beyond the node level (Table 1). Graph-level regression in context has so far meant leaving the graph for a SMILES string or a fixed molecular descriptor (Liu et al., 2024b; Ben Hicham et al., 2026).

We introduce Graft, a single pretrained network that, given labeled examples from a graph, predicts node, edge or graph targets, classes or values, with no gradient step and no per-dataset head (Figure 1). It is the first model that reads the graph to answer all six graph task types in context, and it does so at the level of networks trained on the same labels: on 49 dataset–task pairs from citation, co-purchase, web and social graphs, trust and rating networks, road networks, molecules and images, it matches or beats the better of GCN and GIN on 45 (Table 2, Figure 3).

Three choices make this possible. Binding each feature value to a random code for its column removes the input layer's dependence on a dataset. Building every target as a row over the nodes it spans removes the need for a head per level, and treating classification as a special case of regression removes the head per task type. Pretraining on sixteen families of random graphs with random

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.

Table 1: Which pretrained graph models answer which task at inference, without gradient updates on the target, per each model's own paper or from running it, ordered by tasks covered. All read labeled examples of the target in context with no gradient step, except § zero-shot, with no labeled examples. † Text-attributed graphs only. ∗ Evaluated mainly on text-attributed graphs. ‡ Tabular; run on node features without the graph.

	node		edge		graph		tasks
model	class	value	class	value	class	value	covered
GraphAny (Zhao et al., 2025)	✓	–	–	–	–	–	1
NodePFN (Choi et al., 2026)	✓	–	–	–	–	–	1
TFMLinker (Liao et al., 2026)	–	–	✓	–	–	–	1
TabPFN v2‡ (Hollmann et al., 2025)	✓	✓	–	–	–	–	2
GraphPFN (Eremeev et al., 2025b)	✓	✓	–	–	–	–	2
G2T-FM (Eremeev et al., 2025a)	✓	✓	–	–	–	–	2
PRODIGY (Huang et al., 2023)	✓	–	✓	–	–	–	2
MF-GIA (Zhuo & Luo, 2026)	✓	–	✓	–	–	–	2
AnyGraph§ (Xia & Huang, 2026)	✓	–	✓	–	–	–	2
OpenGraph§ (Xia et al., 2024)	✓	–	✓	–	–	–	2
OFA† (Liu et al., 2024a)	✓	–	✓	–	✓	–	3
UniGraph† (He et al., 2025)	✓	–	✓	–	✓	–	3
GIT∗ (Wang et al., 2025c)	✓	–	✓	–	✓	–	3
GILT (Ma et al., 2025)	✓	–	✓	–	✓	–	3
Graft (ours)	✓	✓	✓	✓	✓	✓	**6**

labeling functions, mixed with real graphs, then yields a model that transfers to dataset families withheld from pretraining and to prompts sixteen times longer than any it was trained on.

Why all six. A model evaluated at one level leaves open whether it learned about graphs or about that level's task. And a classification result in context can come from matching a query to its nearest class prototype; a regression result has no classes to match, so with one head for both, regression is the harder test of the same mechanism. It is also where Graft's margins are largest: at node, edge and graph regression, GNNs trained on the same labels often fall below the mean predictor, and Graft is ahead on all 14 datasets (Table 2).

Contributions.

1. We introduce Graft, a single pretrained model that predicts node, edge and graph targets, classes or values, on graphs with arbitrary feature columns. It works from the labeled examples in its prompt alone, with no gradient step or dataset-specific head, and it is the first in-context model that reads the graph itself to answer all six graph task types and to regress edge- and graph-level targets.
2. We develop an exchangeable tokenizer that binds each feature value to a random code for its column, so graphs with any feature columns enter one network; a construction that builds every target, whether a node, a node pair or a whole graph, as a row over the nodes it spans; and one in-context head that reads all rows and returns classes and values by combining in-context ridge regression, class prototypes and label propagation with a transformer over the rows (Section 3).
3. We show that on 49 dataset–task pairs, frozen Graft matches or beats the better of GCN and GIN trained on its prompt's labels on 45, including every link-prediction and regression benchmark at a standard budget per task, and has the best average rank among same-label methods on five of the six task types (Section 4.2).
4. By withholding each dataset family from one of five pretraining runs, we find that classification transfers almost unchanged to families never seen in pretraining, and that fine-tuning on the dataset's labels recovers much of the regression loss. Without any fine-tuning, Graft also stays stable on prompts sixteen times longer than any seen in pretraining (Sections 4.3 and 4.4).

2 RELATED WORK

Prior-fitted networks and synthetic priors. Prior-fitted networks train a transformer to approximate the posterior predictive of a hand-designed prior over tasks, so that inference is a single forward pass over a labeled context (Müller et al., 2022). On tables the line runs from TabPFN (Hollmann et al., 2023) to TabPFN v2 (Hollmann et al., 2025), whose prior is a structural causal model over attributes. Task diversity, not realism, controls whether in-context learning emerges (Raventós et al., 2023), as fractal images pretrain vision models (Kataoka et al., 2020; Baradad et al., 2021) and synthetic reasoning tasks pretrain language models (Wu et al., 2021); most of our pretraining prompts are synthetic for that reason. Fine-tuning a prior-fitted network re-aligns the query-key similarity of its last layer (Rubachev et al., 2025). Transformers trained on families of functions learn linear regression in context (Garg et al., 2022) and can implement ridge regression or a gradient step on it (Akyürek et al., 2023; von Oswald et al., 2023; Bai et al., 2023); Graft injects that estimator explicitly. TabICL gives each column an identifier so that identically distributed columns stay apart (Qu et al., 2025), as our tokenizer's random column codes do.

Graph foundation models on real data. Graph foundation models (Mao et al., 2024; Wang et al., 2025b) split by what they are pretrained on. Real-data models learn from collections of benchmark graphs and are scored on held-out graphs or datasets (Huang et al., 2023; Liu et al., 2024a; Xia & Huang, 2026). AnyGraph and OpenGraph (Xia & Huang, 2026; Xia et al., 2024) are link predictors with a node head, and we run both on our edge splits (Section 4.1). Table 1 gives each model's task coverage: OFA, UniGraph and GIT (Liu et al., 2024a; He et al., 2025; Wang et al., 2025c) work mainly on text-attributed graphs whose features come from a language model, while GILT (Ma et al., 2025) and MF-GIA (Zhuo & Luo, 2026) read raw or pre-vectorised features. UniLP and ULTRA predict links (Dong et al., 2025; Galkin et al., 2024), KG-ICL and KGPFN do so in context on knowledge graphs (Cui et al., 2024; Gao et al., 2026), and prompt tuning unifies the three levels but trains on each target (Sun et al., 2023). Input-side methods (Section 1; also FUG, Zhao et al., 2024) are complementary: our column tokenizer is one more route, and our contribution is on the output side.

Prior-fitted and tabular graph models. Synthetic-prior graph models are recent and node-only. NodePFN (Choi et al., 2026) fits a prior-data network to synthetic graphs for node classification, and GraphPFN (Eremeev et al., 2025b) combines a multi-level block-model and preferential-attachment prior with graph-aware structural causal models on top of a tabular foundation model, leaving link prediction and graph-level tasks to future work. A fair evaluation of graph foundation models for node property prediction (Platonov et al., 2026) finds that only the prior-fitted ones beat tuned GNNs there. A parallel line tabularises the graph and hands propagated and structural features to a pretrained tabular in-context model, for nodes (Eremeev et al., 2025a; Hayler et al., 2025) and for links (Liao et al., 2026). GraphAny (Zhao et al., 2025) reaches the same place without a sequence model: analytical linear-GNN predictors, combined by a learned attention, classify nodes on graphs with new feature and label spaces without gradient steps. Graft carries closed forms of that kind inside it and learns what to add to them. On relational databases, KumoRFM and the Relational Transformer classify and regress entities in context (Fey et al., 2025; Ranjan et al., 2026; Hudovernik et al., 2026).

Regression beyond nodes. Graph-level regression in context has so far left the graph. Language models read SMILES strings (Liu et al., 2024b; Busch et al., 2026), and tabular in-context models read fixed molecular descriptors or embeddings from a separately trained encoder (Ben Hicham et al., 2026; Li et al., 2025). LangGFM covers all six task types by fine-tuning a language model on each benchmark (Lin et al., 2024), UniGTE regresses graphs zero-shot from graph-text input (Wang et al., 2025a), and GraphBFF regresses at node and edge level by probing frozen embeddings (Bechler-Speicher et al., 2026). To our knowledge no in-context model that reads the graph regresses at more than one level, and none answers all six task types.

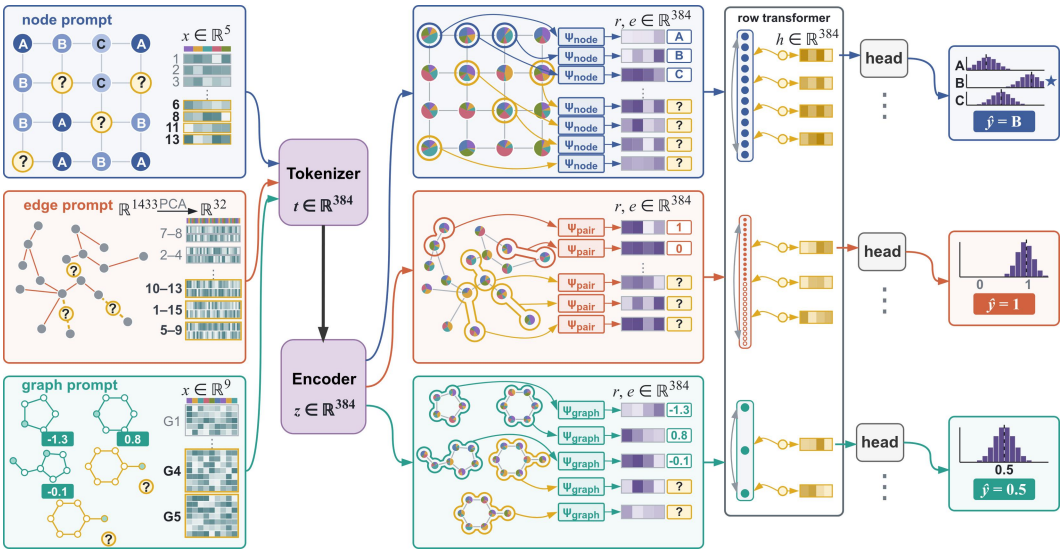

Figure 1: One set of weights for node, edge and graph questions. Three single-level prompts: node classes on a lattice, links on Cora, and log solubility of ESOL molecules; coloured marks are labeled context and question marks are queries. The tokenizer τ, the encoder, the row transformer and the head are shared. Only the projector differs: a target becomes one row through ψ_{node} on a node, ψ_{pair} on a node pair or ψ_{graph} on a whole graph (outlined). Context rows carry their standardised label $\tilde{y}$, queries a learned token, and the head returns a prediction for each query.

3 Method

3.1 Prompts at three levels

A prompt is a set of rows about one or more graphs. A row is a node, a node pair, or a whole graph; a context row carries its label (a class or a continuous target), a query row does not. A link prompt's context holds visible edges as positives and as many sampled non-edges as negatives. The model reads every row of the prompt in one forward pass and returns a prediction for each query row (one pass per class for a multiclass target, Section 3.3). Nothing in the format is specific to a dataset: features are read column by column with no fixed schema, labels are read as ± 1 one-vs-rest targets or as z-scored values, and the number of context rows is whatever the prompt contains, from a handful to the whole training split.

3.2 Tokenizer and encoder

Node features enter through an exchangeable tokenizer: a shared value encoder maps each feature value to a d-dimensional vector, a per-column identity token tells columns apart within a graph, and a presence term distinguishes a feature that is zero from one that is missing, so a graph with any number of feature columns in any order can be read and unseen feature spaces transfer; the column contributions are averaged and the number of observed columns is encoded separately. Wide real feature matrices are first reduced to 32 columns by PCA, fitted without labels and applied to the GNN and closed-form baselines alike (Appendix C). Each node also carries a random identifier so that the encoder can tell structurally identical nodes apart. No hand-built structural descriptor is an input: structure reaches the model only through the graph itself, as a log-degree embedding before the first layer and the neighbourhood term in every layer, and, for graph rows, the log of the graph's size. Tokens and identifier enter the encoder through a *linear* projection. The encoder is a stack of eight linear-attention layers of width 384: attention uses the feature map $\phi(x) = \mathrm{elu}(x) + 1$ so that its cost is linear in the number of nodes, and the graph's edges enter through an additive neighbourhood term on the values (their mean over a node's neighbours, plus a degree-scaled copy) that runs beside the global aggregate.

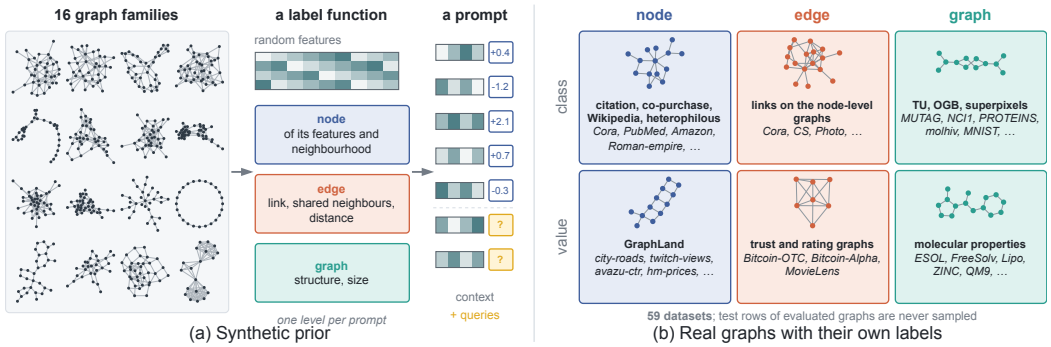

Figure 2: Where the pretraining prompts come from. (a) A synthetic prompt: a graph from one of sixteen random-graph families, random features, and a label function drawn for the prompt at one level. (b) A real prompt: a real graph with its own labels, from 59 datasets covering all six task types.

3.3 The in-context head

The head turns node embeddings into rows and rows into predictions. A row's representation is the pooled embedding of its object (a node's embedding, a pair's pooled and differenced embeddings, a graph's pooled summary of its nodes, their mean, standard deviation and maximum with the log of the graph's size) plus, for a context row, an embedding of its label, or, for a query row, a learned query token; node rows also carry a skip from the node's own pre-encoder token (pair rows: the product and absolute difference of their two tokens). Two computations then run on the rows of a prompt: closed-form estimators that are strong on their own on many graph benchmarks, and a transformer that learns what to add to them. The first is closed-form: a prototype term compares each query row to the context rows of each class, and a ridge term solves a ridge regression from the context rows to their labels in a learned 16-dimensional projection of the rows, with the raw node features and their 2- and 4-hop propagations as extra columns, and reads the query rows off it; the context labels are also propagated over the visible graph for eight steps (mean over neighbours and self, context labels clamped) and the propagated label enters each row as a feature. The second is a six-layer transformer over the rows with block-diagonal attention per prompt, asymmetric (query rows attend to context rows and to themselves; context rows never attend to query rows, and no query attends to another) and, when a prompt carries several levels, masked by level. In the model we report, the two closed-form estimates are injected into each row as two scalar features before the transformer, and the prediction is read off the transformer's output: the target is z-scored on the context rows, the head emits a distribution over 64 fixed quantile buckets of it (trained with cross-entropy; outer buckets half-open), and the prediction is that distribution's expectation plus a learned residual, mapped back through the context's mean and scale. There is no class dimension anywhere in the head: a C-class problem is answered by C one-vs-rest prompts whose context labels are ± 1, z-scored on the context, and the prediction is the argmax of the C scalars; each of the C passes answers every query. In pretraining a classification row task is likewise a continuous target thresholded at a random quantile, so the model is never shown a multiclass target.

3.4 The synthetic prior

Pretraining prompts are drawn from a prior over graphs and label functions (Figure 2a): sixteen random-graph families (block models, geometric and power-law graphs, graphons, planted motifs, and a sparse corner of trees, rings, lattices and clique unions; Appendix A.2), with 10 to 600 nodes, average degree between 1.8 and 80 and 1 to 128 feature columns. Each prompt fixes one labeling function and applies it to a few graphs; a node- or edge-level prompt draws its rows from three such graphs. Node label functions are linear and nonlinear functions of a node's features and propagated neighbourhood; edge label functions are link existence (query positives removed from the visible graph), common neighbours, capped shortest-path distance and a pair readout; graph label functions are structural and size targets. A graph-level prompt holds 4 to 56 context graphs; a node- or edge-level prompt up to 4096 context rows and 32 queries; every training prompt holds rows of a single level, and a fraction of feature columns is masked.

3.5 Real label functions in pretraining, with holdouts

A fraction of the pretraining steps draw their prompt from a real graph with its real labels (Figure 2b): the citation, co-purchase, co-author and web graphs at node and edge level, the TU sets at graph level, GraphLand balls and rated-pair graphs for node and edge regression, and molecular sets for graph regression. Every graph that is evaluated in this paper has a per-dataset holdout file that fixes, once, the test nodes, the test edges (removed from every visible graph) or the test graphs; the trainer never samples them, and the batteries draw their queries from exactly them. The main results therefore measure prediction on unseen rows of graphs whose other rows were available in pretraining. Section 4.3 measures what having seen a dataset's family is worth. Before pretraining we also remove every graph that appears, with the same task, in any benchmark's held-out split (47% of ESOL's test molecules are in AQSOL with the same label; Appendix C lists every exclusion).

3.6 Training

The reported model has 27.0M parameters and is trained for 230k steps (constant learning rate 3×10^{-4} to 200k, then a cosine anneal to 3×10^{-5}) with one prompt per step whose level is drawn as graph, node or edge with fixed probabilities, on one V100 GPU. The loss on the query rows is the same at every level and for both task types (Appendix A.1).

4 Experiments

We ask whether one frozen model competes with networks trained on the same labels at all six task types (Section 4.2), whether that survives withholding a whole dataset family from pretraining (Section 4.3), and whether it uses prompts longer than any seen in pretraining (Section 4.4).

4.1 Setup

Graft is used frozen. Contexts are drawn from each dataset's training split and queries from its held-out test split (Section 3.5), five draws per cell, and every Graft number is also a mean over three pretraining seeds.

Baselines. GCN (Kipf & Welling, 2017) or GIN (Xu et al., 2019), whichever a split of the K labels in Graft's prompt selects, trained on exactly those labels ("same labels"; Appendix F), and the same trained on the whole training split and selected on its validation split ("every label"); closed-form ridge and logistic regressions over raw and propagated features, which use no pretrained weights; and the in-context and foundation models with public code that run on our splits: GraphAny (Zhao et al., 2025), GraphPFN (Eremeev et al., 2025b), G2T-FM (Eremeev et al., 2025a), TabPFN v2 without the graph (Hollmann et al., 2025), and AnyGraph and OpenGraph zero-shot (Xia & Huang, 2026; Xia et al., 2024). **Metrics** follow each dataset's convention: accuracy for multiclass and ROC-AUC for binary classification, AUC against uniformly sampled non-edges for link prediction, the common protocol (Li et al. (2023) discuss harder negatives), and R^2 for regression at every level. Across datasets we report average rank and, in figures, a skill scale on which 0 is the trivial predictor and 1 a perfect one; within a dataset it ranks methods as the native metric does (Appendix B).

4.2 One model, six task types

Table 2 and Figure 3 carry the main result: one frozen model answers all six task types, and on 45 of 49 dataset–task pairs it matches or beats the better of GCN and GIN trained on the labels it reads, at a standard budget per task type (20 labels per class, 128 labeled nodes or pairs, 64 labeled graphs for classification and 32 for regression; marked in Table 2). Among the methods that read the same labels it has the best average rank on five of the six task types; the exception is node regression, and the zero-shot AnyGraph and OpenGraph are compared separately (Table 3).

Regression. The margin is widest here. At node, edge and graph level, the same-label GNN and a closed-form ridge over the features reach a fraction of Graft's score or fall below the mean predictor, while Graft is ahead of both on every node- and graph-regression dataset (6 of 6, 5 of 5), on targets from page views and click rates to solubility and trust ratings. No other in-context model we could

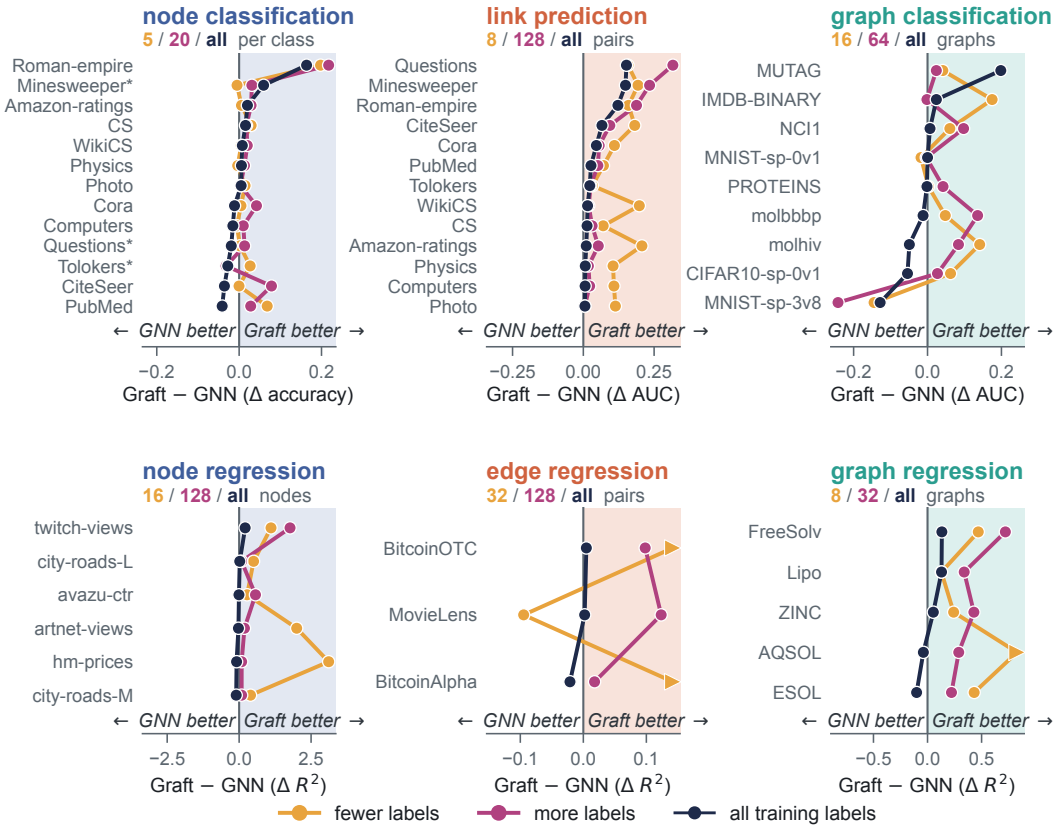

Figure 3: Graft against a GNN trained on the same labels, dataset by dataset. Each point is Graft's score minus that of GCN or GIN, whichever a split of the labels in Graft's prompt selects, in the dataset's own metric (accuracy; ROC-AUC for links, graph classification and the binary node sets marked *; R^2 for regression). One line per context size: two budgets (above each panel) and all training labels. Triangles: clipped at the panel edge. Mean of three pretraining seeds.

Table 2: Number of datasets on which Graft matches or beats the GNN of Figure 3 trained on exactly the K labels in its prompt, at every context size; *all*: every training label, read in one prompt. *Place*: Graft's average rank against the GNNs over the row, with the row's total of datasets won. Every value is in Appendix E.

	Graft $\geq$ GNN trained on the same K labels								place,
labels per class, K =	1	3	5	10	20			all	total
node classification	5/13	13/13	9/13	13/13	**12/13**			7/13	1st (59/78)
labeled nodes, pairs or graphs, K =	8	16	32	64	128	256	512	all	
node regression	5/6	6/6	6/6	6/6	**6/6**	5/6	5/6	2/6	1st (41/48)
link prediction	13/13	13/13	13/13	13/13	**13/13**	13/13	13/13	13/13	1st (104/104)
edge regression	3/3	3/3	2/3	2/3	**3/3**	3/3	2/3	2/3	1st (20/24)
graph classification	8/9	8/9	7/9	**6/9**	8/9			4/9	1st (41/54)
graph regression	5/5	5/5	**5/5**	5/5	4/5			3/5	1st (27/30)

run regresses beyond nodes from the graph. At the node level GraphPFN, G2T-FM and TabPFN v2, which read the raw node features, are close to Graft at the smallest budgets, and GraphPFN leads from 32 labeled nodes (Table 3).

Link prediction. From 128 labeled pairs Graft beats the same-label GNN on all 13 graphs, and it stays ahead at every context size and with every training label in one prompt. AnyGraph and

Table 3: Graft against the other in-context models, on the task types where one runs: number of datasets on which Graft matches or beats each model given the same K labels, on the sizes and datasets each model fits. AnyGraph and OpenGraph ignore the labeled pairs. GraphPFN, G2T-FM and TabPFN v2 read the raw node features; Graft reads a 32-column projection.

node classification, *labels per class*, K =	1	3	5	10	20		
Graft $\geq$ GraphAny	5/13	2/13	7/13	6/13	7/13		
Graft $\geq$ GraphPFN	12/12	13/13	9/13	8/13	8/13		
Graft $\geq$ G2T-FM	10/13	11/13	8/10	11/13	9/10		
Graft $\geq$ TabPFN v2	10/10	10/11	9/9	11/11	9/9		
node regression, *labeled nodes*, K =	8	16	32	64	128	256	512
Graft $\geq$ GraphPFN	4/6	3/6	2/6	2/6	2/6	1/6	0/6
Graft $\geq$ G2T-FM	4/6	4/6	4/6	3/6	3/6	2/6	0/6
Graft $\geq$ TabPFN v2	5/6	5/6	4/6	3/6	3/6	3/6	3/6
link prediction, *labeled pairs*, K =	8	16	32	64	128	256	512
Graft $\geq$ AnyGraph	3/13	5/13	4/13	4/13	4/13	6/13	8/13
Graft $\geq$ OpenGraph	4/13	4/13	4/13	4/13	5/13	7/13	10/13

OpenGraph, pretrained around link prediction, lead on most graphs up to 128 pairs; at 512 Graft passes them on 8 and 10 of 13 (Table 3).

Node classification. From 20 labels per class Graft has the best average rank, above GraphAny and the same-label GNN, and beats GraphPFN and G2T-FM on most of the graphs they run on (Table 3). TabPFN without the graph is far below, by a median 0.36 normalized score. A fixed low-pass filter and a linear model reach much of the reported accuracy on node benchmarks (Wu et al., 2019; Shchur et al., 2018); our closed-form baseline is that model, it is also what Graft injects into its rows, and Graft is ahead of it on 12 of 13 graphs.

Graph classification. From 32 to 64 labeled graphs Graft has the best average rank at the same label budget and is ahead of the same-label GNN on 6 of 9 datasets, from mutagenicity to superpixel images. A GNN trained on the full split, which sees about 2 to 60 times more labels, is the reference it has not yet reached.

4.3 In pretraining and out of it

The main benchmark holds out test rows; this section holds out whole dataset families. Every pretraining dataset is withheld, together with its family, from exactly one of five fold models, so each dataset can be scored by a model that never saw its family. Families are defined by content rather than name, so near-duplicate and same-task datasets are withheld together (ESOL with AQSOL, the two Peptides sets, Tolokers with its GraphLand version).

Classification. Compared with the models that saw the family, the median change is -0.02 normalized score, and 29 of 35 datasets move by less than 0.05 (Figure 4a): the model reads a labeled prompt rather than recognising datasets.

Regression. Domain-specific targets (trust ratings, solubility, road attributes) benefit most from having seen their domain, as expected of a prior: a value's scale and meaning are domain knowledge. The cost is recoverable (Figure 4b). Fine-tuning the model that never saw the family on the target's own training labels, with early stopping on a validation split, improves it on 165 of 189 dataset–context cells, lifts graph regression from 0.10 to 0.44 and edge regression from -0.39 to 0.13, and brings link prediction and graph classification above a GNN trained on the full training split on 9 of 13 link graphs and 5 of 6 graph-classification sets (Appendix H).

4.4 Longer prompts

Graft is pretrained on prompts of at most 4096 context rows. Its memory grows linearly with the number of rows, so it can be prompted far beyond that at inference, and it stays stable up to 65,536 rows, sixteen times its pretraining context: the all-label link prompts reach that length on 8 of 13

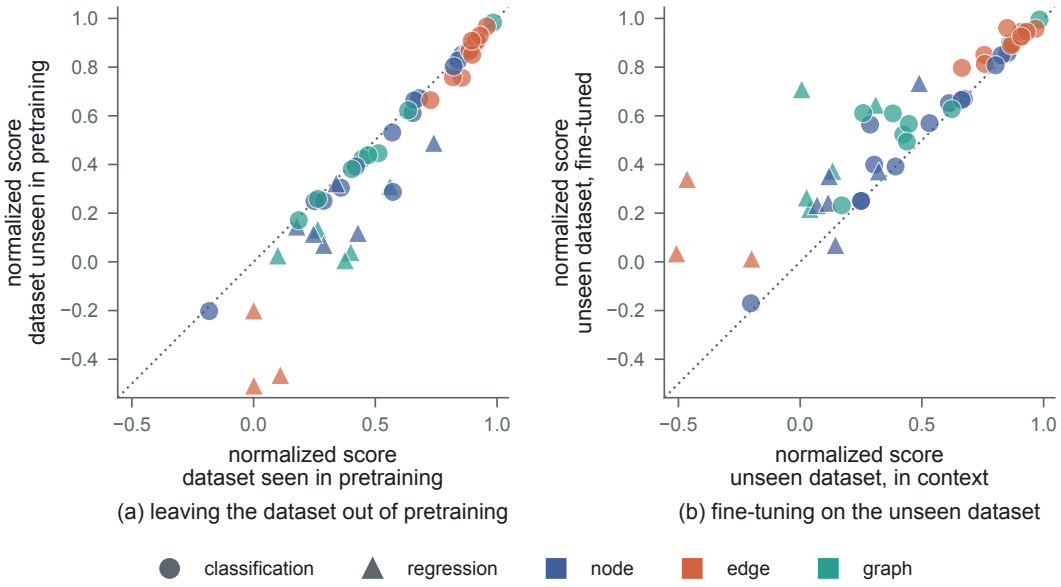

Figure 4: What withholding a dataset family from pretraining costs, and what fine-tuning on the dataset buys back. One point per dataset at the budget of Table 2, in normalized score (Appendix B). (a) x: the fold models whose pretraining included the dataset's family; y: the fold model pretrained without it, in context. (b) x: that fold model in context; y: the same model fine-tuned on the dataset's training labels. Dotted: no change. Circles: classification, triangles: regression; colour: level.

graphs, and Graft still leads the same-label GNN on every link graph there (Table 2; Appendix D, Figure 5). Performance rises with the prompt and levels off a few times past the pretraining context, which points to longer pretraining prompts as the most direct lever for a stronger model.

5 Discussion and Limitations

Limitations. The main benchmark holds out rows, not datasets: Graft may have seen other labeled rows of a dataset in pretraining, while the GNN it is compared with sees only the prompt's labels, so Section 4.3 is the stricter test. There, having seen a dataset's family matters mainly for domain-specific regression, and fine-tuning on the dataset's labels recovers much of it. At graph level a GNN trained on the full split still leads. The model does not yet read edge features, and node features enter through a label-free 32-column projection shared with the baselines (Appendix C).

Implications. A practitioner with a graph and a few dozen labels needs no training loop: a Graft prediction takes 0.1 to 1 s of network time against 1.4 to 4.7 s to train and select the GNN on one GPU (Appendix G). A new kind of label function now enters through the prior, as a new generator, rather than through the architecture. The recipe also has room to grow: Graft was pretrained on one GPU per seed, on prompts of at most 4096 rows. With pretraining at the context lengths it already handles at inference, and at a larger scale, we expect a single model of this kind to close the remaining gaps to networks trained on each dataset's full split; it already leads networks trained on the same labels at link prediction, at every budget, and at regression from few labels.

6 Conclusion

We presented Graft, a single pretrained model that answers node, edge and graph classification and regression from labeled examples in its prompt, with no gradient step. It matches or beats a GNN trained on the same labels on 45 of 49 dataset–task pairs, its classification carries over to dataset families it never saw, and, unlike a network trained for one dataset, it can grow with pretraining data and compute.

REPRODUCIBILITY STATEMENT

Appendix C states what the model is given at test time and how every held-out split is enforced, including the removal of same-task duplicates across benchmarks; Appendix A.1 gives the full training recipe as resolved at training time, and Appendix F the baseline protocol. Anonymized code for pretraining, the evaluation batteries and every table and figure is provided as supplementary material, together with the dataset holdout files.

AI USE STATEMENT

We used large language model assistants throughout this work: to implement the model, the synthetic graph generators and the training and evaluation code; to help set up experiments and ablations; to audit datasets for duplicates and leakage; and to search the literature, prepare figures and draft and edit the text. The authors defined the research problem and made the decisions throughout: what to test next, the method, the evaluation protocol and every claim. We take full responsibility for the content of this paper.

REFERENCES

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? Investigations with linear models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=0g0X4H8yN4I.

Arvindh Arun, Sumit Kumar, Mojtaba Nayyeri, Bo Xiong, Ponnurangam Kumaraguru, Antonio Vergari, and Steffen Staab. SEMMA: A semantic aware knowledge graph foundation model. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 31813–31836, 2025. doi: 10.18653/v1/2025.emnlp-main.1621.

Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. In *Advances in Neural Information Processing Systems*, volume 36, pp. 57125–57211, 2023.

Manel Baradad, Jonas Wulff, Tongzhou Wang, Phillip Isola, and Antonio Torralba. Learning to see by looking at noise. In *Advances in Neural Information Processing Systems*, 2021.

Gleb Bazhenov, Oleg Platonov, and Liudmila Prokhorenkova. GraphLand: Evaluating graph machine learning models on diverse industrial data. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 38, pp. 110211–110237, 2025.

Maya Bechler-Speicher, Yoel Gottlieb, Andrey Isakov, David Abensur, Ami Tavory, Daniel Haimovich, Ido Guy, and Udi Weinsberg. Billion-scale graph foundation models, 2026. URL https://arxiv.org/abs/2602.04768.

Karim K. Ben Hicham, Jan G. Rittig, Martin Grohe, and Alexander Mitsos. Tabular foundation models for in-context prediction of molecular properties. *arXiv preprint arXiv:2604.16123*, 2026.

Matthias Busch, Marius Tacke, Sviatlana V. Lamaka, Mikhail L. Zheludkevich, Christian J. Cyron, Christian Feiler, and Roland C. Aydin. In-context molecular property prediction with LLMs: A blinding study on memorization and knowledge conflicts. *Journal of Chemical Information and Modeling*, 2026. doi: 10.1021/acs.jcim.6c00961.

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.

Yuanning Cui, Zequn Sun, and Wei Hu. A prompt-based knowledge graph foundation model for universal in-context reasoning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 7095–7124, 2024.

Kaiwen Dong, Haitao Mao, Zhichun Guo, and Nitesh V. Chawla. Universal link predictor by in-context learning on graphs. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=EYpqmoejB8.

Vijay Prakash Dwivedi, Ladislav Rampášek, Mikhail Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 35, pp. 22326–22340, 2022.

Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24 (43):1–48, 2023. URL https://jmlr.org/papers/v24/22-0567.html.

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

Dmitry Eremeev, Gleb Bazhenov, Oleg Platonov, Artem Babenko, and Liudmila Prokhorenkova. Turning tabular foundation models into graph foundation models, 2025a.

Dmitry Eremeev, Oleg Platonov, Gleb Bazhenov, Artem Babenko, and Liudmila Prokhorenkova. GraphPFN: A prior-data fitted graph foundation model, 2025b.

Matthias Fey, Vid Kocijan, Federico Lopez, Jan Eric Lenssen, and Jure Leskovec. KumoRFM: A foundation model for in-context learning on relational data. Technical report, Kumo.ai, 2025. URL https://web.archive.org/web/20250524094415/https://kumo.ai/research/kumo_relational_foundation_model.pdf.

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

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.

Yisen Gao, Jiaxin Bai, Haoyu Huang, Zhongwei Xie, Yufei Li, Hong Ting Tsang, Sirui Han, and Yangqiu Song. KGPFN: Unlocking the potential of knowledge graph foundation model via in-context learning. *arXiv preprint arXiv:2605.14907*, 2026.

Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? A case study of simple function classes. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30583–30598, 2022.

F. Maxwell Harper and Joseph A. Konstan. The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):19:1–19:19, 2015. doi: 10.1145/2827872.

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.

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. doi: 10.1145/3690624.3709277.

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*, 637:319–326, 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. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Qian Huang, Hongyu Ren, Peng Chen, Gregor Kržmanc, Daniel Zeng, Percy Liang, and Jure Leskovec. PRODIGY: Enabling in-context learning over graphs. In *Advances in Neural Information Processing Systems*, 2023.

Valter Hudovernik, Federico López, Vid Kocijan, Akihiro Nitta, Jan Eric Lenssen, Jure Leskovec, and Matthias Fey. KumoRFM-2: Scaling foundation models for relational learning, 2026. URL https://arxiv.org/abs/2604.12596.

John J. Irwin, Teague Sterling, Michael M. Mysinger, Erin S. Bolstad, and Ryan G. Coleman. ZINC: A free tool to discover chemistry for biology. *Journal of Chemical Information and Modeling*, 52 (7):1757–1768, 2012. doi: 10.1021/ci3001277.

Sergei Ivanov, Sergei Sviridov, and Evgeny Burnaev. Understanding isomorphism bias in graph data sets. *arXiv preprint arXiv:1910.12091*, 2019. doi: 10.48550/arXiv.1910.12091.

Hirokatsu Kataoka, Kazushige Okayasu, Asato Matsumoto, Eisuke Yamagata, Ryosuke Yamada, Nakamasa Inoue, Akio Nakamura, and Yutaka Satoh. Pre-training without natural images. In *Asian Conference on Computer Vision*, 2020.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Srijan Kumar, Francesca Spezzano, V. S. Subrahmanian, and Christos Faloutsos. Edge weight prediction in weighted signed networks. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pp. 221–230, 2016. doi: 10.1109/ICDM.2016.0033.

Juanhui Li, Harry Shomer, Haitao Mao, Shenglai Zeng, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin. Evaluating graph neural networks for link prediction: Current pitfalls and new benchmarking. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 36, pp. 3853–3866, 2023.

Qinyang Li, Rongzhi Dong, Nicholas Miklaucic, Jeffrey Hu, Sadman Sadeed Omee, Lai Wei, Sourin Dey, Ming Hu, and Jianjun Hu. In context learning foundation models for materials property prediction with small datasets. *arXiv preprint arXiv:2601.00133*, 2025.

Tianyin Liao, Chunyu Hu, Yicheng Sui, Xingxuan Zhang, Peng Cui, Jianxin Li, and Ziwei Zhang. TFMLinker: Universal link predictor by graph in-context learning with tabular foundation models. *arXiv preprint arXiv:2602.08592*, 2026.

Tianqianjin Lin, Pengwei Yan, Kaisong Song, Zhuoren Jiang, Yangyang Kang, Jun Lin, Weikang Yuan, Junjie Cao, Changlong Sun, and Xiaozhong Liu. LangGFM: A large language model alone can be a powerful graph foundation model. *arXiv preprint arXiv:2410.14961*, 2024.

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

Yuyan Liu, Sirui Ding, Sheng Zhou, Wenqi Fan, and Qiaoyu Tan. MolecularGPT: Open large language model (LLM) for few-shot molecular property prediction. *arXiv preprint arXiv:2406.12950*, 2024b.

Weishuo Ma, Yanbo Wang, Xiyuan Wang, Lei Zou, and Muhan Zhang. GILT: An LLM-free, tuning-free graph foundational model for in-context learning, 2025. URL https://arxiv.org/abs/2510.04567.

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 *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*. PMLR, 2024.

Péter Mernyei and Cătălina Cangea. Wiki-CS: A Wikipedia-based benchmark for graph neural networks. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+)*, 2020.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+)*, 2020.

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

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.

Oleg Platonov, Gleb Bazhenov, Dmitry Eremeev, and Liudmila Prokhorenkova. A fair evaluation of graph foundation models for node property prediction, 2026.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 50817–50847, 2025.

Rishabh Ranjan, Valter Hudovernik, Mark Znidar, Charilaos Kanatsoulis, Roshan Upendra, Mahmoud Mohammadi, Joe Meyer, Tom Palczewski, Carlos Guestrin, and Jure Leskovec. Relational transformer: Toward zero-shot foundation models for relational data. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=rpPtgMC5s9.

Allan Raventós, Mansheej Paul, Feng Chen, and Surya Ganguli. Pretraining task diversity and the emergence of non-Bayesian in-context learning for regression. In *Advances in Neural Information Processing Systems*, 2023.

Ivan Rubachev, Akim Kotelnikov, Nikolay Kartashev, and Artem Babenko. On finetuning tabular foundation models, 2025.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Yangyi Shen, Jincheng Zhou, Beatrice Bevilacqua, Joshua Robinson, Charilaos Kanatsoulis, Jure Leskovec, and Bruno Ribeiro. Zero-shot generalization of GNNs over distinct attribute domains. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 54701–54731, 2025. URL https://proceedings.mlr.press/v267/shen25p.html.

Murat Cihan Sorkun, Abhishek Khetan, and Süleyman Er. AqSolDB, a curated reference set of aqueous solubility and 2D descriptors for a diverse set of compounds. *Scientific Data*, 6(1):143, 2019. doi: 10.1038/s41597-019-0151-1.

Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. All in one: Multi-task prompting for graph neural networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2120–2131, 2023. doi: 10.1145/3580305.3599256.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 35151–35174, 2023.

Duo Wang, Yuan Zuo, Guangyue Lu, and Junjie Wu. UniGTE: Unified graph–text encoding for zero-shot generalization across graph tasks and domains. In *Advances in Neural Information Processing Systems*, volume 38, pp. 102197–102222, 2025a. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/84e3cc09f56108559cd28754d94c5e06-Abstract-Conference.html.

Zehong Wang, Zheyuan Liu, Tianyi Ma, Jiazheng Li, Zheyuan Zhang, Xingbo Fu, Yiyang Li, Zhengqing Yuan, Wei Song, Yijun Ma, Qingkai Zeng, Xiusi Chen, Jianan Zhao, Jundong Li, Meng Jiang, Pietro Liò, Nitesh Chawla, Chuxu Zhang, and Yanfang Ye. Graph foundation models: A comprehensive survey. *arXiv preprint arXiv:2505.15116*, 2025b.

Zehong Wang, Zheyuan Zhang, Tianyi Ma, Nitesh V. Chawla, Chuxu Zhang, and Yanfang Ye. Towards graph foundation models: Learning generalities across graphs via task-trees. In *International Conference on Machine Learning*, 2025c.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, pp. 6861–6871, 2019.

Yuhuai Wu, Markus Rabe, Wenda Li, Jimmy Ba, Roger Grosse, and Christian Szegedy. LIME: Learning inductive bias for primitives of mathematical reasoning. In *International Conference on Machine Learning*, 2021.

Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: A benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018. doi: 10.1039/C7SC02664A.

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

Lianghao Xia, Ben Kao, and Chao Huang. OpenGraph: Towards open graph foundation models. In *Findings of the Association for Computational Linguistics: EMNLP*, 2024.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ryGs6iA5Km`.

Pinar Yanardag and S. V. N. Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1365–1374, 2015. doi: 10.1145/2783258.2783417.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 40–48, 2016.

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

Jitao Zhao, Di Jin, Meng Ge, Lianze Shan, Xin Wang, Dongxiao He, and Zhiyong Feng. FUG: Feature-universal graph contrastive pre-training for graphs with diverse node features. In *Advances in Neural Information Processing Systems*, volume 37, pp. 4003–4034, 2024.

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

A REPRODUCIBILITY, TRAINING RECIPE AND PRIOR

Every number in the main text is produced by the code in the supplementary material, from three pretraining seeds of one recipe (230k steps); its resolved configuration is written beside every checkpoint, so the run that produced a number can be reconstructed exactly rather than inferred from prose.

Cost. Each pretraining seed trains in 21 to 22 hours on one V100 (Appendix G).

Seeds. Results are reported over three pretraining seeds, as a mean; every method is also averaged over five context draws per cell.

A.1 HYPERPARAMETERS AND THE TRAINING RECIPE

Table 4: The recommended model, as resolved at training time.

encoder		*in-context head*	
backbone	linear attention	rows transformer	6 layers, 8 heads
layers, heads	8, 8	row pooling	mean, std, max, log size
width	384	readout	64 buckets + learned residual
tokenizer	exchangeable, column mean	ridge projection	16 dimensions
random node identifiers	64 dimensions	ridge feature hops	2 and 4
structural inputs	log degree, log size	label propagation	8 steps
neighbourhood term	mean + degree-scaled	closed forms into rows	prototype, ridge
prior and prompts		*optimisation*	
graph families	16, drawn uniformly	steps	230k
nodes per graph	10 to 600	level probabilities	0.34 / 0.33 / 0.33 (nominal)
average degree	1.8 to 80	schedule	constant to 200k, cosine to 230k
feature columns	1 to 128	peak learning rate	3×10^{-4}
topology pool	512, refreshed	classification share	0.5 of level prompts
graphs per graph prompt	4 to 56	feature masking rate	0.15
rows per node/edge prompt	$\leq$4096 context, $\leq$32 query	real-label fraction	0.30
graphs per node/edge prompt	3 (synthetic), 1 (real)	real-regression fraction	0.15
evaluation feature width	32 (PCA)	parameters	27.0M

A.2 THE PRETRAINING PRIOR

The sixteen random-graph families of the prior, drawn uniformly, are Erdős-Rényi, Barabási-Albert, Watts-Strogatz, stochastic block models with degree-corrected and hierarchical variants, random geometric graphs, graphons, planted motifs and R-MAT, together with a sparse and long-diameter corner of trees, rings, fused rings, bipartite graphs, two-dimensional lattices and unions of overlapping cliques. Sizes are 10 to 600 nodes and average degree is drawn between 1.8 and 80. Label functions are drawn per prompt and span all three levels: linear and nonlinear functions of a node's features and its propagated neighbourhood at node level; link existence with the query positives removed from the visible graph, common-neighbour count and capped shortest-path distance at edge level; structural and size targets at graph level. A step draws a real prompt instead at the real-label and real-regression fractions of Table 4; the two sources are never crossed.

B METRICS

Tables report each dataset's native metric: accuracy for multiclass node classification, ROC-AUC for binary node classification (Minesweeper, Tolokers, Questions), link prediction and graph classification, and R^2 for every regression task. Where metrics are mixed (Figure 5) we use a normalized score on which 0 is the trivial predictor and 1 a perfect one: R^2 for regression, $2\,\mathrm{AUC} - 1$ for ROC-AUC and $(\mathrm{acc} - m)/(1 - m)$ for accuracy, with m the majority rate. Within a dataset it ranks methods as the native metric does.

C DATASETS AND DATA PROCESSING

C.1 SOURCES AND STATISTICS

Table 5 lists every dataset: the benchmark, each of which is also in pretraining with its test rows held out (Appendix C), and the datasets used only in pretraining. Sources: Planetoid (Yang et al., 2016), Amazon and Coauthor (Shchur et al., 2018), WikiCS (Mernyei & Cangea, 2020), the heterophilous graphs (Platonov et al., 2023), GraphLand (Bazhenov et al., 2025), Bitcoin-Alpha and Bitcoin-OTC (Kumar et al., 2016), MovieLens (Harper & Konstan, 2015), TUDataset (Morris et al., 2020) with IMDB (Yanardag & Vishwanathan, 2015), OGB (Hu et al., 2020), MoleculeNet (Wu et al., 2018), AqSolDB (Sorkun et al., 2019), ZINC (Irwin et al., 2012), the superpixel and ZINC benchmarks of Dwivedi et al. (2023), and LRGB (Dwivedi et al., 2022).

Table 5: Every real dataset in this paper, by task. Nodes and edges: the whole graph, or for graph-level datasets the mean per graph; edges are unique undirected pairs. Features: raw width before the 32-column projection (0: featureless, log degree is used). The two MNIST superpixel tasks share one source, so the pretraining corpus counts 59 datasets. Every dataset is used in pretraining by the full model, the benchmark datasets with their test rows held out; regression is reported as R^2.

dataset	domain	metric	nodes	edges	graphs	features	classes / target
node classification							
Amazon-ratings	co-purchase	acc.	24,492	93,050		300	5
CiteSeer	citation	acc.	3,327	4,552		3,703	6
Computers	co-purchase	acc.	13,752	245,861		767	10
Cora	citation	acc.	2,708	5,278		1,433	7
CS	co-authorship	acc.	18,333	81,894		6,805	15
Minesweeper	synthetic grid	ROC-AUC	10,000	39,402		7	2
Photo	co-purchase	acc.	7,650	119,081		745	8
Physics	co-authorship	acc.	34,493	247,962		8,415	5
PubMed	citation	acc.	19,717	44,324		500	3
Questions	Q&A	ROC-AUC	48,921	153,540		301	2
Roman-empire	text	acc.	22,662	32,927		300	18
Tolokers	crowdsourcing	ROC-AUC	11,758	519,000		10	2
WikiCS	web	acc.	11,701	215,603		300	10
Actor	web		7,600	26,659		932	5
artnet-exp	web		50,405	280,348		75	2
chameleon	web		2,277	31,371		2,325	5
Flickr	social		89,250	449,878		500	7
squirrel	web		5,201	198,353		2,089	5
tolokers-2	crowdsourcing		11,758	519,000		19	2
node regression							
artnet-views	web	R^2	50,405	280,348		50	log number of views
avazu-ctr	advertising	R^2	76,269	10,984,077		260	click-through rate
city-roads-L	road networks	R^2	142,257	231,550		207	travel velocity
city-roads-M	road networks	R^2	57,073	107,104		68	travel velocity
hm-prices	co-purchase	R^2	46,563	10,730,995		264	product price
twitch-views	social	R^2	168,114	6,797,557		24	log views
arxiv-year	citation		169,343	1,157,799		128	publication year
city-reviews	social		148,801	1,165,415		204	is_fraudster
link prediction							
Amazon-ratings	co-purchase	ROC-AUC	24,492	93,050		300	binary
CiteSeer	citation	ROC-AUC	3,327	4,552		3,703	binary
Computers	co-purchase	ROC-AUC	13,752	245,861		767	binary
Cora	citation	ROC-AUC	2,708	5,278		1,433	binary
CS	co-authorship	ROC-AUC	18,333	81,894		6,805	binary
Minesweeper	synthetic grid	ROC-AUC	10,000	39,402		7	binary
Photo	co-purchase	ROC-AUC	7,650	119,081		745	binary
Physics	co-authorship	ROC-AUC	34,493	247,962		8,415	binary
PubMed	citation	ROC-AUC	19,717	44,324		500	binary
Questions	Q&A	ROC-AUC	48,921	153,540		301	binary

dataset	domain	metric	nodes	edges	graphs	features	classes / target
Roman-empire	text	ROC-AUC	22,662	32,927		300	binary
Tolokers	crowdsourcing	ROC-AUC	11,758	519,000		10	binary
WikiCS	web	ROC-AUC	11,701	215,603		300	binary
edge regression							
BitcoinAlpha	trust	R^2	7,605	14,124		1	trust rating
BitcoinOTC	trust	R^2	6,005	21,492		1	trust rating
MovieLens100K	recommendation	R^2	2,625	80,000		25	rating
graph classification							
CIFAR10-sp-0v1	vision	ROC-AUC	117	563	2,000	5	2
IMDB-BINARY	social	ROC-AUC	19.8	96.5	1,000	0	2
MNIST-sp-0v1	vision	ROC-AUC	66.7	323	2,115	3	2
MNIST-sp-3v8	vision	ROC-AUC	71.8	348	1,984	3	2
molbbbp	molecules	ROC-AUC	24.1	25.9	2,039	9	2
molhiv	molecules	ROC-AUC	25.5	27.5	41,127	9	2
MUTAG	molecules	ROC-AUC	17.9	19.8	188	7	2
NCI1	molecules	ROC-AUC	29.9	32.3	4,110	37	2
PROTEINS	proteins	ROC-AUC	39.1	72.8	1,113	3	2
AIDS	molecules		15.7	16.2	2,000	38	2
BZR	molecules		35.8	38.4	405	53	2
COLLAB	social		74.5	2,457	5,000	0	3
COX2	molecules		41.2	43.5	467	35	2
DD	proteins		284	716	1,178	89	2
DHFR	molecules		42.4	44.5	756	53	2
ENZYMES	proteins		32.6	62.1	600	3	6
IMDB-MULTI	social		13	65.9	1,500	0	3
moltox21	molecules		18.6	19.3	7,831	9	2
peptides-func	molecules		152	154	10,873	9	2
PTC_MR	molecules		14.3	14.7	344	18	2
REDDIT-BINARY	social		430	498	2,000	0	2
graph regression							
AQSOL	molecules	R^2	15.4	15.3	7,836	1	aqueous solubility
ESOL	molecules	R^2	13.3	13.7	1,128	9	water solubility
FreeSolv	molecules	R^2	8.7	8.4	642	9	hydration free energy
Lipo	molecules	R^2	27.0	29.5	4,200	9	lipophilicity
ZINC	molecules	R^2	23.2	24.9	10,000	1	penalized logP
alchemy	molecules		10.1	10.4	202,579	6	target column 0 of 12
Peptides-struct	molecules		152	154	10,873	9	target column 0 of 11
QM7b	molecules		15.4	115	7,211	0	target column 0 of 14
QM9	molecules		18.0	18.7	130,831	11	target 0 of 19: dipole moment mu

C.2 Duplicates across and within datasets

Overlap between pretraining and test data has been documented for graph foundation models (Arun et al., 2025), and isomorphic duplicates within TU datasets (Ivanov et al., 2019). A graph is removed from pretraining when the same entity with the same task appears in any benchmark's held-out split. Same entity means an identical heavy-atom molecular graph (identical under element-labelled graph isomorphism), or, for the two protein sets built from one source, the same protein. Same task means the same measured quantity. A molecule that appears under a different task stays, because seeing its structure while learning a different property does not reveal the test label. Exclusions of this kind cover AQSOL and ESOL (47% of ESOL's test molecules), AIDS and ogbg-molhiv (a fifth of its test positives), DD and PROTEINS, and within-dataset duplicates of eighteen molecular sets and of IMDB-BINARY and IMDB-MULTI. PowerGraph is excluded because its held-out graphs have input-identical training counterparts.

C.3 Feature preprocessing

Node features are reduced to 32 columns by PCA before the model sees them, with the same reduction at pretraining and at evaluation. The projection is fitted without labels on each dataset's full feature matrix, including the held-out nodes, and the GNN and closed-form baselines receive the same features; the other in-context models read the raw features. A wider projection is not a free improvement: at 128 columns the heterophilous Roman-empire graph gains, four homophilous graphs lose and Amazon-ratings falls to chance, so we report the 32-column setting.

C.4 Long contexts

The model is pretrained on node- and edge-level prompts of at most 4096 context rows and on graph-level prompts of at most 56 context graphs. Longer prompts at evaluation run in one forward pass: blockwise attention keeps memory linear in the number of rows and matches the standard computation to numerical precision.

D Performance against context

Figure 5 follows each method across context sizes, averaged per task type; the tables of Appendix E give every dataset.

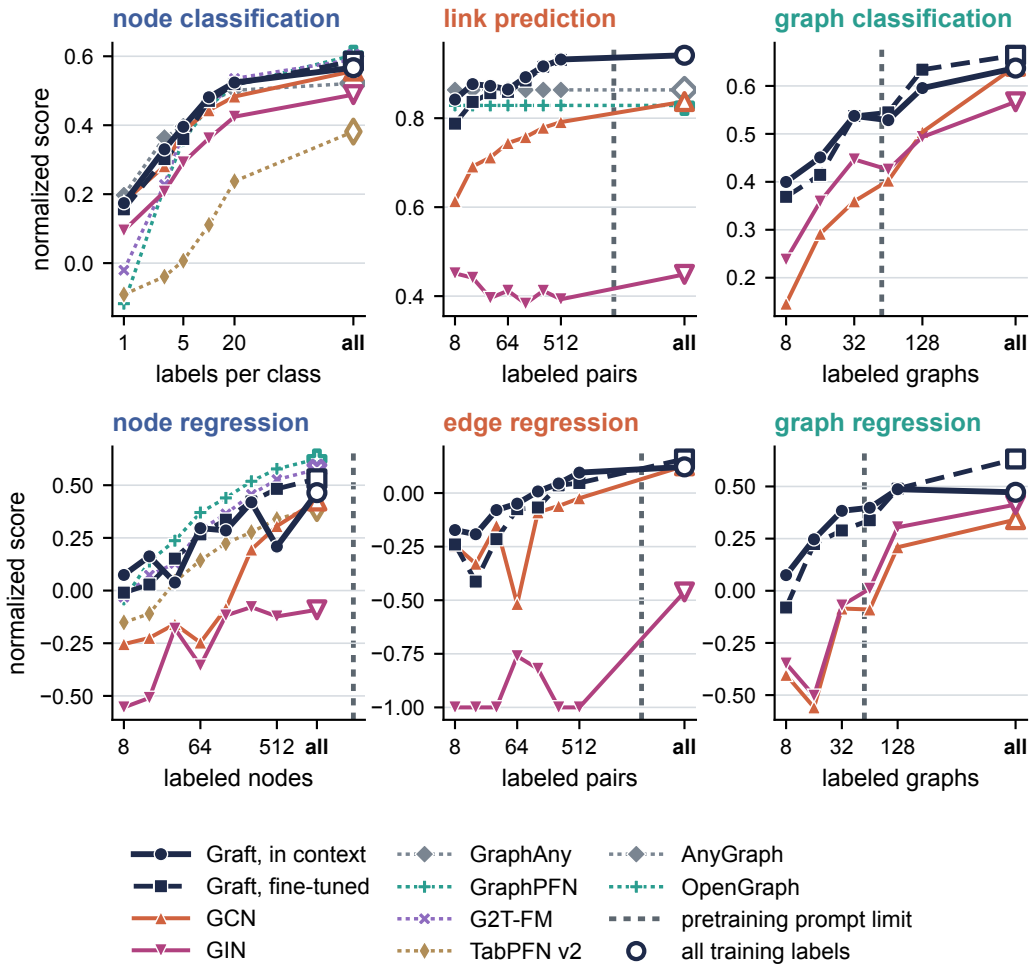

Figure 5: Every method against context size: mean normalized score over each task type's datasets, with each dataset's score floored at -1. Graft in context (three seeds) and fine-tuned on the same K labels; GCN and GIN on the same K labels; the other in-context models on the datasets they run on. *all*: every training label, at the task's median prompt size. Thick dotted line: the largest prompt in pretraining (4096 rows; 56 graphs).

E EVERY RESULT

Tables 6–11 give every method on every dataset of the main benchmark at every context size K, in each dataset's native metric. Graft is averaged over three pretraining seeds and five context draws, "Graft FT" is the same model fine-tuned on the K labels, and every other entry is the mean of five context draws; large negative R^2 values are GNN fits that diverged on few labels. GCN and GIN are trained on exactly the K labels in Graft's prompt, which they split into training and validation halves; the closed forms are fitted on the same K labels. *all*: every training label (their number in brackets), in one prompt for Graft and in training for the GNN. In each row the three best methods given the same labels are marked **1st**, **2nd** and **3rd**. A grey dash marks a cell not scored: Graft FT at all labels where the fine-tuning prompt does not fit in 40 GB (most link graphs, MovieLens, molhiv, ZINC, Lipo and the superpixel sets), TabPFN v2 beyond ten classes, and degenerate one-label-per-class cells.

Table 6: Node classification, every method at every context size (accuracy; ROC-AUC for Minesweeper, Tolokers and Questions; K labels per class).

dataset	K	Graft	Graft FT	GCN	GIN	ridge	log. reg.	Graph Any	Graph PFN	G2T-FM	Tab PFN
Amazon-ratings	1	.186	**.242**	**.211**	.195	.202	.204	**.208**	.142	.200	.184
	3	.203	.204	**.210**	.199	**.213**	.205	**.209**	.192	.196	.208
	5	**.225**	.209	**.225**	.186	**.216**	.197	.197	.199	.166	.202
	10	.226	.212	.215	**.232**	**.230**	.213	.221	**.232**	.217	.212
	20	**.254**	.236	.208	.230	**.257**	.197	.220	.234	**.238**	.213
	all (250)	**.265**	**.264**	.232	.248			.230	.232	**.267**	.218
CiteSeer	1	.371	.344	**.401**	.353	**.373**	.246	**.390**	.227	.218	.221
	3	.480	.389	**.483**	.462	**.482**	.345	**.481**	.385	.346	.226
	5	.540	.539	**.550**	.536	**.546**	.404	**.599**	.539	.452	.256
	10	**.608**	.605	**.606**	.559	.577	.459	**.643**	.594	.506	.277
	20	**.673**	**.676**	.644	.591	.613	.514	**.678**	.655	.574	.348
	all (620)	.689	**.757**	**.733**	.706			.703	**.740**	.720	.584
Computers	1	.557	**.566**	**.566**	.472	.551	.261	**.564**	.237	.183	.170
	3	**.715**	.699	**.711**	.569	.698	.405	**.721**	.631	.592	.178
	5	**.694**	.676	**.736**	.587	.668	.466	**.729**	.671	.656	.168
	10	**.765**	.757	.761	.672	.712	.561	**.788**	**.766**	.708	.320
	20	**.804**	.803	.794	.755	.774	.635	**.811**	**.806**	.749	.485
	all (500)	.821	**.822**	**.837**	.784			.765	**.858**	.813	.592
Cora	1	.406	.403	**.487**	**.427**	**.447**	.238	.366	.172	.291	.192
	3	**.567**	.552	**.571**	.518	.562	.333	**.593**	.417	.444	.198
	5	**.650**	.618	**.642**	.624	**.640**	.378	.620	.574	.579	.239
	10	**.724**	**.704**	.673	.656	.696	.450	**.736**	.680	.679	.289
	20	**.763**	**.759**	.741	.708	.720	.515	**.759**	.726	.707	.410
	all (640)	.796	**.842**	.827	.794			.751	**.851**	**.830**	.594
CS	1	**.544**	.526	**.597**	.447	.519	.294	**.641**	.207	.210	–
	3	**.796**	**.797**	.794	.667	.770	.482	**.876**	.740	.688	–
	5	**.808**	.802	.807	.721	.794	.553	**.867**	**.820**	.798	–
	10	**.874**	.873	.861	.783	.823	.691	**.900**	**.902**	.863	–
	20	**.896**	.893	.878	.822	.801	.774	**.908**	**.910**	.889	–
	all (750)	**.914**	.912	.898	.848			.909	**.929**	**.914**	–
Minesweeper	1	**.518**	.511	.503	.505	.507	.496	**.520**	–	**.530**	–
	3	**.573**	.567	.547	.556	.547	.500	**.594**	.515	**.620**	.502
	5	.553	.464	**.558**	.541	.529	.502	**.632**	**.558**	–	–
	10	**.659**	.651	.642	.640	.562	.499	**.719**	.606	**.800**	.500
	20	**.642**	.628	.611	**.634**	.525	.498	**.730**	.599	–	–
	all (100)	.737	**.749**	.679	.661			**.788**	.687	**.830**	.498
Photo	1	**.639**	**.635**	**.646**	.535	.601	.340	.634	.305	.315	.239
	3	**.739**	.692	.737	.588	**.741**	.519	**.765**	.705	.651	.278
	5	.806	.796	**.815**	.739	.807	.625	**.816**	.792	**.811**	.325
	10	**.862**	**.868**	.848	.791	.853	.722	.851	**.868**	.836	.448

continued

dataset	K	Graft	Graft FT	GCN	GIN	ridge	log. reg.	Graph Any	Graph PFN	G2T-FM	Tab PFN
	20	.886	**.891**	.883	.822	.842	.782	.876	**.892**	**.889**	.622
	all (400)	**.910**	**.915**	.905	.864			.872	**.924**	.900	.708
Physics	1	.608	.500	**.624**	.585	.529	.324	**.761**	.327	**.609**	.493
	3	**.795**	.730	.733	.749	.739	.497	**.890**	.720	**.802**	.410
	5	.845	.837	**.858**	.772	.800	.591	**.905**	.826	**.863**	.540
	10	**.882**	.870	.870	.812	.844	.665	**.914**	.860	**.880**	.569
	20	**.916**	**.917**	.903	.819	.890	.798	**.923**	.914	.898	.671
	all (250)	**.932**	.925	.927	.886			**.936**	**.932**	.926	.839
PubMed	1	**.533**	**.522**	.486	.446	.502	.439	**.509**	.361	.426	.355
	3	**.601**	.572	.507	.547	**.608**	.533	**.632**	.457	.494	.515
	5	**.664**	.571	.620	.626	**.658**	.524	**.659**	.502	.583	.551
	10	**.724**	**.709**	.701	.660	.704	.621	**.724**	.625	.671	.618
	20	**.752**	**.751**	.735	.726	.734	.664	**.747**	.721	.726	.656
	all (560)	.787	.824	**.828**	.796			.627	**.829**	.827	**.873**
Questions	1	**.598**	**.593**	**.580**	.526	.566	.510	.531	.478	.559	.517
	3	**.572**	**.590**	**.574**	.481	.531	.532	.537	.533	.559	.522
	5	.630	**.631**	**.641**	.575	.595	.576	.581	**.630**	–	.578
	10	**.627**	**.631**	**.615**	.525	.563	.557	.550	.614	.554	.583
	20	**.643**	**.645**	.634	.628	.579	.562	.562	**.671**	–	.586
	all (100)	.653	.657	**.667**	**.661**			.568	**.694**	.593	.615
Roman-empire	1	**.185**	.148	.113	.085	.178	**.237**	**.246**	.109	.094	–
	3	**.284**	.272	.139	.113	.262	**.290**	**.302**	.272	.182	–
	5	**.348**	**.345**	.160	.127	.299	**.352**	.341	.338	.311	–
	10	**.426**	**.428**	.209	.183	.281	.398	.368	**.403**	.370	–
	20	**.468**	**.476**	.256	.214	.215	.437	.350	**.451**	.434	–
	all (900)	.490	**.537**	.326	.283			.482	**.523**	**.517**	–
Tolokers	1	**.577**	**.589**	**.580**	.545	.484	.515	.558	.493	.525	.530
	3	.562	**.583**	.508	.523	.501	**.582**	**.565**	.554	.510	.541
	5	.638	**.647**	.642	.602	.624	**.657**	.512	**.658**	–	–
	10	.666	.657	**.690**	.603	.648	.645	.620	**.711**	**.673**	.640
	20	.676	.667	**.708**	.637	.627	.675	**.703**	**.729**	–	–
	all (100)	.691	.663	.719	.577			**.738**	**.775**	**.758**	.673
WikiCS	1	.388	**.400**	.351	.311	**.415**	.260	**.391**	.187	.194	.231
	3	**.594**	.586	.567	.434	**.592**	.430	**.593**	.461	.454	.403
	5	**.617**	**.614**	**.612**	.478	.594	.459	.597	.583	.485	.406
	10	**.699**	**.697**	.678	.601	**.679**	.529	.598	.672	.625	.532
	20	**.746**	**.747**	.727	.641	.689	.597	.597	**.738**	.684	.641
	all (500)	**.769**	**.770**	.761	.681			.702	**.780**	.734	.697

Table 7: Link prediction, every method at every context size (ROC-AUC; K labeled pairs).

dataset	K	Graft	Graft FT	GCN	GIN	log. reg.	Any Graph	Open Graph
Amazon-ratings	8	.966	.940	.789	.723	**.971**	**.993**	**.995**
	16	**.981**	.944	.849	.755	.979	**.993**	**.995**
	32	.981	.968	.889	.721	**.985**	**.993**	**.995**
	64	.969	.960	.905	.747	**.985**	**.993**	**.995**
	128	.975	.974	.922	.729	**.987**	**.993**	**.995**
	256	.988	**.988**	.924	.756	.987	**.993**	**.995**
	512	.990	**.991**	.930	.680	.989	**.993**	**.995**
	all (65,536)	**.995**	–	.985	.789		**.993**	**.995**
CiteSeer	8	**.929**	.882	.793	.682	**.896**	**.909**	.866
	16	**.944**	**.925**	.851	.694	.890	**.909**	.866
	32	**.930**	**.913**	.848	.715	.887	**.909**	.866
	64	**.917**	**.914**	.837	.716	.886	**.909**	.866

continued

dataset	K	Graft	Graft FT	GCN	GIN	log. reg.	Any Graph	Open Graph
	128	**.933**	**.925**	.840	.738	.886	**.909**	.866
	256	**.947**	**.942**	.856	.736	.877	**.909**	.866
	512	**.953**	**.952**	.861	.749	.871	**.909**	.866
	all (4,097)	**.963**	**.963**	.897	.767		**.909**	.866
Computers	8	**.941**	.906	.838	.805	.888	**.977**	**.970**
	16	**.953**	.926	.906	.727	.951	**.977**	**.970**
	32	**.953**	.950	.881	.719	.952	**.977**	**.970**
	64	.943	.946	.923	.680	**.950**	**.977**	**.970**
	128	.963	**.969**	.940	.695	.955	**.977**	**.970**
	256	**.974**	**.973**	.946	.685	.961	**.977**	.970
	512	**.979**	**.977**	.948	.711	.965	**.977**	.970
	all (65,536)	**.982**	–	**.976**	.707		**.977**	.970
Cora	8	**.902**	**.887**	.820	.690	.874	**.908**	.877
	16	**.908**	**.891**	.817	.695	.875	**.908**	.877
	32	**.908**	**.901**	.830	.696	.881	**.908**	.877
	64	**.893**	.880	.851	.703	**.889**	**.908**	.877
	128	**.908**	**.908**	.852	.704	.880	**.908**	.877
	256	**.930**	**.934**	.859	.695	.878	**.908**	.877
	512	**.939**	**.939**	.869	.705	.869	**.908**	.877
	all (4,750)	**.952**	**.955**	.906	.774		**.908**	.877
CS	8	**.924**	.889	.854	.792	.837	**.951**	**.966**
	16	**.951**	.937	.867	.764	**.958**	.951	**.966**
	32	.938	.938	.908	.691	**.962**	**.951**	**.966**
	64	.926	.920	.911	.700	**.961**	**.951**	**.966**
	128	.945	.932	.915	.682	**.965**	**.951**	**.966**
	256	**.960**	.959	.928	.677	**.962**	.951	**.966**
	512	**.970**	**.969**	.923	.716	.961	.951	**.966**
	all (65,536)	**.976**	–	**.963**	.660		.951	**.966**
Minesweeper	8	.845	**.853**	.662	.619	.717	**.999**	**.999**
	16	.926	**.927**	.686	.652	.810	**.999**	**.999**
	32	.917	**.931**	.670	.664	.817	**.999**	**.999**
	64	.950	**.952**	.731	.674	.832	**.999**	**.999**
	128	.961	**.963**	.727	.681	.838	**.999**	**.999**
	256	.969	**.972**	.767	.723	.830	**.999**	**.999**
	512	.974	**.976**	.792	.742	.843	**.999**	**.999**
	all (35,462)	**.974**	–	.790	.825		**.999**	**.999**
Photo	8	**.945**	.934	.831	.769	.807	**.980**	**.977**
	16	.951	.934	.894	.750	**.962**	**.980**	**.977**
	32	.953	.936	.915	.763	**.961**	**.980**	**.977**
	64	.957	.953	.932	.755	**.965**	**.980**	**.977**
	128	.958	.957	.953	.641	**.965**	**.980**	**.977**
	256	.975	**.978**	.958	.696	.969	**.980**	**.977**
	512	**.979**	**.977**	.959	.725	.965	**.980**	.977
	all (65,536)	**.987**	–	**.982**	.680		**.980**	.977
Physics	8	.954	.887	.857	.779	**.969**	**.977**	**.979**
	16	.952	.931	.862	.690	**.965**	**.977**	**.979**
	32	.945	.935	.903	.615	**.975**	**.977**	**.979**
	64	.913	.919	.896	.708	**.965**	**.977**	**.979**
	128	.944	.948	.926	.645	**.971**	**.977**	**.979**
	256	.966	.965	.928	.605	**.970**	**.977**	**.979**
	512	**.975**	.975	.938	.588	.969	**.977**	**.979**
	all (65,536)	**.980**	–	.973	.571		**.977**	**.979**
PubMed	8	**.935**	**.898**	.864	.748	**.865**	.788	.808
	16	**.947**	**.930**	.907	.766	**.921**	.788	.808
	32	**.957**	**.951**	**.914**	.741	.913	.788	.808
	64	**.948**	**.947**	**.908**	.687	.897	.788	.808
	128	**.957**	**.958**	.905	.707	**.908**	.788	.808

continued

dataset	K	Graft	Graft FT	GCN	GIN	log. reg.	Any Graph	Open Graph
	256	**.969**	**.965**	**.924**	.687	.892	.788	.808
	512	**.975**	**.972**	**.916**	.696	.896	.788	.808
	all (39,892)	**.980**	–	**.953**	.760		.788	**.808**
Questions	8	**.958**	**.944**	.810	.758	.741	**.851**	.633
	16	**.954**	**.931**	.830	.717	.740	**.851**	.633
	32	**.954**	**.934**	**.853**	.601	.768	.851	.633
	64	**.950**	**.953**	**.868**	.622	.756	.851	.633
	128	**.953**	**.946**	.847	.561	.753	**.851**	.633
	256	**.966**	**.966**	**.877**	.650	.772	.851	.633
	512	**.969**	**.969**	**.866**	.516	.763	.851	.633
	all (65,536)	**.969**	–	**.817**	.649		**.851**	.633
Roman-empire	8	**.802**	.783	.643	.522	.740	**.832**	**.898**
	16	**.835**	.795	.705	.555	.741	**.832**	**.898**
	32	**.858**	**.853**	.684	.576	.741	.832	**.898**
	64	**.851**	**.841**	.708	.572	.737	.832	**.898**
	128	**.878**	**.852**	.711	.597	.733	.832	**.898**
	256	**.881**	**.880**	.723	.623	.741	.832	**.898**
	512	**.905**	**.904**	.743	.652	.755	.832	**.898**
	all (29,634)	**.910**	–	.787	.701		**.832**	**.898**
Tolokers	8	**.923**	.894	.884	.881	.705	**.967**	**.957**
	16	**.936**	.924	.904	.914	.879	**.967**	**.957**
	32	**.931**	.918	.916	.836	.877	**.967**	**.957**
	64	**.954**	.949	.929	.850	.903	**.967**	**.957**
	128	**.964**	**.961**	.945	.938	.911	**.967**	.957
	256	**.963**	**.962**	.935	.936	.915	**.967**	.957
	512	**.972**	**.973**	.947	.922	.935	**.967**	.957
	all (65,536)	**.967**	–	.945	.889		**.967**	**.957**
WikiCS	8	**.948**	.921	.844	.664	.900	**.980**	**.962**
	16	**.960**	.943	.918	.690	.948	**.980**	**.962**
	32	**.948**	.938	.915	.734	.943	**.980**	**.962**
	64	.949	**.950**	.940	.766	.948	**.980**	**.962**
	128	**.960**	.956	.941	.670	.943	**.980**	**.962**
	256	**.970**	**.967**	.936	.709	.940	**.980**	.962
	512	**.979**	**.979**	.950	.652	.946	**.980**	.962
	all (65,536)	**.984**	–	**.969**	.645		**.980**	.962

Table 8: Graph classification, every method at every context size (ROC-AUC; K labeled graphs).

dataset	K	Graft	Graft FT	GCN	GIN	log. reg.
CIFAR10-sp-0v1	8	**.673**	**.670**	.602	.571	**.606**
	16	**.818**	**.799**	**.739**	.717	.602
	32	**.816**	**.834**	.794	**.810**	.774
	64	**.825**	**.817**	**.802**	.801	.799
	128	**.845**	**.896**	**.841**	.812	.810
	all (1,936)	**.849**	–	**.904**	**.881**	
IMDB-BINARY	8	**.611**	**.636**	**.567**	.564	.485
	16	**.586**	**.584**	**.499**	.472	.386
	32	**.712**	**.700**	**.703**	.690	.619
	64	**.734**	**.737**	.706	**.728**	.706
	128	**.732**	**.744**	.680	**.696**	.662
	all (700)	**.795**	**.837**	**.774**	.769	
MNIST-sp-0v1	8	.881	.861	**.989**	**.987**	**.957**
	16	**.991**	.937	.985	**.990**	**.985**
	32	**.996**	.988	**.999**	**.999**	.995

continued

dataset	K	Graft	Graft FT	GCN	GIN	log. reg.
	64	**.998**	.990	**1.00**	**.999**	.995
	128	**.998**	.995	**.999**	.997	**.997**
	all (2,051)	**.996**	–	**.994**	**.996**	
MNIST-sp-3v8	8	**.558**	**.544**	.490	.495	**.520**
	16	.648	**.660**	**.695**	**.724**	.509
	32	**.663**	**.729**	**.680**	.624	.508
	64	**.632**	**.752**	**.746**	.564	.456
	128	**.772**	**.892**	**.899**	.635	.548
	all (1,920)	**.786**	–	**.915**	**.687**	
molbbbp	8	**.696**	**.665**	.555	.574	**.630**
	16	**.657**	**.646**	.560	**.609**	.587
	32	**.718**	**.732**	.549	.612	**.638**
	64	**.737**	**.760**	.603	.602	**.636**
	128	**.755**	**.772**	.650	.630	**.675**
	all (844)	**.807**	**.811**	**.830**	.744	
molhiv	8	**.571**	**.553**	.508	.445	**.544**
	16	**.553**	**.601**	.513	.412	**.547**
	32	**.596**	**.604**	**.565**	.520	.554
	64	**.571**	**.557**	**.555**	.436	.551
	128	**.638**	**.614**	.512	.542	**.581**
	all (28,789)	**.674**	–	**.727**	**.718**	
MUTAG	8	**.898**	**.834**	.480	.634	**.717**
	16	**.911**	.863	.739	**.869**	**.954**
	32	**.912**	**.819**	.589	**.831**	.805
	64	**.923**	**.926**	.708	.899	**.950**
	128	**.939**	**.945**	.800	.935	**.957**
	all (132)	**.938**	**.941**	**.791**	.740	
NCI1	8	**.676**	**.678**	.512	**.616**	.557
	16	**.668**	**.645**	.509	.608	**.635**
	32	**.753**	**.761**	.577	**.717**	.685
	64	**.741**	**.695**	.581	**.698**	.687
	128	**.767**	**.752**	.662	**.742**	.718
	all (852)	**.762**	**.823**	.731	**.755**	
PROTEINS	8	**.735**	**.720**	.453	**.682**	.580
	16	**.697**	**.631**	.574	**.712**	.579
	32	**.754**	**.750**	.661	**.707**	.649
	64	**.717**	**.718**	.610	.687	**.736**
	128	**.735**	**.743**	.725	.733	**.735**
	all (774)	**.759**	**.747**	.719	**.761**	

Table 9: Node regression, every method at every context size (R^2; K labeled nodes).

dataset	K	Graft	Graft FT	GCN	GIN	ridge	Graph PFN	G2T-FM	Tab PFN
artnet-views	8	.152	**.185**	−.671	−.300	−.547	**.152**	**.278**	−.127
	16	**.120**	.077	−.168	−.293	−.879	**.202**	**.220**	−.142
	32	**.238**	.195	−.160	−.078	−1.56	**.367**	**.284**	.024
	64	.285	**.317**	−.330	−.268	−1.49	**.435**	**.352**	.091
	128	.355	**.371**	−.063	.196	−.015	**.526**	**.432**	.203
	256	**.355**	.325	−.009	.091	.079	**.537**	**.461**	.212
	512	.359	**.438**	.034	.255	.160	**.568**	**.512**	.262
	all (5,040)	.385	**.523**	.148	.386		**.608**	**.577**	.334
avazu-ctr	8	**.005**	**.005**	−.090	−3.11	−.097	−.055	**−.037**	−.217
	16	**.014**	−.055	**−.035**	−8.40	−.286	**.014**	−.060	−.055
	32	**.094**	**.129**	−.028	−8.79	−1.54	**−.023**	−.235	−.122

continued

dataset	K	Graft	Graft FT	GCN	GIN	ridge	Graph PFN	G2T-FM	Tab PFN
	64	**.137**	**.120**	.077	−1.59	−.742	**.106**	.027	−.018
	128	**.166**	**.130**	−.042	−1.11	−.435	**.149**	.094	.037
	256	**.228**	**.241**	.104	−.154	−.012	.223	**.229**	.050
	512	.246	**.255**	.166	−1188.94	.112	**.307**	**.267**	.146
	all (1,497)	.228	**.271**	.234	−.934		**.317**	**.301**	.203
city-roads-L	8	**.130**	−.134	−.183	−.267	−.793	**.012**	−.076	**.053**
	16	**.154**	−.134	−.339	−.205	−.684	**−.010**	**−.049**	−.157
	32	**.224**	.020	−.100	.047	−.167	**.290**	**.152**	.146
	64	.319	.283	−.172	−.060	−.281	**.408**	**.341**	**.355**
	128	.322	.291	.238	.206	−.075	**.434**	**.401**	**.416**
	256	.490	**.500**	.397	.335	.403	**.549**	.484	**.504**
	512	.504	.512	.460	.415	.465	**.603**	**.550**	**.529**
	all (1,261)	.554	**.586**	.533	.512		**.666**	.583	**.586**
city-roads-M	8	**−.078**	−.295	**−.183**	−.248	−.917	−.211	−.251	**−.174**
	16	.063	−.024	−.353	−.244	−1.26	**.125**	**.089**	**.116**
	32	.012	−.049	−.272	−.165	−.537	**.121**	**.056**	**.134**
	64	.140	.002	−.135	−.158	−.626	**.330**	**.333**	**.254**
	128	.195	.154	.087	.057	−.730	**.364**	**.358**	**.333**
	256	.339	.374	.357	.118	.179	**.481**	**.432**	**.421**
	512	.364	.505	.454	.310	.292	**.565**	**.542**	**.514**
	all (3,610)	.458	.567	.555	.503		**.644**	**.582**	**.577**
hm-prices	8	−.395	−.408	**−.231**	−8.37	−.320	**−.209**	−.296	**−.194**
	16	**−.050**	−.319	−.295	−3.08	−10.59	**.001**	−.256	**−.233**
	32	**.022**	−.082	−.102	−.033	−5.67	**.118**	−.006	**.111**
	64	.196	**.200**	.079	−.544	−24.92	**.255**	.055	**.209**
	128	.349	**.367**	.274	−.330	.043	**.485**	.270	**.371**
	256	.361	.371	.299	−2.53	.266	**.580**	**.427**	**.481**
	512	.408	.446	.373	−1.26	.331	**.668**	**.551**	**.576**
	all (1,510)	.433	.513	.526	−.016		**.763**	**.680**	**.649**
twitch-views	8	**.631**	**.588**	−.168	−.512	−1.17	.060	**.196**	−.249
	16	**.673**	**.628**	−.158	−.315	−.316	**.494**	.493	−.188
	32	**.709**	**.695**	−.292	.145	−2.56	**.547**	.522	−.040
	64	**.699**	**.679**	−92.16	−.106	−.081	**.688**	.620	−.031
	128	**.708**	**.705**	−2.14	.162	−1.45	**.681**	.628	−.012
	256	**.742**	**.726**	.022	.139	−.152	**.743**	.705	−.005
	512	.738	**.741**	.349	.283	.155	**.752**	**.738**	.008
	all (1,530)	**.731**	.724	.554	−2.75		**.750**	**.730**	−.001

Table 10: Edge regression, every method at every context size (R^2; K labeled pairs).

dataset	K	Graft	Graft FT	GCN	GIN	ridge
BitcoinAlpha	8	**−.127**	**−.105**	**−.206**	−4.79	−.260
	16	**−.188**	**−.239**	**−.145**	−5.29	−.261
	32	**−.039**	**−.055**	−.277	−61.12	**−.091**
	64	**−.030**	**−.043**	−78.32	−1.08	**−.030**
	128	**−.025**	−.057	**−.043**	−1.07	**.007**
	256	**.003**	**−.030**	−.092	−2.28	**.032**
	512	−.020	**−.014**	**−.008**	−2.67	**−.001**
	all (11,300)	**−.003**	**−.026**	**.018**	−.315	
BitcoinOTC	8	**−.066**	**−.050**	**−.145**	−10711.60	−.568
	16	**−.119**	−.614	**−.360**	−5.39	**−.189**
	32	**.014**	−.341	**−.059**	−44.37	**−.090**
	64	**.097**	**−.124**	−.464	−.283	**.006**
	128	**.057**	−.173	**−.041**	−.459	**−.035**
	256	**.075**	**.076**	−.011	−1.32	**−.008**

continued

dataset	K	Graft	Graft FT	GCN	GIN	ridge
	512	**.201**	**.085**	.011	−1.31	**.021**
	all (17,194)	**.188**	**.344**	**.183**	−.069	
MovieLens100K	8	**−.325**	**−.565**	**−.383**	−673.26	−.911
	16	**−.269**	**−.386**	**−.484**	−21.99	−.935
	32	**−.213**	**−.249**	**−.119**	−63.54	−.850
	64	**−.215**	**−.058**	**−.088**	−10.68	−1.38
	128	**−.009**	**.027**	**−.186**	−16.46	−2.94
	256	**.056**	**.063**	**−.079**	−59.73	−.204
	512	**.105**	**.070**	−.076	−28.40	**−.068**
	all (64,000)	**.181**	–	**.179**	**−18.86**	

Table 11: Graph regression, every method at every context size (R^2; K labeled graphs).

dataset	K	Graft	Graft FT	GCN	GIN	ridge
AQSOL	8	**−.413**	**−.733**	−16.33	**−1.30**	−2.18
	16	**.214**	**.094**	−1.45	−4.46	**−.965**
	32	**.224**	**.241**	**−.061**	−.064	−.244
	64	**.396**	**.455**	−.235	.171	**.236**
	128	**.443**	**.434**	.169	**.347**	.270
	all (2,400)	**.497**	**.710**	.331	**.535**	
ESOL	8	**.499**	**.438**	−.552	**.070**	.035
	16	**.573**	**.594**	−.038	**.104**	−.179
	32	**.609**	**.461**	−.198	**.388**	.131
	64	**.622**	**.632**	−.344	**.577**	.157
	128	**.670**	**.655**	.356	**.692**	.270
	all (270)	**.692**	**.630**	.591	**.793**	
FreeSolv	8	**.074**	**−.150**	**−.291**	−.409	−1.33
	16	**−.007**	**.176**	−.589	**−.399**	−.917
	32	**.429**	**.341**	**.080**	−.741	−.196
	64	**.522**	**.466**	**.179**	−.579	−.639
	128	**.592**	**.580**	**.297**	.224	−.143
	all (154)	**.603**	**.556**	**.409**	.388	
Lipo	8	**−.075**	**−.214**	**−.198**	−.267	−.562
	16	**.022**	**−.111**	**−.167**	−.220	−.510
	32	**.173**	**−.025**	−.170	−.031	**−.012**
	64	**.066**	**−.125**	**−.081**	−.175	−.165
	128	**.225**	**.225**	−.012	.032	**.093**
	all (2,400)	**.183**	–	**.043**	**.076**	
ZINC	8	**.291**	**.257**	.026	−.137	**.087**
	16	**.432**	**.368**	−1.35	−1.39	**.020**
	32	**.480**	**.428**	−.078	**.097**	−.125
	64	**.383**	**.263**	.032	.057	**.143**
	128	**.503**	**.552**	**.233**	.226	.114
	all (2,400)	**.385**	–	**.331**	**.273**	

F BASELINES, AND HOW EACH WAS CONFIGURED

A GNN baseline is given exactly the labels Graft reads: the K labeled rows of the prompt, which it splits in two halves, stratified where there are classes; GCN and GIN are each trained on one half with early stopping on the other, the better of the two is chosen on that half, and it is refitted on all K. Below four labels per class, where a split is impossible, it trains for a fixed number of epochs (GCN at node and edge level, GIN at graph level). The closed forms (ridge and logistic regression over raw and propagated features) are selected on the same split. The full-split GNN is trained on the dataset's training split and selected on its validation split. Each GNN number is the mean of five context draws. The other in-context models are run from their released code and checkpoints, on the datasets they fit. Table 12 gives the configuration; there is no hyperparameter search. Each variant trains for its full epoch budget on one half of the labels, the epoch with the best score on the other half is kept, and the chosen variant is refitted on all K for that many epochs. Regression targets are z-scored on the training labels.

Table 12: GNN baselines (GCN and GIN), per task type: two message-passing layers of width 64 in every case.

	node cls.	link	graph cls.	node reg.	edge reg.	graph reg.
readout	node	dot product	mean / sum pool	node MLP	pair MLP	mean / sum pool
dropout	0.5	0	0	0	0	0
optimizer	Adam	Adam	AdamW	AdamW	AdamW	AdamW
learning rate	10^{-2}	10^{-2}	10^{-2}	10^{-2}	10^{-2}	10^{-2}
weight decay	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	10^{-4}	10^{-4}	10^{-4}	10^{-4}
epochs	300	300	200	300	300	300
loss	cross-entropy	BCE	weighted BCE	MSE	MSE	MSE

G COMPUTATIONAL COST

Each of the three pretraining seeds trains for 230k steps on one V100 GPU, in 21 to 22 hours. At inference, one pass answers every query of a prompt (one per class for a multiclass target); Table 13 sets it against training and selecting the GNN on the same labels, which takes 4 to 21 times longer, and Graft's time is nearly flat in K.

Table 13: Wall-clock seconds for one Graft forward pass and for training and selecting the GNN on the same K labels (GCN and GIN on the split, selection, refit), per draw, on one V100; one representative dataset per task type. For node classification K counts all labels (8 classes: 4, 16 and 64 per class); Graft's time includes one pass per class. Data loading, PCA and closed forms are excluded from both, so these are network times, not end-to-end latency.

		$K = 32$		$K = 128$		$K = 512$	
task type	dataset	Graft	GNN	Graft	GNN	Graft	GNN
node classification	Photo	0.81	4.7	0.85	3.8	0.99	4.5
node regression	city-roads-M	0.48	3.9	0.47	4.0	0.48	4.0
link prediction	Computers	0.20	4.1	0.20	3.3	0.21	3.3
edge regression	BitcoinOTC	0.11	1.8	0.11	2.0	0.11	1.8
graph classification	NCI1	0.12	1.4	0.14	1.4	0.23	1.7
graph regression	ZINC	0.12	1.6	0.14	2.3	0.18	2.3

H FINE-TUNING

Graft is used frozen throughout the main results, but its weights can also be trained on a target. We ask two questions (Table 14). *Same labels*: given the K labels of a prompt, is it better to read them in context or to fine-tune on them? We fine-tune under the GNN's protocol: the K labels are split in half, the number of steps is chosen by early stopping on the held-out half (zero steps is a candidate),

and the model is refitted on all K for that many steps. *Unseen dataset*: when the dataset's family was withheld from pretraining, how much of the cost does fine-tuning on the dataset's training labels recover? Here the fold model is fine-tuned with early stopping on the dataset's validation split. With a few labels, reading them in context is better than training on them at every task type, because the held-out half is too small to choose a number of steps; with hundreds of labels a short fine-tune helps regression slightly and a long one overfits: the fine-tuned model matches the in-context one on fewer than half of the cells at every task type, and both stay ahead of the GNN trained on the same labels. On an unseen dataset with its whole training split, fine-tuning helps on 165 of 189 dataset–context cells and closes most of the gap to the model that saw the family (Figure 4b); once both are fine-tuned, the model that saw the family stays ahead on most datasets by a few hundredths. Fine-tuning therefore pays when the target is out of domain and labels are plentiful, and the frozen model is the better choice in the few-label regime it was built for. GraphPFN, G2T-FM and TabPFN v2 also report gains from fine-tuning (Eremeev et al., 2025b;a; Hollmann et al., 2025).

Table 14: Fine-tuning, per task type, in normalized score. *Same K labels*: Graft in context, Graft fine-tuned on those labels, and the GNN trained on them, averaged over every context size where all three are scored (each dataset's score floored at -1), and the share of cells where fine-tuning matches or beats reading the labels in context. *Unseen dataset*: the fold model pretrained without the dataset's family, in context and fine-tuned on the dataset's training labels, at the budget of Table 2, and the GNN trained on the full split.

	same K labels, every K				unseen dataset, all its labels		
task type	in context	fine-tuned	GNN	FT $\geq$ ICL	in context	fine-tuned	GNN, full
node classification	0.38	0.36	0.34	19/65	0.48	0.52	0.52
node regression	0.29	0.26	−0.08	25/60	0.21	0.33	0.48
link prediction	0.89	0.87	0.69	20/91	0.86	0.91	0.83
edge regression	−0.05	−0.13	−0.45	7/21	−0.39	0.13	0.15
graph classification	0.51	0.51	0.42	24/51	0.51	0.62	0.52
graph regression	0.32	0.25	−0.09	7/25	0.10	0.44	0.52